man pages section 1M: System Administration Commands

Part No: 816–5166–16
September 2010

ORACLE

110417@25097

Contents

Preface

Both novice users and those familar with the SunOS operating system can use online man pages to obtain information about the system and its features. A man page is intended to answer concisely the question "What does it do?" The man pages in general comprise a reference manual. They are not intended to be a tutorial.

Overview

The following contains a brief description of each man page section and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.
- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5 contains miscellaneous documentation such as character-set tables.
- Section 6 contains available games and demos.
- Section 7 describes various special files that refer to specific hardware peripherals and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.
- Section 9 provides reference information needed to write device drivers in the kernel environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver/Kernel Interface (DKI).
- Section 9E describes the DDI/DKI, DDI-only, and DKI-only entry-point routines a developer can include in a device driver.
- Section 9F describes the kernel functions available for use by device drivers.
- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the `intro` pages for more information and detail about each section, and `man(1)` for more information about man pages in general.

NAME — This section gives the names of the commands or functions documented, followed by a brief description of what they do.

SYNOPSIS — This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full path name is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

- [ ] — Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.
- . . . — Ellipses. Several values can be provided for the previous argument, or the previous argument can be specified multiple times, for example, "`filename...`".
- | — Separator. Only one of the arguments separated by this character can be specified at a time.
- { } — Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

PROTOCOL — This section occurs only in subsection 3R to indicate the protocol description file.

DESCRIPTION — This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, and functions are described under USAGE.

IOCTL — This section appears on pages in Section 7 only. Only the device class that supplies appropriate parameters to the `ioctl(2)` system call is called `ioctl` and generates its own

	heading. `ioctl` calls for a specific device are listed alphabetically (on the man page for that specific device). `ioctl` calls are used for a particular class of devices all of which have an `io` ending, such as `mtio(7I)`.
OPTIONS	This secton lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.
OPERANDS	This section lists the command operands and describes how they affect the actions of the command.
OUTPUT	This section describes the output – standard output, standard error, or output files – generated by the command.
RETURN VALUES	If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared void do not return values, so they are not discussed in RETURN VALUES.
ERRORS	On failure, most functions place an error code in the global variable `errno` indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than one condition can cause the same error, each condition is described in a separate paragraph under the error code.
USAGE	This section lists special rules, features, and commands that require in-depth explanations. The subsections listed here are used to explain built-in functionality: Commands Modifiers Variables Expressions Input Grammar

EXAMPLES	This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command-line entry and machine response is shown. Whenever an example is given, the prompt is shown as `example%`, or if the user must be superuser, `example#`. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS, and USAGE sections.
ENVIRONMENT VARIABLES	This section lists any environment variables that the command or function affects, followed by a brief description of the effect.
EXIT STATUS	This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion, and values other than zero for various error conditions.
FILES	This section lists all file names referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.
ATTRIBUTES	This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See `attributes(5)` for more information.
SEE ALSO	This section lists references to other man pages, in-house documentation, and outside publications.
DIAGNOSTICS	This section lists diagnostic messages with a brief explanation of the condition causing the error.
WARNINGS	This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.
NOTES	This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here.
BUGS	This section describes known bugs and, wherever possible, suggests workarounds.

REFERENCE

Introduction

Name Intro – introduction to maintenance commands and application programs

Description This section describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.

Because of command restructuring for the Virtual File System architecture, there are several instances of multiple manual pages that begin with the same name. For example, the `mount`, pages – `mount(1M)`, `mount_cachefs(1M)`, `mount_hsfs(1M)`, `mount_nfs(1M)`, `mount_tmpfs(1M)`, and `mount_ufs(1M)`. In each such case the first of the multiple pages describes the syntax and options of the generic command, that is, those options applicable to all FSTypes (file system types). The succeeding pages describe the functionality of the FSType-specific modules of the command. These pages list the command followed by an underscore (_) and the FSType to which they pertain. Note that the administrator should not attempt to call these modules directly. The generic command provides a common interface to all of them. Thus the FSType-specific manual pages should not be viewed as describing distinct commands, but rather as detailing those aspects of a command that are specific to a particular FSType.

Command Syntax Unless otherwise noted, commands described in this section accept options and other arguments according to the following syntax:

name `[`*option*`(`*s*`)] [`*cmdarg*`(`*s*`)]`

where:

name	The name of an executable file.
option	– *noargletter*(*s*) or, – *argletter*< >*optarg* where < > is optional white space.
noargletter	A single letter representing an option without an argument.
argletter	A single letter representing an option requiring an argument.
optarg	Argument (character string) satisfying preceding *argletter*.
cmdarg	Pathname (or other command argument) *not* beginning with – or, – by itself indicating the standard input.

Attributes See `attributes(5)` for a discussion of the attributes listed in this section.

See Also `getopt(1)`, `getopt(3C)`, `attributes(5)`

Diagnostics Upon termination, each command returns 0 for normal termination and non-zero to indicate troubles such as erroneous parameters, bad or inaccessible data, or other inability to cope with the task at hand. It is called variously "exit code," "exit status," or "return code," and is described only where special conventions are involved.

Notes Unfortunately, not all commands adhere to the standard syntax.

REFERENCE

System Administration Commands - Part 1

Name 6to4relay – administer configuration for 6to4 relay router communication

Synopsis

```
/usr/sbin/6to4relay

/usr/sbin/6to4relay [-e] [-a addr]

/usr/sbin/6to4relay [-d]

/usr/sbin/6to4relay [-h]
```

Description The `6to4relay` command is used to configure 6to4 relay router communication. Relay router communication support is enabled by setting the value of a variable that stores an IPv4 address within the `tun` module. This variable is global to all tunnels and defines the policy for communication with relay routers. By default, the address is set to `INADDR_ANY` (`0.0.0.0`), and the kernel interprets the value to indicate that support for relay router communication is disabled. Otherwise, support is enabled, and the specified address is used as the IPv4 destination address when packets destined for native IPv6 (non-6to4) hosts are sent through the 6to4 tunnel interface. The `6to4relay` command uses a project private ioctl to set the variable.

`6to4relay` used without any options outputs the current, in-kernel, configuration status. Use the `-a` option to send packets to a specific relay router's `unicast` address instead of the default `anycast` address. The address specified with the `-a` option does not specify the policy for receiving traffic from relay routers. The source relay router on a received packet is non-deterministic, since a different relay router may be chosen for each sending native IPv6 end-point.

Configuration changes made by using the `6to4relay` are not persistent across reboot. The changes will persist in the kernel only until you take the tunnel down

Options The `6to4relay` command supports the following options:

`-a` *addr* Use the specified address, *addr*.

`-e` Enable support for relay router. Use `-a` *addr* if it is specified. Otherwise, use the default `anycast` address, 192.88.99.1.

`-d` Disable support for the relay router.

`-h` Help

Operands The following operands are supported:

addr A specific relay router's unicast address. *addr* must be specified as a dotted decimal representation of an IPv4 address. Otherwise, an error will occur, and the command will fail.

Examples **EXAMPLE 1** Printing the In-Kernel Configuration Status

Use `/usr/sbin/6to4relay` without any options to print the in-kernel configuration status.

```
example# /usr/sbin/6to4relay
```

EXAMPLE 1 Printing the In-Kernel Configuration Status *(Continued)*

If 6to4 relay router communication is disabled, the administrator will see the following message:

```
6to4relay: 6to4 Relay Router communication support is disabled.
```

If 6to4 router communication is enabled, the user will see this message:

```
6to4relay: 6to4 Relay Router communication support is enabled.
IPv4 destination address of Relay Router = 192.88.99.1
```

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/usr/sbin/6to4relay` The default installation root

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also `ifconfig(1M)`, `attributes(5)`

Huitema, C. *RFC 3068, An Anycast Prefix for 6to4 Relay Routers*. Network Working Group. June, 2001.

Carpenter, B. and Moore, K. *RFC 3056, Connection of IPv6 Domains via IPv4 Clouds*. Network Working Group. February, 2001.

Diagnostics The `6to4relay` reports the following messages:

```
6to4relay: input (0.0.0.0) is not a valid IPv4 unicast address
```

Example: The following example provides an incorrect unicast address.

```
example# 6to4relay -e -a 0.0.0.0
```

Description: The address specified with the `-a` option must be a valid unicast address.

```
6to4relay: option requires an argument —a
usage:
    6to4relay
    6to4relay -e [-a <addr>]
```

```
    6to4relay -d
    6to4relay -h
```

Example: The following example does not include an argument for the `-a` option.

```
example# 6to4relay -e -a
```

Description: The `-a` option requires an argument.

```
usage:
    6to4relay
    6to4relay -e [-a <addr>]
    6to4relay -d
    6to4relay -h
```

Example: The following example specifies options that are not permitted.

```
example# 6to4relay -e -d
```

Description: The options specified are not permitted. A usage message is output to the screen.

```
usage:
    6to4relay
    6to4relay -e [-a <addr>]
    6to4relay -d
    6to4relay -h
```

Example: The following example specifies the `-a` option without specifying the `-e` option.

```
example# 6to4relay -a 1.2.3.4
```

Description: The `-e` option is required in conjunction with the `-a` option. A usage message is output to the screen.

```
6to4relay: ioctl (I_STR) : Invalid argument
```

Example: The following example specifies an invalid address.

```
example# 6to4relay -e -a 239.255.255.255
```

Description: The address specified with the `-a` option must not be a class d *addr*.

Name accept, reject – accept or reject print requests

Synopsis

```
accept destination...

reject [-r reason] destination...
```

Description

`accept` allows the queueing of print requests for the named destinations.

`reject` prevents queueing of print requests for the named destinations.

Use `lpstat -a` to check if destinations are accepting or rejecting print requests.

Generally, `accept` and `reject` are run on the print server to control local print queues. Under some configurations, `accept` and `reject` are run on client systems when IPP is being used to communicate between client and server.

Options The following options are supported for `reject`:

`-r` *reason* Assigns a reason for rejection of print requests for *destination*.

reason is reported by `lpstat -a`. By default, *reason* is `unknown reason` for existing destinations, and `new printer` for destinations added to the system but not yet accepting requests. Enclose *reason* in quotes if it contains blanks.

Operands The following operands are supported:

destination The name of the destination accepting or rejecting print requests. Destination specifies the name of a printer or class of printers (see `lpadmin(1M)`). Specify *destination* using atomic name or URI-style (scheme://endpoint) names. See `printers.conf(4)` for information regarding the naming conventions for destinations.

Exit Status The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Files

`/etc/printers.conf`	System printer configuration database
`$HOME/.printers`	User-configurable printer database
`ou=printers`	LDAP version of `/etc/printers.conf`
`printers.conf.byname`	NIS version of `/etc/printers.conf`
`printers.org_dir`	NIS+ version of `/etc/printers.conf`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlp-cmds
CSI	Enabled. See NOTES.
Interface Stability	Obsolete

See Also enable(1), lp(1), lpstat(1), lpadmin(1M), lpsched(1M), printers.conf(4), attributes(5)

Notes accept and reject affect only queueing on the print server's spooling system. Requests made from a client system remain queued in the client system's queueing mechanism until they are cancelled or accepted by the print server's spooling system.

accept is CSI-enabled except for the *destination* name.

When IPP is in use, the user is prompted for a passphrase if the remote print service is configured to require authentication.

Name acct, acctdisk, acctdusg, accton, acctwtmp, closewtmp, utmp2wtmp – overview of accounting and miscellaneous accounting commands

Synopsis

```
/usr/lib/acct/acctdisk

/usr/lib/acct/acctdusg [-u filename] [-p filename]

/usr/lib/acct/accton [filename]

/usr/lib/acct/acctwtmp reason filename

/usr/lib/acct/closewtmp

/usr/lib/acct/utmp2wtmp
```

Description Accounting software is structured as a set of tools (consisting of both C programs and shell procedures) that can be used to build accounting systems. acctsh(1M) describes the set of shell procedures built on top of the C programs.

Connect time accounting is handled by various programs that write records into /var/adm/wtmpx, as described in utmpx(4). The programs described in acctcon(1M) convert this file into session and charging records, which are then summarized by acctmerg(1M).

Process accounting is performed by the system kernel. Upon termination of a process, one record per process is written to a file (normally /var/adm/pacct). The programs in acctprc(1M) summarize this data for charging purposes; acctcms(1M) is used to summarize command usage. Current process data may be examined using acctcom(1).

Process accounting records and connect time accounting records (or any accounting records in the tacct format described in acct.h(3HEAD)) can be merged and summarized into total accounting records by acctmerg (see tacct format in acct.h(3HEAD)). prtacct (see acctsh(1M)) is used to format any or all accounting records.

acctdisk reads lines that contain user ID, login name, and number of disk blocks and converts them to total accounting records that can be merged with other accounting records. acctdisk returns an error if the input file is corrupt or improperly formatted.

acctdusg reads its standard input (usually from find / -print) and computes disk resource consumption (including indirect blocks) by login.

accton without arguments turns process accounting off. If *filename* is given, it must be the name of an existing file, to which the kernel appends process accounting records (see acct(2) and acct.h(3HEAD)).

acctwtmp writes a utmpx(4) record to *filename*. The record contains the current time and a string of characters that describe the *reason*. A record type of ACCOUNTING is assigned (see utmpx(4)) *reason* must be a string of 11 or fewer characters, numbers, $, or spaces. For example, the following are suggestions for use in reboot and shutdown procedures, respectively:

```
acctwtmp "acctg on" /var/adm/wtmpx
acctwtmp "acctg off" /var/adm/wtmpx
```

For each user currently logged on, `closewtmp` puts a false `DEAD_PROCESS` record in the `/var/adm/wtmpx` file. `runacct` (see `runacct(1M)`) uses this false `DEAD_PROCESS` record so that the connect accounting procedures can track the time used by users logged on before `runacct` was invoked.

For each user currently logged on, `runacct` uses `utmp2wtmp` to create an entry in the file `/var/adm/wtmpx`, created by `runacct`. Entries in `/var/adm/wtmpx` enable subsequent invocations of `runacct` to account for connect times of users currently logged in.

Options The following options are supported:

`-u` *filename* Places in *filename* records consisting of those filenames for which `acctdusg` charges no one (a potential source for finding users trying to avoid disk charges).

`-p` *filename* Specifies a password file, *filename*. This option is not needed if the password file is `/etc/passwd`.

Environment Variables If any of the `LC_*` variables (`LC_TYPE`, `LC_MESSAGES`, `LC_TIME`, `LC_COLLATE`, `LC_NUMERIC`, and `LC_MONETARY`) (see `environ(5)`) are not set in the environment, the operational behavior of `acct` for each corresponding locale category is determined by the value of the `LANG` environment variable. If `LC_ALL` is set, its contents are used to override both the `LANG` and the other `LC_*` variables. If none of the above variables are set in the environment, the "C" (U.S. style) locale determines how `acct` behaves.

`LC_CTYPE` Determines how `acct` handles characters. When `LC_CTYPE` is set to a valid value, `acct` can display and handle text and filenames containing valid characters for that locale. `acct` can display and handle Extended Unix Code (EUC) characters where any character can be 1, 2, or 3 bytes wide. `acct` can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

`LC_TIME` Determines how `acct` handles date and time formats. In the "C" locale, date and time handling follows the U.S. rules.

Files

`/etc/passwd` Used for login name to user ID conversions.

`/usr/lib/acct` Holds all accounting commands listed in sub-class 1M of this manual.

`/var/adm/pacct` Current process accounting file.

`/var/adm/wtmpx` History of user access and administration information..

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

See Also acctcom(1), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), fwtmp(1M), runacct(1M), acct(2), acct.h(3HEAD), passwd(4), utmpx(4), attributes(5), environ(5)

System Administration Guide: Basic Administration

Name acctadm – configure extended accounting facility

Synopsis

```
/usr/sbin/acctadm [-DErux] [-d resource_list]
     [-e resource_list] [-f filename]
     [task | process | flow]
```

Description `acctadm` configures various attributes of the extended accounting facility. Without arguments, `acctadm` displays the current status of the extended accounting facility.

Options The following options are supported:

`-d` *resource_list* Disable reporting of resource usage for resource. Specify *resource_list* as a comma-separated list of resources or resource groups.

This option requires an operand. See `OPERANDS`.

`-D` Disable accounting of the given operand type without closing the accounting file. This option can be used to temporarily stop writing accounting records to the accounting file without closing it. To close the file use the `-x` option. See `-x`.

`-e` *resource_list* Enable reporting of resource usage for resource. Specify *resource_list* as a comma-separated list of resources or resource groups.

This option requires an operand. See `OPERANDS`.

`-E` Enable accounting of the given operand type without sending the accounting output to a file. This option requires an operand. See `OPERANDS`.

`-f` *filename* Send the accounting output for the given operand type to *filename*. If *filename* exists, its contents are lost.

This option requires an operand. See `OPERANDS`.

`-r` Display available resource groups.

When this option is used with an operand, it displays resource groups available for a given accounting type. When no operand is specified, this option displays resource groups for all available accounting types. See OPERANDS.

`-u` Configure accounting based on the contents of `/etc/acctadm.conf`.

`-x` Deactivate accounting of the given operand type. This option also closes the accounting file for the given accounting type if it is currently open.

This option requires an operand. See `OPERANDS`.

Operands The `-d`, `-D`, `-e`, `-E`, `-f`, and `-x` options require an operand.

The following operands are supported:

process Run `acctadm` on the process accounting components of the extended accounting facility.

task Run `acctadm` on the task accounting components of the extended accounting facility.

flow Run `acctadm` on the IPQoS accounting components of the extended accounting facility.

The optional final parameter to `acctadm` represents whether the command should act on the process, system task or IPQoS accounting components of the extended accounting facility.

Examples **EXAMPLE 1** Displaying the Current Status

The following command displays the current status. In this example, system task accounting is active and tracking only CPU resources. Process and flow accounting are not active.

```
$ acctadm
           Task accounting: active
      Task accounting file: /var/adm/exacct/task
    Tracked task resources: extended
  Untracked task resources: host
        Process accounting: inactive
   Process accounting file: none
 Tracked process resources: none
Untracked process resources: extended,host
           Flow accounting: inactive
      Flow accounting file: none
    Tracked flow resources: none
  Untracked flow resources: extended
```

EXAMPLE 2 Activating Basic Process Accounting

The following command activates basic process accounting:

```
$ acctadm -e basic -f /var/adm/exacct/proc process
```

EXAMPLE 3 Displaying Available Resource Groups

The following command displays available resource groups:

```
$ acctadm -r
  process:
  extended pid,uid,gid,cpu,time,command,tty,projid, \
  taskid,ancpid,wait-status,zone,flag,memory,mstate
  basic    pid,uid,gid,cpu,time,command,tty,flag
  task:
```

EXAMPLE 3 Displaying Available Resource Groups *(Continued)*

```
  extended taskid,projid,cpu,time,host,mstate,anctaskid,zone
  basic    taskid,projid,cpu,time
  flow:
  extended saddr,daddr,sport,dport,proto,dsfield,nbytes,npkts, \
  action,ctime,lseen,projid,uid
  basic    saddr,daddr,sport,dport,proto,nbytes,npkts,action
```

In the output above, the lines beginning with `extended` are shown with a backslash character. In actual `acctadm` output, these lines are displayed as unbroken, long lines.

EXAMPLE 4 Displaying Resource Groups for Task Accounting

The following command displays resource groups for task accounting:

```
$ acctadm -r task
  extended taskid,projid,cpu,time,host,mstate,anctaskid,zone
  basic    taskid,projid,cpu,time
```

Exit Status

The following exit values are returned:

`0` Successful completion.

The modifications to the current configuration were valid and made successfully.

`1` An error occurred.

A fatal error occured either in obtaining or modifying the accounting configuration.

`2` Invalid command line options were specified.

Files

`/etc/acctadm.conf`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`acct(2)`, `attributes(5)`, `ipqos(7ipp)`

Notes

Both extended accounting and regular accounting can be active.

Available resources can vary from system to system, and from platform to platform.

Name acctcms – command summary from process accounting records

Synopsis

```
/usr/lib/acct/acctcms [-a [-o] [-p]] [-c] [-j] [-n] [-s]
     [-t] filename...
```

Description `acctcms` reads one or more *filename*s, normally in the form described in `acct.h`(3HEAD). It adds all records for processes that executed identically named commands, sorts them, and writes them to the standard output, normally using an internal summary format.

Options

`-a` Print output in ASCII rather than in the internal summary format. The output includes command name, number of times executed, total kcore-minutes, total CPU minutes, total real minutes, mean size (in K), mean CPU minutes per invocation, "hog factor," characters transferred, and blocks read and written, as in `acctcom`(1). Output is normally sorted by total kcore-minutes.

Use the following options only with the `-a` option:

`-o` Output a (non-prime) offshift-time-only command summary.

`-p` Output a prime-time-only command summary.

When `-o` and `-p` are used together, a combination prime-time and non-prime-time report is produced. All the output summaries are total usage except number of times executed, CPU minutes, and real minutes, which are split into prime and non-prime.

`-c` Sort by total CPU time, rather than total kcore-minutes.

`-j` Combine all commands invoked only once under "***other".

`-n` Sort by number of command invocations.

`-s` Any file names encountered hereafter are already in internal summary format.

`-t` Process all records as total accounting records. The default internal summary format splits each field into prime and non-prime-time parts. This option combines the prime and non-prime time parts into a single field that is the total of both, and provides upward compatibility with old style `acctcms` internal summary format records.

Examples **EXAMPLE 1** Using the `acctcms` command.

A typical sequence for performing daily command accounting and for maintaining a running total is:

```
example% acctcms filename ... > today
example% cp total previoustotal
example% acctcms -s today previoustotal > total
example% acctcms -a -s today
```

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

See Also acctcom(1), acct(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), fwtmp(1M), runacct(1M), acct(2), acct.h(3HEAD), utmpx(4), attributes(5)

Notes Unpredictable output results if `-t` is used on new style internal summary format files, or if it is not used with old style internal summary format files.

Name acctcon, acctcon1, acctcon2 – connect-time accounting

Synopsis

```
/usr/lib/acct/acctcon [-l lineuse] [-o reboot]

/usr/lib/acct/acctcon1 [-p] [-t] [-l lineuse] [-o reboot]

/usr/lib/acct/acctcon2
```

Description `acctcon` converts a sequence of login/logoff records to total accounting records (see the `tacct` format in `acct.h`(3HEAD)). The login/logoff records are read from standard input. The file `/var/adm/wtmpx` is usually the source of the login/logoff records; however, because it might contain corrupted records or system date changes, it should first be fixed using `wtmpfix`. The fixed version of file `/var/adm/wtmpx` can then be redirected to `acctcon`. The `tacct` records are written to standard output.

`acctcon` is a combination of the programs `acctcon1` and `acctcon2`. `acctcon1` converts login/logoff records, taken from the fixed `/var/adm/wtmpx` file, to ASCII output. `acctcon2` reads the ASCII records produced by `acctcon1` and converts them to `tacct` records. `acctcon1` can be used with the `-l` and `-o` options, described below, as well as with the `-p` and `-t` options.

Options

`-p` Print input only, showing line name, login name, and time (in both numeric and date/time formats).

`-t` `acctcon1` maintains a list of lines on which users are logged in. When it reaches the end of its input, it emits a session record for each line that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session still in progress. The `-t` flag causes it to use, instead, the last time found in its input, thus assuring reasonable and repeatable numbers for non-current files.

`-l` *lineuse* *lineuse* is created to contain a summary of line usage showing line name, number of minutes used, percentage of total elapsed time used, number of sessions charged, number of logins, and number of logoffs. This file helps track line usage, identify bad lines, and find software and hardware oddities. Hangup, termination of `login`(1) and termination of the login shell each generate logoff records, so that the number of logoffs is often three to four times the number of sessions. See `init`(1M) and `utmpx`(4).

`-o` *reboot* `reboot` is filled with an overall record for the accounting period, giving starting time, ending time, number of reboots, and number of date changes.

Examples **EXAMPLE 1** Using the `acctcon` command.

The `acctcon` command is typically used as follows:

```
example% acctcon -l lineuse -o reboots < tmpwtmp > ctacct
```

The `acctcon1` and `acctcon2` commands are typically used as follows:

EXAMPLE 1 Using the `acctcon` command. *(Continued)*

```
example% acctcon1 -l lineuse -o reboots < tmpwtmp | sort +1n +2 > ctmp
example% acctcon2 < ctmp > ctacct
```

Files `/var/adm/wtmpx` History of user access and administration information

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

See Also `acctcom(1)`, `login(1)`, `acct(1M)`, `acctcms(1M)`, `acctmerg(1M)`, `acctprc(1M)`, `acctsh(1M)`, `fwtmp(1M)`, `init(1M)`, `runacct(1M)`, `acct(2)`, `acct.h(3HEAD)`, `utmpx(4)`, `attributes(5)`

System Administration Guide: Basic Administration

Notes The line usage report is confused by date changes. Use `wtmpfix` (see `fwtmp(1M)`), with the `/var/adm/wtmpx` file as an argument, to correct this situation.

During a single invocation of any given command, the `acctcon`, `acctcon1`, and `acctcon2` commands can process a maximum of:

- 6000 distinct session
- 1000 distinct terminal lines
- 2000 distinct login names

If at some point the actual number of any one of these items exceeds the maximum, the command will not succeed.

Name acctmerg – merge or add total accounting files

Synopsis

```
/usr/lib/acct/acctmerg [-a] [-i] [-p] [-t] [-u] [-v]
     [filename] ...
```

Description `acctmerg` reads its standard input and up to nine additional files, all in the `tacct` format (see `acct.h`(3HEAD)) or an ASCII version thereof. It merges these inputs by adding records whose keys (normally user ID and name) are identical, and expects the inputs to be sorted on those keys.

Options

`-a` Produce output in ASCII version of `tacct`.

`-i` Produce input in ASCII version of `tacct`.

`-p` Print input with no processing.

`-t` Produce a single record that totals all input.

`-u` Summarize by user ID, rather than by user ID and name.

`-v` Produce output in verbose ASCII format, with more precise notation for floating-point numbers.

Examples **EXAMPLE 1** Using the `acctmerg` command.

The following sequence is useful for making "repairs" to any file kept in this format:

```
example% acctmerg   -v  <filename1  >filename2
```

Edit *filename2* as you want:

```
example% acctmerg   -i  <filename2  >filename1
```

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

See Also `acctcom`(1), `acct`(1M), `acctcms`(1M), `acctcon`(1M), `acctprc`(1M), `acctsh`(1M), `fwtmp`(1M), `runacct`(1M), `acct`(2), `acct.h`(3HEAD), `utmpx`(4), `attributes`(5)

System Administration Guide: Basic Administration

Name acctprc, acctprc1, acctprc2 – process accounting

Synopsis

```
/usr/lib/acct/acctprc

/usr/lib/acct/acctprc1 [ctmp]

/usr/lib/acct/acctprc2
```

Description `acctprc` reads the standard input and converts it to total accounting records (see the `tacct` record in `acct.h`(3HEAD)). `acctprc` divides CPU time into prime time and non-prime time and determines mean memory size (in memory segment units). `acctprc` then summarizes the `tacct` records, according to user IDs, and adds login names corresponding to the user IDs. The summarized records are then written to the standard output. `acctprc1` reads input in the form described by `acct.h`(3HEAD), adds login names corresponding to user IDs, then writes for each process an ASCII line giving user ID, login name, prime CPU time (tics), non-prime CPU time (tics), and mean memory size (in memory segment units). If *ctmp* is given, it should contain a list of login sessions sorted by user ID and login name. If this file is not supplied, it obtains login names from the password file, just as `acctprc` does. The information in *ctmp* helps it distinguish between different login names that share the same user ID.

From the standard input, `acctprc2` reads records in the form written by `acctprc1`, summarizes them according to user ID and name, then writes the sorted summaries to the standard output as total accounting records.

Examples **EXAMPLE 1** Examples of `acctprc`.

The `acctprc` command is typically used as shown below:

```
example% acctprc  < /var/adm/pacct  > ptacct
```

The `acctprc1` and `acctprc2s` commands are typically used as shown below:

```
example% acctprc1  ctmp  </var/adm/pacct
example% acctprc2 > ptacct
```

Files `/etc/passwd` system password file

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

See Also `acctcom`(1), `acct`(1M), `acctcms`(1M), `acctcon`(1M), `acctmerg`(1M), `acctsh`(1M), `cron`(1M), `fwtmp`(1M), `runacct`(1M), `acct`(2), `acct.h`(3HEAD), `utmpx`(4), `attributes`(5)

Notes Although it is possible for `acctprc1` to distinguish among login names that share user IDs for commands run from a command line, it is difficult for `acctprc1` to make this distinction for commands invoked in other ways. A command run from `cron`(1M) is an example of where

`acctprc1` might have difficulty. A more precise conversion can be done using the `acctwtmp` program in `acct(1M)`. `acctprc` does not distinguish between users with identical user IDs.

A memory segment of the mean memory size is a unit of measure for the number of bytes in a logical memory segment on a particular processor.

During a single invocation of any given command, the `acctprc`, `acctprc1`, and `acctprc2` commands can process a maximum of

- 6000 distinct sessions
- 1000 distinct terminal lines
- 2000 distinct login names

If at some point the actual number of any one of these items exceeds the maximum, the command will not succeed.

Name acctsh, chargefee, ckpacct, dodisk, lastlogin, monacct, nulladm, prctmp, prdaily, prtacct, shutacct, startup, turnacct – shell procedures for accounting

Synopsis

```
/usr/lib/acct/chargefee login-name number

/usr/lib/acct/ckpacct [blocks]

/usr/lib/acct/dodisk [-o] [filename]...

/usr/lib/acct/lastlogin

/usr/lib/acct/monacct number

/usr/lib/acct/nulladm filename...

/usr/lib/acct/prctmp filename

/usr/lib/acct/prdaily [-c] [-l] [mmdd]

/usr/lib/acct/prtacct filename ['' heading '']

/usr/lib/acct/shutacct ['' reason '']

/usr/lib/acct/startup

/usr/lib/acct/turnacct on | off | switch
```

Description

chargefee Command — `chargefee` can be invoked to charge a *number* of units to *login-name*. A record is written to `/var/adm/fee`, to be merged with other accounting records by `runacct`(1M).

ckpacct Command — `ckpacct` should be initiated using `cron`(1M) to periodically check the size of `/var/adm/pacct`. If the size exceeds *blocks*, `500` by default, `turnacct` will be invoked with argument `switch`. To avoid a conflict with `turnacct switch` execution in `runacct`, do not run `ckpacct` and `runacct` simultaneously. If the number of free disk blocks in the `/var` file system falls below `500`, `ckpacct` will automatically turn off the collection of process accounting records via the `off` argument to `turnacct`. When at least `500` blocks are restored, the accounting will be activated again on the next invocation of `ckpacct`. This feature is sensitive to the frequency at which `ckpacct` is executed, usually by the `cron`(1M) command.

dodisk Command — `dodisk` should be invoked by `cron`(1M) to perform the disk accounting functions.

lastlogin Command — `lastlogin` is invoked by `runacct`(1M) to update `/var/adm/acct/sum/loginlog`, which shows the last date on which each person logged in.

monacct Command — `monacct` should be invoked once each month or each accounting period. *number* indicates which month or period it is. If *number* is not given, it defaults to the current month (01–12). This default is useful if `monacct` is to executed using `cron`(1M) on the first day of each month. `monacct` creates summary files in `/var/adm/acct/fiscal` and restarts the summary files in `/var/adm/acct/sum`.

nulladm Command — `nulladm` creates *filename* with mode 664 and ensures that owner and group are `adm`. It is called by various accounting shell procedures.

prctmp Command — `prctmp` can be used to print the session record file (normally `/var/adm/acct/nite/ctmp` created by `acctcon1` (see `acctcon(1M)`).

prdaily Command — `prdaily` is invoked by `runacct(1M)` to format a report of the previous day's accounting data. The report resides in `/var/adm/acct/sum/rprt/mmdd` where *mmdd* is the month and day of the report. The current daily accounting reports may be printed by typing `prdaily`. Previous days' accounting reports can be printed by using the *mmdd* option and specifying the exact report date desired.

prtacct Command — `prtacct` can be used to format and print any total accounting (`tacct`)file.

shutacct Command — `shutacct` is invoked during a system shutdown to turn process accounting off and append a *reason* record to `/var/adm/wtmpx`.

startup Command — `startup` can be invoked when the system is brought to a multi-user state to turn process accounting on.

turnacct Command — `turnacct` is an interface to `accton` (see `acct(1M)`) to turn process accounting `on` or `off`. The `switch` argument moves the current `/var/adm/pacct` to the next free name in `/var/adm/pacct`.*incr* (where *incr* is a number starting with `0` and incrementing by one for each additional `pacct` file), then turns accounting back on again. This procedure is called by `ckpacct` and thus can be taken care of by the `cron(1M)` command and used to keep `pacct` to a reasonable size. `shutacct` uses `turnacct` to stop process accounting. `startup` uses `turnacct` to start process accounting.

Options The following options are supported:

`-c` This option prints a report of exceptional resource usage by command, and may be used on current day's accounting data only.

`-l` This option prints a report of exceptional usage by login id for the specified date. Previous daily reports are cleaned up and therefore inaccessible after each invocation of `monacct`.

`-o` This option uses `acctdusg` (see `acct(1M)`) to do a slower version of disk accounting by login directory. *filename*s specifies the one or more filesystem names where disk accounting will be done. If *filename*s are used, disk accounting will be done on these filesystems only. If the `-o` option is used, *filename*s should be mount points of mounted filesystems. If the `-o` option is omitted, *filename*s should be the special file names of mountable filesystems.

Files

`/etc/logadm.conf`	Configuration file for the `logadm(1M)` command
`/usr/lib/acct`	Holds all accounting commands listed in section 1M of this manual

`/usr/lib/acct/ptecms.awk`	Contains the limits for exceptional usage by command name
`/usr/lib/acct/ptelus.awk`	Contains the limits for exceptional usage by login ID
`/var/adm/acct/fiscal`	Fiscal reports directory
`/var/adm/acct/nite`	Working directory
`/var/adm/acct/sum`	Summary directory that contains information for `monacct`
`/var/adm/acct/sum/loginlog`	File updated by last login
`/var/adm/fee`	Accumulator for fees
`/var/adm/pacct`	Current file for per-process accounting
`/var/adm/pacct`*incr*	Used if `pacct` gets large and during execution of daily accounting procedure
`/var/adm/wtmpx`	History of user access and administration information

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

See Also `acctcom(1)`, `acct(1M)`, `acctcms(1M)`, `acctcon(1M)`, `acctmerg(1M)`, `acctprc(1M)`, `cron(1M)`, `fwtmp(1M)`, `logadm(1M)`, `runacct(1M)`, `acct(2)`, `acct.h(3HEAD)`, `utmpx(4)`, `attributes(5)`

Notes See `runacct(1M)` for the main daily accounting shell script, which performs the accumulation of connect, process, fee, and disk accounting on a daily basis. It also creates summaries of command usage.

Name adbgen – generate adb script

Synopsis `/usr/lib/adb/adbgen [-m` *model*`]` *filename*`.adb . . .`

Description `adbgen` makes it possible to write `adb(1)` scripts that do not contain hard-coded dependencies on structure member offsets. The input to `adbgen` is a file named *filename*`.adb` that contains header information, then a null line, then the name of a structure, and finally an `adb` script. `adbgen` only deals with one structure per file; all member names are assumed to be in this structure. The output of `adbgen` is an `adb` script in *filename*. `adbgen` operates by generating a C program which determines structure member offsets and sizes, which in turn generate the `adb` script.

The header lines, up to the null line, are copied verbatim into the generated C program. Typically, these are `#include` statements, which include the headers containing the relevant structure declarations.

The `adb` script part may contain any valid `adb` commands (see `adb(1)`), and may also contain `adbgen` requests, each enclosed in braces (`{ }`). Request types are:

- Print a structure member. The request form is `{`*member*`,` *format*`}.` *member* is a member name of the *structure* given earlier, and `format` is any valid `adb` format request or any of the `adbgen` format specifiers (such as `{POINTER}`) listed below. For example, to print the `p_pid` field of the *proc* structure as a decimal number, you would write `{p_pid,d}`.
- Print the appropriate `adb` format character for the given `adbgen` format specifier. This action takes the data model into consideration. The request form is `{`*format specifier*`}.` The valid `adbgen` format specifiers are:

`{POINTER}`	pointer value in hexadecimal
`{LONGDEC}`	long value in decimal
`{ULONGDEC}`	unsigned long value in decimal
`{ULONGHEX}`	unsigned long value in hexadecimal
`{LONGOCT}`	long value in octal
`{ULONGOCT}`	unsigned long value in octal

- Reference a structure member. The request form is `{*`*member*`,` *base*`}.` *member* is the member name whose value is desired, and *base* is an `adb` register name which contains the base address of the structure. For example, to get the `p_pid` field of the *proc* structure, you would get the *proc* structure address in an `adb` register, for example `<f`, and write `{*p_pid,<f}`.
- Tell `adbgen` that the offset is valid. The request form is `{OFFSETOK}`. This is useful after invoking another `adb` script which moves the `adb` *dot*.
- Get the size of the *structure*. The request form is `{SIZEOF}`. `adbgen` replaces this request with the size of the structure. This is useful in incrementing a pointer to step through an array of structures.

- Calculate an arbitrary C expression. The request form is {EXPR, *expression*}. adbgen replaces this request with the value of the expression. This is useful when more than one structure is involved in the script.
- Get the offset to the end of the structure. The request form is {END}. This is useful at the end of the structure to get adb to align the *dot* for printing the next structure member.

adbgen keeps track of the movement of the adb *dot* and generates adb code to move forward or backward as necessary before printing any structure member in a script. adbgen's model of the behavior of adb's *dot* is simple: it is assumed that the first line of the script is of the form *struct_address*/*adb text* and that subsequent lines are of the form +/*adb text*. The adb *dot* then moves in a sane fashion. adbgen does not check the script to ensure that these limitations are met. adbgen also checks the size of the structure member against the size of the adb format code and warns if they are not equal.

Options The following option is supported:

-m *model* Specifies the data type model to be used by adbgen for the macro. This affects the outcome of the {*format specifier*} requests described under DESCRIPTION and the offsets and sizes of data types. *model* can be ilp32 or lp64. If the -m option is not given, the data type model defaults to ilp32.

Operands The following operand is supported:

filename.adb Input file that contains header information, followed by a null line, the name of the structure, and finally an adb script.

Examples **EXAMPLE 1** A sample adbgen file.

For an include file x.h which contained

```
struct x {
        char    *x_cp;
        char    x_c;
        int    x_i;
};
```

then , an adbgen file (call it script.adb) to print the file x.h would be:

```
#include "x.h"
x
./"x_cp"16t"x_c"8t"x_i"n{x_cp,{POINTER}}{x_c,C}{x_i,D}
```

After running adbgen as follows,

```
% /usr/lib/adb/adbgen  script.adb
```

the output file script contains:

```
./"x_cp"16t"x_c"8t"x_i"nXC3+D
```

EXAMPLE 1 A sample adbgen file. *(Continued)*

For a macro generated for a 64-bit program using the `lp64` data model as follows,

```
% /usr/lib/adb/adbgen/ -m lp64  script.adb
```

the output file `script` would contain:

```
./"x_cp"16t"x_c"8t"x_i"nJC3+D
```

To invoke the script, type:

```
example% adb program
x$<script
```

Files

`/usr/platform/`*platform-name*`/lib/adb/*`
platform-specific `adb` scripts for debugging the 32-bit kernel

`/usr/platform/`*platform-name*`/lib/adb/sparcv9/*`
platform-specific `adb` scripts for debugging the 64-bit SPARC V9 kernel

`/usr/lib/adb/*`
`adb` scripts for debugging the 32-bit kernel

`/usr/lib/adb/sparcv9/*`
`adb` scripts for debugging the 64-bit SPARC V9 kernel

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu

See Also

`adb(1)`, `uname(1)`, `kadb(1M)`, `attributes(5)`

Diagnostics

Warnings are given about structure member sizes not equal to `adb` format items and about badly formatted requests. The C compiler complains if a structure member that does not exist is referenced. It also complains about an ampersand before array names; these complaints may be ignored.

Notes

platform-name can be found using the `-i` option of `uname(1)`.

Bugs

`adb` syntax is ugly; there should be a higher level interface for generating scripts.

Structure members which are bit fields cannot be handled because C will not give the address of a bit field. The address is needed to determine the offset.

Name add_allocatable – add entries to allocation databases

Synopsis

```
/usr/sbin/add_allocatable [-f] [-s] [-d] -n name -t type -l device-list
    [-a authorization] [-c clean] [-o key=value]
```

Description

`add_allocatable` creates new entries for user allocatable devices that are to be managed by the device allocation mechanism. `add_allocatable` can also be used to update existing entries of such devices.

`add_allocatable` can also create and update entries for non-allocatable devices, such as printers, whose label range is managed by the device allocation mechanism.

`add_allocatable` can be used in shell scripts, such as installation scripts for driver packages, to automate the administrative work of setting up a new device.

Use `list_devices(1)` to see the names and types of allocatable devices, their attributes, and device paths.

Options

`-f`	Force an update of an already-existing entry with the specified information. `add_allocatable` exits with an error if this option is not specified when an entry with the specified device name already exists.
`-s`	Turn on silent mode. `add_allocatable` does not print any error or warning messages.
`-d`	If this option is present, `add_allocatable` updates the system-supplied default attributes of the device type specified with `-t`.
`-n` *name*	Adds or updates an entry for device that is specified by *name*.
`-t` *type*	Adds or updates device entries that are of a type that are specified by *type*.
`-l` *device-list*	Adds or updates device paths to the device that is specified with `-n`. Multiple paths in *device-list* must be separated by white spaces and the list must be quoted.
`-a` *authorization*	Adds or updates the authorization that is associated with either the device that is specified with `-n` or with devices of the type that is specified with `-t`. When more than one authorization is specified, the list must be separated by commas and must be quoted. When the device is not allocatable, *authorization* is specified with an asterisk (*) and must be quoted. When the device is allocatable by any user, *authorization* is specified with the at sign (@) and must be quoted. Default authorization is '@'.
`-c` *clean*	Specifies the `device_clean(5)` program *clean* to be used with the device that is specified with `-n` or with devices of the type that is specified with `-t`. The default clean program is `/bin/true`.

`-o` *key=value* Accepts a string of colon-separated *key=value* pairs for a device that is specified with `-n` or with devices of the type that is specified with `-t`. The following keys are currently interpreted by the system:

`minlabel` The minimum label at which the device can be used.

`maxlabel` The maximum label at which the device can be used.

`class` Specifies a logical grouping of devices. For example, all Sun Ray devices of all device types is a logical grouping. The `class` keyword has no default value.

`xdpy` Specifies the display name of the X session. This keyword is used to identify devices that are associated with the X session. The `xdpy` keyword has no default value.

Exit Status When successful, `add_allocate` returns an exit status of `0` (true). `add_allocate` returns a nonzero exit status in the event of an error. The exit codes are as follows:

`1` Invocation syntax error

`2` Unknown system error

`3` An entry already exists for the specified device. This error occurs only when the `-f` option is not specified.

`4` Permission denied. User does not have DAC or MAC access record updates.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	See below.

The invocation is Uncommitted. The options are Uncommitted. The output is Not-an-Interface.

See Also `allocate(1)`, `deallocate(1)`, `list_devices(1)`, `remove_allocatable(1M)`, `attributes(5)`, `device_clean(5)`

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

Name addbadsec – map out defective disk blocks

Synopsis `addbadsec [-p] [-a` *blkno* `[`*blkno*`]...] [-f` *filename*`]` *raw_device*

Description `addbadsec` is used by the system administrator to map out bad disk blocks. Normally, these blocks are identified during surface analysis, but occasionally the disk subsystem reports unrecoverable data errors indicating a bad block. A block number reported in this way can be fed directly into `addbadsec`, and the block will be remapped. `addbadsec` will first attempt hardware remapping. This is supported on SCSI drives and takes place at the disk hardware level. If the target is an IDE drive, then software remapping is used. In order for software remapping to succeed, the partition must contain an alternate slice and there must be room in this slice to perform the mapping.

It should be understood that bad blocks lead to data loss. Remapping a defective block does not repair a damaged file. If a bad block occurs to a disk-resident file system structure such as a superblock, the entire slice might have to be recovered from a backup.

Options The following options are supported:

`-a` Adds the specified blocks to the hardware or software map. If more than one block number is specified, the entire list should be quoted and block numbers should be separated by white space.

`-f` Adds the specified blocks to the hardware or software map. The bad blocks are listed, one per line, in the specified file.

`-p` Causes `addbadsec` to print the current software map. The output shows the defective block and the assigned alternate. This option cannot be used to print the hardware map.

Operands The following operand is supported:

raw_device The address of the disk drive (see `FILES`).

Files The raw device should be `/dev/rdsk/c?[t?]d?p0`. See `disks(1M)` for an explanation of SCSI and IDE device naming conventions.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWcsu

See Also `disks(1M)`, `diskscan(1M)`, `fdisk(1M)`, `fmthard(1M)`, `format(1M)`, `attributes(5)`

Notes The `format(1M)` utility is available to format, label, analyze, and repair SCSI disks. This utility is included with the `addbadsec`, `diskscan(1M)`, `fdisk(1M)`, and `fmthard(1M)` commands available for x86. To format an IDE disk, use the DOS "format" utility; however, to label, analyze, or repair IDE disks on x86 systems, use the Solaris `format(1M)` utility.

Name add_drv – add a new device driver to the system

Synopsis

```
add_drv  [-b basedir] [-c class_name] [-i 'identify_name...']
 [-m 'permission','...'] [-p 'policy'] [-P privilege]
 [-n] [-f] [-v] device_driver
```

Description The `add_drv` command is used to inform the system about newly installed device drivers.

Each device on the system has a name associated with it. This name is represented by the `name` property for the device. Similarly, the device may also have a list of driver names associated with it. This list is represented by the `compatible` property for the device.

The system determines which devices will be managed by the driver being added by examining the contents of the `name` property and the `compatible` property (if it exists) on each device. If the value in the `name` property does not match the driver being added, each entry in the `compatible` property is tried, in order, until either a match occurs or there are no more entries in the `compatible` property.

In some cases, adding a new driver may require a reconfiguration boot. See the `NOTES` section.

Aliases might require quoting (with double-quotes) if they contain numbers. See `EXAMPLES`.

The `/etc/minor_perm` File

`add_drv` and `update_drv(1M)` read the `/etc/minor_perm` file to obtain permission information. The permission specified is applied to matching minor nodes created when a device bound to the driver is attached. A minor node's permission may be manually changed by `chmod(1)`. For such nodes, the specified permissions apply, overriding the default permissions specified via `add_drv` or `update_drv(1M)`.

The format of the `/etc/minor_perm` file is as follows:

name:*minor_name permissions owner group*

minor_name may be the actual name of the minor node, or contain shell metacharacters to represent several minor nodes (see `sh(1)`).

For example:

```
sd:* 0640 root sys
zs:[a-z],cu 0600 uucp uucp
mm:kmem 0640 root bin
```

The first line sets all devices exported by the `sd` node to `0640` permissions, owned by `root`, with group `sys`. In the second line, devices such as `a,cu` and `z,cu` exported by the `zs` driver are set to `0600` permission, owned by `uucp`, with group `uucp`. In the third line the `kmem` device exported by the `mm` driver is set to `0640` permission, owned by `root`, with group `bin`.

Running `add_drv` from a `postinstall` Script

When running `add_drv` from within the context of a package's postinstall script, you must consider whether the package is being added to a system image or to a running system. When a package is being installed on a system image, such as occurs with the Live Upgrade or flash

features (see live_upgrade(5) and flarcreate(1M)), the BASEDIR variable refers to the image's base directory. In this situation, add_drv should be invoked with -b $BASEDIR. This causes add_drv only to update the image's system files; a reboot of the system or client would be required to make the driver operational.

When a package is being installed on the running system itself, the system files need to be updated, as in the case above. However, the running kernel can be informed of the existence of the new driver without requiring a reboot. To accomplish this, the postinstall script must invoke add_drv without the -b option. Accordingly, postinstall scripts invoking add_drv should be written thusly:

```
if [ "${BASEDIR:=/}" = "/" ]
then
        ADD_DRV="add_drv"
else
        ADD_DRV="add_drv -b ${BASEDIR}"
fi
$ADD_DRV [<options>] <driver>
```

...or, alternatively:

```
if [ "${BASEDIR:=/}" != "/" ]
then
         BASEDIR_OPT="-b $BASEDIR"
fi
         add_drv $BASEDIR_OPT [<options>] <driver>
```

The -b option is described below.

Options -b *basedir*

Installs the driver on the system with a root directory of *basedir* rather than installing on the system executing add_drv. This option is typically used in package post-installation scripts when the package is not being installed on the system executing the pkgadd command. The system using *basedir* as its root directory must reboot to complete the driver installation.

Note – The root file system of any non-global zones must not be referenced with the -b option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See zones(5).

-c *class_name*

The driver being added to the system exports the class *class_name*.

-f

Normally if a reconfiguration boot is required to complete the configuration of the driver into the system, add_drv will not add the driver. The force flag forces add_drv to add the driver even if a reconfiguration boot is required. See the -v flag.

`-i` *'identify_name'*
A white-space separated list of aliases for the driver *device_driver*.

`-m` *'permission'*
Specify the file system permissions for device nodes created by the system on behalf of *device_driver*.

`-n`
Do not try to load and attach *device_driver*, just modify the system configuration files for the *device_driver*.

`-p` *'policy'*
Specify an additional device security policy.

The device security policy constists of several whitespace separated tokens:

{*minorspec* `{token=value}+}+`

minorspec is a simple wildcard pattern for a minor device. A single * matches all minor devices. Only one * is allowed in the pattern.

Patterns are matched in the following order:

- entries without a wildcard
- entries with wildcards, longest wildcard first

The following tokens are defined: `read_priv_set` and `write_priv_set`. `read_priv_set` defines the privileges that need to be asserted in the effective set of the calling process when opening a device for reading. `write_priv_set` defines the privileges that need to be asserted in the effective set of the calling process when opening a device for writing. See `privileges(5)`.

A missing minor spec is interpreted as a *.

`-P` *'privilege'*
Specify additional, comma-separated privileges used by the driver. You can also use specific privileges in the device's policy.

`-v`
The verbose flag causes `add_drv` to provide additional information regarding the success or failure of a driver's configuration into the system. See the `EXAMPLES` section.

Examples

EXAMPLE 1 Adding SUNW Example Driver to the System

The following example adds the `SUNW,example` driver to a 32–bit system, with an alias name of `SUNW,alias`. It assumes the driver has already been copied to `/usr/kernel/drv`.

```
example# add_drv -m '* 0666 bin bin','a 0644 root sys' \
      -p 'a write_priv_set=sys_config  * write_priv_set=none' \
      -i 'SUNW,alias' SUNW,example
```

EXAMPLE 1 Adding SUNW Example Driver to the System *(Continued)*

Every minor node created by the system for the `SUNW,example` driver will have the permission `0666`, and be owned by user `bin` in the group `bin`, except for the minor device `a`, which will be owned by `root`, group `sys`, and have a permission of `0644`. The specified device policy requires no additional privileges to open all minor nodes, except minor device `a`, which requires the `sys_config` privilege when opening the device for writing.

EXAMPLE 2 Adding Driver to the Client `/export/root/sun1`

The following example adds the driver to the client `/export/root/sun1`. The driver is installed and loaded when the client machine, `sun1`, is rebooted. This second example produces the same result as the first, except the changes are on the diskless client, `sun1`, and the client must be rebooted for the driver to be installed.

```
example# add_drv -m '* 0666 bin bin','a 0644 root sys' \
        -i 'SUNW,alias' -b /export/root/sun1 \
    SUNW,example
```

See the note in the description of the `-b` option, above, specifying the caveat regarding the use of this option with the Solaris zones feature.

EXAMPLE 3 Adding Driver for a Device Already Managed by an Existing Driver

The following example illustrates the case where a new driver is added for a device that is already managed by an existing driver. Consider a device that is currently managed by the driver `dumb_framebuffer`. The `name` and `compatible` properties for this device are as follows:

```
name="display"
compatible="whizzy_framebuffer", "dumb_framebuffer"
```

If `add_drv` is used to add the `whizzy_framebuffer` driver, the following will result.

```
example# add_drv whizzy_framebuffer
Error: Could not install driver (whizzy_framebuffer)
Device managed by another driver.
```

If the `-v` flag is specified, the following will result.

```
example# add_drv -v whizzy_framebuffer
Error: Could not install driver (whizzy_framebuffer)
Device managed by another driver.
Driver installation failed because the following
entries in /devices would be affected:

        /devices/iommu@f,e0000000/sbus@f,e0001000/display[:*]
        (Device currently managed by driver "dumb_framebuffer")

The following entries in /dev would be affected:

        /dev/fbs/dumb_framebuffer0
```

EXAMPLE 3 Adding Driver for a Device Already Managed by an Existing Driver *(Continued)*

If the `-v` and `-f` flags are specified, the driver will be added resulting in the following.

```
example# add_drv -vf whizzy_framebuffer
A reconfiguration boot must be performed to complete the
installation of this driver.

The following entries in /devices will be affected:

        /devices/iommu@f,e0000000/sbus@f,e0001000/display[:*]
        (Device currently managed by driver "dumb_framebuffer"

The following entries in /dev will be affected:

        /dev/fbs/dumb_framebuffer0
```

The above example is currently only relevant to devices exporting a generic device name.

EXAMPLE 4 Use of Double Quotes in Specifying Driver Alias

The following example shows the use of double quotes in specifying a driver alias that contains numbers.

```
example# add_drv -i '"pci10c5,25"' smc
```

Exit Status `add_drv` returns `0` on success and `1` on failure.

Files

`/kernel/drv`
32–bit boot device drivers

`/kernel/drv/sparcv9`
64–bit SPARC boot device drivers

`/kernel/drv/amd64`
64–bit x86 boot device drivers

`/usr/kernel/drv`
other 32–bit drivers that could potentially be shared between platforms

`/usr/kernel/drv/sparcv9`
other 64–bit SPARC drivers that could potentially be shared between platforms

`/usr/kernel/drv/amd64`
other 64–bit x86 drivers that could potentially be shared between platforms

`/platform/‘uname -i‘/kernel/drv`
32–bit platform-dependent drivers

`/platform/‘uname -i‘/kernel/drv/sparcv9`
64–bit SPARC platform-dependent drivers

`/platform/‘uname -i‘/kernel/drv/amd64`
64–bit x86 platform-dependent drivers

`/etc/driver_aliases`
driver aliases file

`/etc/driver_classes`
driver classes file

`/etc/minor_perm`
minor node permissions

`/etc/name_to_major`
major number binding

`/etc/security/device_policy`
device policy

`/etc/security/extra_privs`
device privileges

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `boot(1M)`, `chmod(1)`, `devfsadm(1M)`, `flarcreate(1M)`, `kernel(1M)`, `modinfo(1M)`, `rem_drv(1M)`, `update_drv(1M)`, `driver.conf(4)`, `system(4)`, `attributes(5)`, `live_upgrade(5)`, `privileges(5)`, `devfs(7FS)`, `ddi_create_minor_node(9F)`

Writing Device Drivers

Notes It is possible to add a driver for a device already being managed by a different driver, where the driver being added appears in the device's `compatible` list before the current driver. In such cases, a reconfiguration boot is required (see `boot(1M)` and `kernel(1M)`). After the reconfiguration boot, device links in `/dev` and references to these files may no longer be valid (see the `-v` flag). If a reconfiguration boot would be required to complete the driver installation, `add_drv` will fail unless the `-f` option is specified. See `Example 3` in the `EXAMPLES` section.

With the introduction of the device policy several drivers have had their minor permissions changed and a device policy instated. The typical network driver should use the following device policy:

```
add_drv -p 'read_priv_set=net_rawaccess\
   write_priv_set=net_rawaccess' -m '* 666 root sys'\
   mynet
```

This document does not constitute an API. `/etc/minor_perm`, `/etc/name_to_major`, `/etc/driver_classes`, and `/devices` may not exist or may have different contents or interpretations in a future release. The existence of this notice does not imply that any other documentation that lacks this notice constitutes an API.

`/etc/minor_perm` can only be updated by `add_drv`, `rem_drv(1M)` or `update_drv(1M)`.

Bugs Previous versions of `add_drv` accepted a pathname for *device_driver*. This feature is no longer supported and results in failure.

Name afbconfig, SUNWafb_config – configure the AFB Graphics Accelerator

Synopsis

```
/usr/sbin/afbconfig [-dev device-filename]
     [-res video-mode [now | try] [noconfirm | nocheck]]
     [-file machine | system] [-deflinear true | false]
     [-defoverlay true | false]
     [-overlayorder first | last]
     [-expvis enable | disable] [-sov enable | disable]
     [-maxwinds n] [-extovl enable | disable]
     [-g gamma-correction-value]
     [-gfile gamma-correction-file] [-propt] [-prconf]
     [-defaults]

/usr/sbin/afbconfig [-propt] [-prconf]

/usr/sbin/afbconfig [-help] [-res ?]
```

Description `afbconfig` configures the AFB Graphics Accelerator and some of the `X11` window system defaults for AFB.

The following form of `afbconfig` stores the specified options in the `OWconfig` file:

```
 /usr/sbin/afbconfig [-devdevice-filename]
     [-res video-mode [now | try] [noconfirm | nocheck]]
     [-file machine | system] [-deflinear true | false]
     [-defoverlay true | false]
     [-overlayorderfirst | last] [-expvisenable | disable]
     [-sov enable | disable] [-maxwindsn]
     [-extovl enable | disable] [-ggamma-correction-value]
     [-gfilegamma-correction-file] [-propt] [-prconf]
     [-defaults]
```

The options are used to initialize the AFB device the next time the window system is run on that device. Updating options in the `OWconfig` file provides persistence of these options across window system sessions and system reboots.

The following forms of the `afbconfig` command invoke only the `-prconf`, `-propt`, `-help`, and `-res ?` options. None of these options update the `OWconfig` file.

```
/usr/sbin/afbconfig [-propt] [-prconf]

/usr/sbin/afbconfig [-help] [-res ?]
```

Additionally, the following invokation of `afbconfig` ignores all other options:

```
/usr/sbin/afbconfig [-help] [-res ?]
```

You can only specify options for one AFB device at a time. Specifying options for multiple AFB devices requires multiple invocations of the `afbconfig` command.

Only AFB-specific options can be specified through `afbconfig`. The normal window system options for specifying default depth, visual class and so forth are still specified as device modifiers on the `openwin` command line.

You can also specify the `OWconfig` file that is to be updated. By default, the machine-specific file in the `/etc/openwin` directory tree is updated. The `-file` option can be used to specify an alternate file to use. For example, the system-global OWconfig file in the `/usr/openwin` directory tree can be updated instead.

Both of these standard `OWconfig` files can only be written by root. Consequently, the `afbconfig` program, which is owned by the root user, always runs with `setuid root` permission.

Option Defaults

For a given invocation of `afbconfig` command line if an option does not appear on the command line, the corresponding `OWconfig` option is not updated; it retains its previous value. When the window system is run, if an AFB option has never been specified by way of `afbconfig`, a default value is used. The option defaults are as follows:

`-dev`	/dev/fbs/afb0
`-file`	machine
`-res`	none
`-deflinear`	false
`-defoverlay`	false
`-linearorder`	last
`-overlayorder`	last
`-expvis`	enabled
`-sov`	enabled
`-maxwids`	32
`-extovl`	enabled
`-g`	2.22

The default for the `-res` option of `none` means that when the window system is run the screen resolution is the video mode currently programmed in the device.

This provides compatibility for users who are used to specifying the device resolution through the PROM. On some devices (for example, `GX`) this is the only way of specifying the video mode. This means that the PROM ultimately determines the default AFB video mode.

Options

The following options are supported:

`-defaults`
Resets all option values to their default values.

`-deflinear` true | false
AFB possesses two types of visuals: linear and nonlinear. Linear visuals are gamma corrected and nonlinear visuals are not. There are two visuals that have both linear and nonlinear versions: 24-bit TrueColor and 8-bit StaticGray.

If `true`, the default visual is set to the linear visual that satisfies other specified default visual selection options (specifically, the Xsun(1) defdepth and defclass options described in the OpenWindows Reference Manual).

If false, or if there is no linear visual that satisfies the other default visual selection options, the non-linear visual specified by these other options are chosen as the default. This option cannot be used when the `-defoverlay` option is present, because AFB doesn't possess a linear overlay visual.

`-defoverlay` true | false
The AFB provides an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the AFB visuals. This is called the overlay visual. Windows created in this visual do not damage windows created in other visuals. The converse, however, is not true. Windows created in other visuals damage overlay windows.

The number of colors available to the windows created using this visual depends on the settings for the `-extovl` option. If the `-extovl` is enabled, extended overlay with 256 opaque color values is available. See `-extovl`. If `-extovl` is disabled, extended overlay is not available and the visual has `256 -maxwids`) number of opaque color values. See `-maxwids`.

If the value of `-defoverlay` is `true`, the overlay visual is made the default visual. If the value of `-defoverlay` is `false`, the nonoverlay visual that satisfies the other default visual selection options, such as `def`, `depth`, and `defclass`, are chosen as the default visual. See the OpenWindows Reference Manual.

Whenever the `defoverlay true` option is used, the default depth and class specified on the `openwin` command line must be 8-bit PseudoColor. If not, a warning message is printed and the `-defoverlay` option is treated as `false`.

The `-defoverlay` option can not be used when the `-deflinear` option specified, because AFB doesn't possess a linear overlay visual.

`-dev` *device-filename*
Specifies the AFB special file. The default is `/dev/fbs/afb0`.

`-expvis` enable | disable
If enabled, activates OpenGL Visual Expansion. Multiple instances of selected visual groups (8-bit PseudoColor, 24-bit TrueColor and so forth) are in the screen visual list.

`-extovl` enable | disable
If enabled, makes extended overlay available. The overlay visuals have 256 opaque colors. The SOV visuals have 255 opaque colors and 1 transparent color.

This option also enables hardware supported transparency, thus provides better performance for windows using the SOV visuals.

`-file machine| system`
Specifies which `OWconfig` file to update. If *machine* is specified, the machine-specific `OWconfig` file in the `/etc/openwin` directory tree is updated. If *system* is specified, the global `OWconfig` file in the `/usr/openwin` directory tree is updated. If the specified file does not exist, it is created. This option has no effect unless other options are specified. The default is *machine*.

`-g` *gamma-correction value*
Allows changing the gamma correction value. All linear visuals provide gamma correction. By default, the *gamma-correction-value* is `2.22`. Any value less than `0` is illegal. The gamma correction value is applied to the linear visual, which then has an effective gamma value of 1.0, which is the value returned by `XSolarisGetVisualGamma`(3). See `XSolarisGetVisualGamma`(3) for a description of that function.

This option can be used while the window system is running. Changing the gamma correction value affects all the windows being displayed using the linear visuals.

`-gfile` *gamma-correction-file*
Loads the gamma correction table from the specified file (*gamma-correction-file*). This file should be formatted to provide the gamma correction values for `R`, `G` and `B` channels on each line. Each of these values should be in hexadecimal format and separated from each other by at least one space. *gamma-correction-file* should also provide `256` such triplets.

An example of a *gamma-correction-file* follows.

```
0x00 0x00 0x00
0x01 0x01 0x01
0x02 0x02 0x02
...
...
0xff 0xff 0xff
```

Using this option, the gamma correction table can be loaded while the window system is running. The new gamma correction affects all the windows being displayed using the linear visuals. When gamma correction is being done using user specified table, the gamma correction value is undefined. By default, the window system assumes a gamma correction value of `2.22` and loads the gamma table it creates corresponding to this value.

`-help`
Prints a list of the `afbconfig` command line options, along with a brief explanation of each.

`-linearorder` first | last
If `first`, linear visuals come before their non-linear counterparts on the X11 screen visual list for the AFB screen. If `last`, the nonlinear visuals come before the linear ones.

`-maxwids` *n*
Specifies the maximum number of `AFB X` channel pixel values that are reserved for use as window IDs (WIDs). The remainder of the pixel values in overlay colormaps are used for normal `X11` opaque color pixels. The reserved WIDs are allocated on a first-come first-serve basis by 3D graphics windows (such as `XGL`), MBX windows, and windows that have a non-default visual. The `X` channel codes `0` to (`255` - *n*) are opaque color pixels. The `X` channel codes (`255` - *n* + `1`) to `255` are reserved for use as WIDs. Legal values are `1`, `2`, `4`, `8`, `16`, `32`, and `64`.

This option is available only if the `-extovl` is disabled.

`-overlayorder` first | last
If `first`, the depth 8 PseudoColor Overlay visual comes before the non-overlay visual on the X11 screen visual list for the AFB screen. If `last`, the non-overlay visual comes before the overlay one.

`-propt`
Prints the current values of all `AFB` options in the `OWconfig` file specified by the `-file` option for the device specified by the `-dev` option. Prints the values of options as they will be in the `OWconfig` file after the call to `afbconfig` completes.

The following is a typical display:

```
--- OpenWindows Configuration for /dev/fbs/afb0 ---
OWconfig: machine
Video Mode: 1280x1024x76
Default Visual: Non-Linear Normal Visual
Visual Ordering: Linear Visuals are last
                 Overlay Visuals are last
OpenGL Visual Expansion: enabled
Server Overlay Visuals: enabled
Extended Overlay: enabled
Underlay WIDs: 64 (not configurable)
Overlay WIDs: 4 (not configurable)
Gamma Correction Value: 2.220
Gamma Correction Table: Available
```

`-prconf`
Prints the AFB hardware configuration.

The following is a typical display:

```
--- Hardware Configuration for /dev/fbs/afb0 ---
Type: double-buffered AFB with Z-buffer
Board: rev 0 (Horizontal)
Number of Floats: 6
PROM Information: @(#)afb.fth x.xx xx/xx/xx
AFB ID: 0x101df06d
DAC: Brooktree 9070, version 1 (Pac2)
```

```
3DRAM: Mitsubishi 130a, version x
EDID Data: Available - EDID version 1 revision x
Monitor Sense ID: 4  (Sun 37x29cm RGB color monitor)
Monitor possible resolutions: 1024x768x77, 1024x800x84, 1
        1152x900x76, 1280x1024x67, 1280x1024x76, 960x680xx108s
Current resolution setting: 1280x1024x76
```

`-sov` enable | disable
If enabled, the root window's SERVER_OVERLAY_VISUALS property are advertised. SOV visuals are exported and their transparent types, values and layers can be retrieved through this property. If disabled, the SERVER_OVERLAY_VISUALS property are not defined and SOV visuals are not exported.

`-res` *video-mode* [now | try [noconfirm | nocheck]]
Specifies the video mode used to drive the monitor connected to the specified AFB device.

The format of these built-in video modes is: *width*x*height*x*rate*, where *width* is the screen width in pixels, *height* is the screen height in pixels, and *rate* is the vertical frequency of the screen refresh.

The `s` suffix of `960x680x112s` and `960x680x108s` means that these are stereo video modes. The `i` suffix of `640x480x60i` and `768x575x50i` designates interlaced video timing. If absent, non-interlaced timing is used.

As a convenience, the `-res` also accepts formats with an at sign (`@`) in front of the refresh rate instead of *n*, (`1280x1024@76`). Some video-modes, supported by AFB, may not be supported by the monitor. The list of video-modes supported by the AFB device and the monitor can be obtained by running `afbconfig` with the `-res ?` option (the third form shown SYNOPSIS).

A list of all possible video-modes supported on AFB follows:

```
1024x768x60
1024x768x70
1024x768x75
1024x768x77
1024x800x84
1152x900x66
1152x900x76
1280x800x76
1280x1024x60
1280x1024x67
1280x1024x76
960x680x112s    (Stereo)
960x680x108s    (Stereo)
640x480x60
640x480x60i     (Interlaced)
768x575x50i     (Interlaced)
```

For convenience, some of the video-modes supported on the AFB have *symbolic names* defined for them. Instead of the form *width*x*height*x*rate*, one of these names may be supplied as the argument to the `-res` option. The meaning of the symbolic name `none` is that when the window system is run, the screen resolution is the video mode that is currently programmed in the device.

A list of symbolic names for video-modes supported on AFB follows:

```
Name    Corresponding Video Mode
svga     1024x768x60
1152     1152x900x76
1280     1280x1024x76
stereo   960x680x112s
ntsc     640x480x60i
pal      768x575x50i
none     (see text above)
```

The `-res` option also accepts the additional, optional arguments immediately following the video mode specification. Any or all of the following arguments can be specified:

noconfirm
: Using the `-res` option, the user could potentially put the system into an unusable state, a state where there is no video output. This can happen if there is ambiguity in the monitor sense codes for the particular code read. To reduce the chance of this, the default behavior of `afbconfig` is to print a warning message to this effect and to prompt the user to find out if it is okay to continue. The `noconfirm` option instructs `afbconfig` to bypass this confirmation and to program the requested video mode anyway. This option is useful when `afbconfig` is being run from a shell script.

nocheck
: If present, the normal error checking based on the monitor sense code is suspended. The video mode specified by the user is accepted regardless of whether it is appropriate for the currently attached monitor. (This option is useful if a different monitor is to be connected to the AFB device). Use of this option implies `noconfirm` well.

now
: Updates the video mode in the `OWconfig` file, and immediately programs the AFB device to display this video mode. This is useful for changing the video mode before starting the window system.

  This argument should not be used with `afbconfig` while the configured device is being used (for example, while running the window system); unpredictable results may occur. To run `afbconfig` with the now argument, first bring the window system down. If the now argument is used within a window system session, the video mode is changed immediately, but the width and height of the affected screen won't change until the window system is exited and re-entered again. In addition, the system may not recognize changes in stereo mode. Consequently, this usage is strongly discouraged.

try — If present, the specified video mode is programmed on a trial basis. The user is asked to confirm the video mode by typing `y` within 10 seconds. Or the user may terminate the trial before 10 seconds are up by typing any character. Any character other than `y` or Return is considered a `no`. The previous video mode is restored and `afbconfig` does not change the video mode in the OWconfig file (other options specified still take effect). If a Return is typed, the user is prompted for a `yes` or `no` answer on whether to keep the new video mode.

This sub-option should not be used with `afbconfig` while the configured device is being used (for example, while running the window system) as unpredictable results may occur. To run `afbconfig` with the `try` sub-option, the window system should be brought down first.

Examples

EXAMPLE 1 Switching the monitor type

The following example switches the monitor type to a resolution of `1280` x `1024` `at 76 Hz`:

```
example% /usr/sbin/afbconfig -res 1280x1024x76
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWafbcf

See Also

`mmap(2)`, `attributes(5)`

Name aliasadm – manipulate the NIS+ aliases map

Synopsis

```
aliasadm -a alias expansion [options comments] optional flags

aliasadm -c alias expansion [options comments]
     [optional flags]

aliasadm -d alias [optional flags]

aliasadm -e alias [optional flags]

aliasadm -l alias [optional flags]

aliasadm -m alias [optional flags]

aliasadm [-I] [-D domainname] [-f filename] [-M mapname]
```

Description `aliasadm` makes changes to the alias map.

The alias map is an NIS+ table object with four columns:

`alias`	The name of the alias as a null terminated string.
expansion	The value of the alias as it would appear in a `sendmail` `/etc/aliases` file.
options	A list of options applicable to this alias. The only option currently supported is `CANON`. With this option, if the user has requested an inverse alias lookup, and there is more than one alias with this expansion, this alias is given preference.
comments	An arbitrary string containing comments about this alias. The `sendmail(1M)` command reads this map in addition to the NIS aliases map and the local `/etc/aliases` database.

Options

`-a`	Add an alias.
`-c`	Change an alias.
`-d`	Delete an alias.
`-e`	Edit the alias map.
`-I`	Initialize the NIS+ aliases database.
`-l`	List the alias map.
`-m`	Print or match an alias.
`-D` *domainname*	Edit the map in domain `domainname` instead of the current domain.
`-f` *filename*	When editing or listing the database, use *filename* instead of invoking the editor.
`-M` *mapname*	Edit *mapname* instead of *mail_aliases*.

Files `/etc/aliases` mail aliases for the local host in ASCII format

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `sendmail(1M)`, `attributes(5)`

Notes NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name answerbook2_admin – bring up AnswerBook2 administration tool GUI

Synopsis

```
/usr/dt/bin/answerbook2_admin [-h]
```

Description The AnswerBook2 server product is no longer included with Solaris or the Solaris Documentation CD products. Solaris docmentation is now provided in HTML and PDF format on the Documentation CD and does not require the Answerbook2 server to be viewed.

`answerbook2_admin` brings up the default web browser showing the administration interface for the local AnswerBook2 server.

The administration functionality is also accessible through the AnswerBook2 `Admin` option within the `System_Admin` subset of the Application Manager function on the CDE front panel Applications menu.

If you need an AnswerBook2 server, you can download the AnswerBook2 server software from http://www.sun.com.

Options The following option is supported:

`-h` Displays a usage statement.

Usage At startup time, `answerbook2_admin` starts up the default web browser (for example, Mozilla) and displays the URL specified for administering the local AnswerBook2 server (`http://localhost:8888`). If the user has set up administration access control, the web browser prompts for a valid administrator login and password for this document server before displaying the administration tool.

Files `/usr/lib/ab2/dweb/data/config/admin_passwd` File containing *username*: *password*

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	http://www.sun.com

See Also `attributes(5)`

Notes Once there is an open web browser and access to the AnswerBook2 Administration tool, use its online Help system to find out more about administering the AnswerBook2 server.

Name apache – Apache hypertext transfer protocol server overview

Description apache consists of a main server daemon, loadable server modules, some additional support utilities, configuration files, and documentation.

Files The apache HTTPD server is integrated with Solaris.

The following files specify the installation locations for apache:

/etc/apache	Contains server configuration files. A newly-installed server must be manually configured before use. Typically this involves copying httpd.conf-example to the httpd.conf file and making local configuration adjustments.
/usr/apache/bin	Contains the httpd executable as well as other utility programs.
/usr/apache/htdocs	Contains the Apache manual in HTML format. This documentation is accessible by way of a link on the server test page that gets installed upon fresh installation.
/usr/apache/include	Contains the Apache header files, which are needed for building various optional server extensions with apxs(8)
/usr/apache/jserv	Contains documention for the mod_jserv java servlet module. Documention can be read with a web browser using the url: file:/usr/apache/jserv/docs/index.html
/usr/apache/libexec	Contains loadable modules (DSOs) supplied with the server. Any modules which are added using apxs(8)are also copied into this directory.
/usr/apache/man	Contains man pages for the server, utility programs, and mod_perl. Add this directory to your MANPATH to read the Apache man pages. See NOTES.
/usr/apache/perl5	Contains the modules and library files used by the mod_perl extension to Apache.
/var/apache/cgi-bin	Default location for the CGI scripts. This can be changed by altering the httpd.conf file and restarting the server.
/var/apache/htdocs	Default document root. This can be changed by altering the httpd.conf file and restarting the server.

`/var/apache/icons`	Icons used by the server. This normally shouldn't need to be changed.
`/var/apache/logs`	Contains server log files. The formats, names, and locations of the files in this directory can be altered by various configuration directives in the `httpd.conf` file.
`/var/apache/proxy`	Directory used to cache pages if the caching feature of `mod_proxy` is enabled in the `httpd.conf` file. The location of the cache can also be changed by changing the proxy configuration in the `httpd.conf` file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Obsolete
Availability	SUNWapchr SUNWapchu SUNWapchd

See Also `attributes(5)`

http://www.apache.org

Notes In addition to the documentation and man pages included with Solaris, more information is available at http://www.apache.org

The Apache man pages are provided with the programming modules. To view the manual pages for the Apache modules with the man command, add `/usr/apache/man` to the `MANPATH` environment variable. See `man(1)` for more information. Running `catman(1M)` on the Apache manual pages is not supported.

Name arp – address resolution display and control

Synopsis

```
arp hostname

arp -a [-n]

arp -d hostname

arp -f filename

arp -s hostname ether_address [temp] [pub] [trail]
     [permanent]
```

Description The `arp` program displays and modifies the Internet-to-MAC address translation tables used by the address resolution protocol (see `arp(7P)`).

With no flags, the program displays the current ARP entry for *hostname*. The host may be specified by name or by number, using Internet dot notation.

Options that modify the ARP translation tables (`-d`, `-f`, and `-s`) can be used only when the invoked command is granted the `sys_net_config` privilege. See `privileges(5)`.

Options

`-a` Display all of the current ARP entries. The definition for the flags in the table are:

- `d` Unverified; this is a local IP address that is currently undergoing Duplicate Address Detection. ARP will not respond to requests for this address until Duplicate Address Detection completes.
- `o` Old; this entry is aging away. If IP requests it again, a new ARP query will be generated. This state is used for detecting peer address changes.
- `y` Delayed; periodic address defense and conflict detection was unable to send a packet due to internal network use limits for non-traffic-related messages (100 packets per hour per interface). This occurs only on interfaces with very large numbers of aliases.
- `A` Authority; this machine is authoritative for this IP address. ARP will not accept updates from other machines for this entry.
- `L` Local; this is a local IP address configured on one of the machine's logical interfaces. ARP will defend this address if another node attempts to claim it.
- `M` Mapping; only used for the multicast entry for `224.0.0.0`
- `P` Publish; includes IP address for the machine and the addresses that have explicitly been added by the `-s` option. ARP will respond to ARP requests for this address.
- `S` Static; entry cannot be changed by learned information. This indicates that the `permanent` flag was used when creating the entry.
- `U` Unresolved; waiting for ARP response.

You can use the `-n` option with the `-a` option to disable the automatic numeric IP address-to-name translation. Use `arp -an` or `arp -na` to display numeric IP addresses. The `arp -a` option is equivalent to:

```
# netstat -p -f inet
```

...and `-an` and `-na` are equivalent to:

```
# netstat -pn -f inet
```

`-d` Delete an entry for the host called *hostname*.

`-f` Read the file named *filename* and set multiple entries in the ARP tables. Entries in the file should be of the form:

hostname MACaddress `[temp] [pub] [trail] [permanent]`

See the `-s` option for argument definitions.

`-s` Create an ARP entry for the host called *hostname* with the MAC address *MACaddress*. For example, an Ethernet address is given as six hexadecimal bytes separated by colons. The entry will not be subject to deletion by aging unless the word `temp` is specified in the command. If the word `pub` is specified, the entry will be published, which means that this system will respond to ARP requests for *hostname* even though the *hostname* is not its own. The word `permanent` indicates that the system will not accept MAC address changes for *hostname* from the network.

Solaris does not implement trailer encapsulation, and the word `trail` is accepted on entries for compatibility only.

`arp -s` can be used for a limited form of proxy ARP when a host on one of the directly attached networks is not physically present on a subnet. Another machine can then be configured to respond to ARP requests using `arp -s`. This is useful in certain SLIP configurations.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `ifconfig(1M)`, `netstat(1M)`, `attributes(5)`, `privileges(5)`, `arp(7P)`

Name aset – monitors or restricts accesses to system files and directories

Synopsis

```
aset  [-p] [-d aset_dir] [-l sec_level] [-n user@host]
 [-u userlist_file]
```

Description The Automated Security Enhancement Tool (ASET) is a set of administrative utilities that can improve system security by allowing the system administrators to check the settings of system files, including both the attributes (permissions, ownership, and the like) and the contents of the system files. It warns the users of potential security problems and, where appropriate, sets the system files automatically according to the security level specified.

The security level for `aset` can be specified by setting the `-l` command line option or the `ASETSECLEVEL` environment variable to be one of 3 values: `low`, `med`, or `high`. All the functionality operates based on the value of the security level.

At the `low` level, `aset` performs a number of checks and reports any potential security weaknesses.

At the `med` level, `aset` modifies some of the settings of system files and parameters, thus restricting system access, to reduce the risks from security attacks. Again reports the security weaknesses and the modifications performed to restrict access. This does not affect the operations of system services. All the system applications and commands maintain all of their original functionality.

At the `high` level, further restrictions are made to system access, rendering a very defensive system. Security practices which are not normally required are included. Many system files and parameters settings are modified to minimum access permissions. At this level, security is the foremost concern, higher than any other considerations that affect system behavior. The vast majority of system applications and commands maintain their functionality, although there may be a few that exhibit behaviors that are not familiar in normal system environment.

More exact definitions of what exactly `aset` does at each level can be found in the *System Administration Guide: Basic Administration*. The `asetenv(4)` file and the *master files* determine to a large extent what `aset` performs at each level, and can be used by the experienced administrators to redefine the definitions of the levels to suit their particular needs. See `asetmasters(4)`. These files are provided by default to fit most security conscious environments and in most cases provide adequate security safeguards without modification. They are, however, designed in a way that can be easily edited by experienced administrators with specific needs.

`aset` can be periodically activated at the specified security level with default definitions using the `-p` option. `aset` is automatically activated at a frequency specified by the administrator starting from a designated future time (see `asetenv(4)`). Without the `-p` option, `aset` operates only once immediately.

Options The following options are supported:

`-d` *aset_dir* — Specifies a working directory other than `/usr/aset` for ASET. `/usr/aset` is the default working directory. It is where ASET is installed, and is the *root* directory of all ASET utilities and data files. If another directory is to be used as the ASET working directory, you can either define it with the `-d` option, or set the `ASETDIR` environment variable before invoking `aset`. The command line option, if specified, overwrites the environment variable.

`-l` *sec_level* — Specifies a security level, `low`, `med`, or `high`, for `aset` to operate at. The default level is `low`. Each security level is explained in detail above. The level can also be specified by setting the `ASETSECLEVEL` environment variable before invoking `aset`. The command line option, if specified, overwrites the environment variable.

`-n` *user@host* — Notifies *user* at machine *host*. Send the output of `aset` to *user* through e-mail. If this option is not specified, the output is sent to the standard output. Note that this is not the reports of ASET, but rather an execution log including error messages if there are any. This output is typically brief. The actual reports of ASET are found in the `/usr/aset/reports/latest` directory. See the `-d` option.

`-p` — Schedules `aset` to be executed periodically. This adds an entry for `aset` in the `/etc/crontab` file. The `PERIODIC_SCHEDULE` environment variable in the `/usr/aset/asetenv` file is used to define the time for execution. See `crontab(1)` and `asetenv(4)`. If a `crontab(1)` entry for `aset` already exists, a warning is produced in the execution log.

`-u` *userlist_file* — Specifies a file containing a list of users. `aset` performs environment checks, for example, `UMASK` and `PATH` variables, on these users. By default, `aset` only checks for `root`. *userlist_file* is an ASCII text file. Each entry in the file is a line that contains only one user name (login name).

Usage The following paragraphs discuss the features provided by ASET. Hereafter, each feature is referred to as a *task*. The first task, `tune`, is executed only once per installation of ASET. The other tasks are executed periodically at the specified frequency.

tune Task — This task is used to tighten system file permissions. In standard releases, system files or directories have permissions defined to maximize open information sharing. In a more

security conscious environment, the administrator may want to redefine these permission settings to more restrictive values. `aset` allows resetting of these permissions, based on the specified security level. Generally, at the low level the permissions are set to what they should be as released. At the medium level, the permissions are tightened to ensure reasonable security that is adequate for most environments. At the high level they are further tightened to very restrictive access. The system files affected and the respective restrictions at different levels are configurable, using the `tune.low`, `tune.med`, and `tune.high` files. See `asetmasters(4)`.

cklist Task — System directories that contain relatively static files, that is, their contents and attributes do not change frequently, are examined and compared with a master description file. The `/usr/aset/masters/cklist.`*level* files are automatically generated the first time the `cklist` task is executed. See `asetenv(4)`. Any discrepancy found is reported. The directories and files are compared based on the following:

- owner and group
- permission bits
- size and checksum (if file)
- number of links
- last modification time

The lists of directories to check are defined in `asetenv(4)`, based on the specified security level, and are configurable using the `CKLISTPATH_LOW` , `CKLISTPATH_MED` , and `CKLISTPATH_HIGH` environment variables. Typically, the lower level lists are subsets of the higher level lists.

usrgrp Task — `aset` checks the consistency and integrity of user accounts and groups as defined in the `passwd` and `group` databases, respectively. Any potential problems are reported. Potential problems for the `passwd` file include:

- `passwd` file entries are not in the correct format.
- User accounts without a password.
- Duplicate user names.
- Duplicate user IDs. Duplicate user IDs are reported unless allowed by the `uid_alias` file. See `asetmasters(4)`).
- Invalid login directories.
- If C2 is enabled, check C2 hidden `passwd` format.

Potential problems for the group file include:

- Group file entries not in the right format.
- Duplicate group names.
- Duplicate group IDs.
- Null group passwords.

`aset` checks the local `passwd` file. If the `YPCHECK` environment variable is set to `true`, `aset` also checks the NIS `passwd` files. See `asetenv(4)`. Problems in the NIS passwd file are only reported and not corrected automatically. The checking is done for all three security levels except where noted.

sysconfTask `aset` checks various system configuration tables, most of which are in the `/etc` directory. `aset` checks and makes appropriate corrections for each system table at all three levels except where noted. The following discussion assumes familiarity with the various system tables. See the manual pages for these tables for further details.

The operations for each system table are:

`/etc/hosts.equiv` The default file contains a single "+" line, thus making every known host a trusted host, which is not advised for system security. `aset` performs the following operations:

`Low` Warns the administrators about the "+" line.

`Medium`

`High` Warns about and deletes that entry.

`/etc/inetd.conf` The following entries for system daemons are checked for possible weaknesses.

`tftp(1)` does not do any authentication. `aset` ensures that `in.tftpd(1M)` is started in the right directory on the server and is not running on clients. At the low level, it gives warnings if the mentioned condition is not true. At the medium and high levels it gives warnings, and changes (if necessary) the `in.tftpd` entry to include the `-s /tftpboot` option after ensuring the directory `/tftpboot` exists.

`ps(1)` and `netstat(1M)` provide valuable information to potential system crackers. These are disabled when `aset` is executed at a high security level.

`rexd` is also known to have poor authentication mechanism. `aset` disables `rexd` for medium and high security levels by commenting out this entry. If `rexd` is activated with the `-s` (secure RPC) option, it is not disabled.

`/etc/aliases` The decode alias of UUCP is a potential security weakness. `aset` disables the alias for medium and high security levels by commenting out this entry.

`/etc/default/login` The `CONSOLE=` line is checked to allow root login only at a specific terminal depending on the security level:

	`Low`	No action taken.
	`Medium`	
	`High`	Adds the following line to the file: `CONSOLE=/dev/console`
`/etc/vfstab`		`aset` checks for world-readable or writable device files for mounted file systems.
`/etc/dfs/dfstab`		`aset` checks for file systems that are exported without any restrictions.
`/etc/ftpd/ftpusers`		At high security level, `aset` ensures `root` is in `/etc/ftpd/ftpusers`, thus disallowing root from logging into `in.ftpd`(1M). If necessary, create `/etc/ftpd/ftpusers`. See `ftpusers`(4).
`/var/adm/utmpx`		`aset` makes these files not world-writable for the high level (some applications may not run properly with this setting.)
`/.rhosts`		The usage of a `.rhosts` file for the entire system is not advised. `aset` gives warnings for the low level and moves it to `/.rhosts.bak` for levels medium and high.

env Task `aset` checks critical environment variables for `root` and users specified with the `-u` *userlist_file* option by parsing the `/.profile`, `/.login`, and `/.cshrc` files. This task checks the `PATH` variable to ensure that it does not contain '`.`' as a directory, which makes an easy target for *trojan horse* attacks. It also checks that the directories in the `PATH` variable are not world-writable. Furthermore, it checks the `UMASK` variable to ensure files are not created as readable or writable by world. Any problems found by these checks are reported.

eeprom Task Newer versions of the EEPROM allow specification of a `secure` parameter. See `eeprom`(1M). `aset` recommends that the administrator sets the parameter to `command` for the medium level and to `full` for the high level. It gives warnings if it detects the parameter is not set adequately.

firewall Task At the high security level, `aset` takes proper measures such that the system can be safely used as a firewall in a network. This mainly involves disabling IP packets forwarding and making routing information invisible. Firewalling provides protection against external access to the network.

Environment Variables

`ASETDIR`	Specify ASET's working directory. Defaults to `/usr/aset`.
`ASETSECLEVEL`	Specify ASET's security level. Defaults to `low`.
`TASKS`	Specify the tasks to be executed by `aset`. Defaults to `all tasks`.

Files `/usr/aset/reports` directory of ASET reports

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWast

See Also `crontab(1)`, `ps(1)`, `tftp(1)`, `aset.restore(1M)`, `eeprom(1M)`, `in.ftpd(1M)`, `in.tftpd(1M)`, `netstat(1M)`, `asetenv(4)`, `asetmasters(4)`, `ftpusers(4)`, `attributes(5)`

System Administration Guide: Basic Administration

Name aset.restore – restores system files to their content before ASET is installed

Synopsis

```
aset.restore  [-d aset_dir]
```

Description `aset.restore` restores system files that are affected by the Automated Security Enhancement Tool (ASET) to their pre-ASET content. When ASET is executed for the first time, it saves and archives the original system files in the `/usr/aset/archives` directory. The `aset.restore` utility reinstates these files. It also deschedules ASET, if it is currently scheduled for periodic execution. See `asetenv(4)`.

If you have made changes to system files after running ASET, these changes are lost when you run `aset.restore`. If you want to be absolutely sure that you keep the existing system state, it is recommended that you back-up your system before using `aset.restore`.

You should use `aset.restore`, under the following circumstances:

You want to remove ASET permanently and restore the original system (if you want to deactivate ASET, you can remove it from scheduling).

You are unfamiliar with ASET and want to experiment with it. You can use `aset.restore` to restore the original system state.

When some major system functionality is not working properly and you suspect that ASET is causing the problem; you may want to restore the system to see if the problem persists without ASET.

`aset.restore` requires root privileges to execute.

Options The following options are supported:

`-d` *aset_dir* Specify the working directory for ASET. By default, this directory is `/usr/aset`. With this option the `archives` directory will be located under *aset_dir*.

Files `/usr/aset/archives` archive of system files prior to executing `aset`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWast

See Also `aset(1M)`, `asetenv(4)`, `attributes(5)`

System Administration Guide: Basic Administration

Name atohexlabel – convert a human readable label to its internal text equivalent

Synopsis

```
/usr/sbin/atohexlabel [human-readable-sensitivity-label]

/usr/sbin/atohexlabel -c [human-readable-clearance]
```

Description `atohexlabel` converts a human readable label into an internal text representation that is safe for storing in a public object. If no option is supplied, the label is assumed to be a sensitivity label.

Internal conversions can later be parsed to their same value. This internal form is often hexadecimal. The converted label is written to the standard output file. If no human readable label is specified, the label is read from the standard input file. The expected use of this command is emergency repair of labels that are stored in internal databases.

Options

`-c` Identifies the human readable label as a clearance.

Exit Status The following exit values are returned:

`0` On success.

`1` On failure, and writes diagnostics to the standard error file.

Files

`/etc/security/tsol/label_encodings`
The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	See below.

The command output is Committed for systems with the same `label_encodings` file. The command invocation is Committed for systems that implement the DIA MAC policy.

See Also `hextoalabel(1M)`, `label_to_str(3TSOL)`, `str_to_label(3TSOL)`, `label_encodings(4)`, `attributes(5)`

"How to Obtain the Hexadecimal Equivalent for a Label" in *Oracle Solaris Trusted Extensions Administrator's Procedures*

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

This file is part of the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. This file might not be applicable to other MAC policies that might be developed for future releases of Solaris Trusted Extensions software.

Name audit – control the behavior of the audit daemon

Synopsis

```
audit -n | -s | -t | -v [path]
```

Description The `audit` command is the system administrator's interface to maintaining the audit trail. The audit daemon can be notified to read the contents of the `audit_control(4)` file and re-initialize the current audit directory to the first directory listed in the `audit_control` file or to open a new audit file in the current audit directory specified in the `audit_control` file, as last read by the audit daemon. Reading `audit_control` also causes the `minfree` and `plugin` configuration lines to be re-read and reset within `auditd`. The audit daemon can also be signaled to close the audit trail and disable auditing.

Options

`-n` Notify the audit daemon to close the current audit file and open a new audit file in the current audit directory.

`-s` Notify the audit daemon to read the audit control file. The audit daemon stores the information internally. If the audit daemon is not running but audit has been enabled by means of `bsmconv(1M)`, the audit daemon is started.

`-t` Direct the audit daemon to close the current audit trail file, disable auditing, and die. Use `-s` to restart auditing.

`-v` *path* Verify the syntax for the audit control file stored in *path*. The `audit` command displays an approval message or outputs specific error messages for each error found.

Diagnostics The `audit` command will exit with `0` upon success and a positive integer upon failure.

Files

- `/etc/security/audit_user`
- `/etc/security/audit_control`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Stability	Evolving

See Also `bsmconv(1M)`, `praudit(1M)`, `audit(2)`, `audit_control(4)`, `audit_user(4)`, `attributes(5)`

See the section on Solaris Auditing in *System Administration Guide: Security Services*.

Notes The functionality described in this man page is available only if the Solaris Auditing feature has been enabled. See `bsmconv(1M)` for more information.

The `audit` command does not modify a process's preselection mask. It functions are limited to the following:

- affects which audit directories are used for audit data storage;

- specifies the minimum free space setting;
- resets the parameters supplied by means of the plugin directive.

For the `-s` option, `audit` validates the `audit_control` syntax and displays an error message if a syntax error is found. If a syntax error message is displayed, the audit daemon does not re-read `audit_control`. Because `audit_control` is processed at boot time, the `-v` option is provided to allow syntax checking of an edited copy of `audit_control`. Using `-v`, `audit` exits with 0 if the syntax is correct; otherwise, it returns a positive integer.

The `-v` option can be used in any zone, but the `-t`, `-s`, and `-n` options are valid only in local zones and, then, only if the `perzone` audit policy is set. See `auditd(1M)` and `auditconfig(1M)` for per-zone audit configuration.

Name auditconfig – configure auditing

Synopsis

```
auditconfig option...
```

Description `auditconfig` provides a command line interface to get and set kernel audit parameters.

This functionality is available only if the Basic Security Module (BSM) has been enabled. See `bsmconv(1M)` for more information.

The setting of the `perzone` policy determines the scope of the audit setting controlled by `auditconfig`. If `perzone` is set, then the values reflect the local zone except as noted. Otherwise, the settings are for the entire system. Any restriction based on the `perzone` setting is noted for each option to which it applies.

A non-global zone administrator can set all audit policy options except `perzone` and `ahlt`. `perzone` and `ahlt` apply only to the global zone; setting these policies requires the privileges of a global zone administrator. `perzone` and `ahlt` are described under the `-setpolicy` option, below.

Options

`-aconf`

Set the non-attributable audit mask from the `audit_control(4)` file. For example:

```
# auditconfig -aconf
Configured non-attributable events.
```

`-audit` *event sorf retval string*

This command constructs an audit record for audit event *event* using the process's audit characteristics containing a text token *string*. The return token is constructed from the *sorf* (success/failure flag) and the *retval* (return value). The event is type `char*`, the *sorf* is 0/1 for success/failure, *retval* is an errno value, *string* is type `*char`. This command is useful for constructing an audit record with a shell script. An example of this option:

```
# auditconfig -audit AUE_ftpd 0 0 "test string"
#

audit record from audit trail:
    header,76,2,ftp access,,Fri Dec 08 08:44:02 2000, + 669 msec
    subject,abc,root,other,root,other,104449,102336,235 197121 elbow
    text,test string
    return,success,0
```

`-chkaconf`

Checks the configuration of the non-attributable events set in the kernel against the entries in `audit_control(4)`. If the runtime class mask of a kernel audit event does not match the configured class mask, a mismatch is reported.

`-chkconf`

Check the configuration of kernel audit event to class mappings. If the runtime class mask of a kernel audit event does not match the configured class mask, a mismatch is reported.

`-conf`
Configure kernel audit event to class mappings. Runtime class mappings are changed to match those in the audit event to class database file.

`-getasid`
Prints the audit session ID of the current process. For example:

```
# auditconfig -getasid
audit session id = 102336
```

`-getaudit`
Returns the audit characteristics of the current process.

```
# auditconfig -getaudit
audit id = abc(666)
process preselection mask = lo(0x1000,0x1000)
terminal id (maj,min,host) = 235,197121,elbow(172.146.89.77)
audit session id = 102336
```

`-getauid`
Prints the audit ID of the current process. For example:

```
# auditconfig -getauid
audit id = abc(666)
```

`-getcar`
Prints current active root location (anchored from root [or local zone root] at system boot). For example:

```
# auditconfig -getcar
current active root = /
```

`-getclass` *event*
Display the preselection mask associated with the specified kernel audit event. *event* is the kernel event number or event name.

`-getcond`
Display the kernel audit condition. The condition displayed is the literal string `auditing` meaning auditing is enabled and turned on (the kernel audit module is constructing and queuing audit records); `noaudit`, meaning auditing is enabled but turned off (the kernel audit module is not constructing and queuing audit records); `disabled`, meaning that the audit module has not been enabled; or `nospace`, meaning there is no space for saving audit records. See auditon(2) and auditd(1M) for further information.

`-getestate` *event*
For the specified event (string or event number), print out classes *event* has been assigned. For example:

```
# auditconfig -getestate 20
audit class mask for event AUE_REBOOT(20) = 0x800
# auditconfig -getestate AUE_RENAME
audit class mask for event AUE_RENAME(42) = 0x30
```

`-getkaudit`
Get audit characteristics of the current zone. For example:

```
# auditconfig -getkaudit
audit id = unknown(-2)
process preselection mask = lo,na(0x1400,0x1400)
terminal id (maj,min,host) = 0,0,(0.0.0.0)
audit session id = 0
```

If the audit policy `perzone` is not set, the terminal id is that of the global zone. Otherwise, it is the terminal id of the local zone.

`-getkmask`
Get non-attributable pre-selection mask for the current zone. For example:

```
# auditconfig -getkmask
audit flags for non-attributable events = lo,na(0x1400,0x1400)
```

If the audit policy `perzone` is not set, the kernel mask is that of the global zone. Otherwise, it is that of the local zone.

`-getpinfo` *pid*
Display the audit ID, preselection mask, terminal ID, and audit session ID for the specified process.

`-getpolicy`
Display the kernel audit policy. The `ahlt` and `perzone` policies reflect the settings from the global zone. If `perzone` is set, all other policies reflect the local zone's settings. If `perzone` is not set, the policies are machine-wide.

`-getcwd`
Prints current working directory (anchored from zone root at system boot). For example:

```
# cd /usr/tmp
# auditconfig -getcwd
current working directory = /var/tmp
```

`-getqbufsz`
Get audit queue write buffer size. For example:

```
# auditconfig -getqbufsz
        audit queue buffer size (bytes) = 1024
```

`-getqctrl`
Get audit queue write buffer size, audit queue `hiwater` mark, audit queue `lowater` mark, audit queue `prod` interval (ticks).

```
# auditconfig -getqctrl
audit queue hiwater mark (records) = 100
audit queue lowater mark (records) = 10
audit queue buffer size (bytes) = 1024
audit queue delay (ticks) = 20
```

`-getqdelay`
Get interval at which audit queue is prodded to start output. For example:

```
# auditconfig -getqdelay
audit queue delay (ticks) = 20
```

`-getqhiwater`
Get high water point in undelivered audit records when audit generation will block. For example:

```
# ./auditconfig -getqhiwater
audit queue hiwater mark (records) = 100
```

`-getqlowater`
Get low water point in undelivered audit records where blocked processes will resume. For example:

```
# auditconfig -getqlowater
audit queue lowater mark (records) = 10
```

`-getstat`
Print current audit statistics information. For example:

```
# auditconfig -getstat
gen nona kern  aud  ctl  enq wrtn wblk rblk drop  tot  mem
910    1  725  184    0  910  910    0  231    0   88   48
```

See `auditstat(1M)` for a description of the headings in `-getstat` output.

`-gettid`
Print audit terminal ID for current process. For example:

```
# auditconfig -gettid
terminal id (maj,min,host) = 235,197121,elbow(172.146.89.77)
```

`-lsevent`
Display the currently configured (runtime) kernel and user level audit event information.

`-lspolicy`
Display the kernel audit policies with a description of each policy.

`-setasid` *session-ID* [*cmd*]
Execute shell or *cmd* with specified *session-ID*. For example:

```
# ./auditconfig -setasid 2000 /bin/ksh
#
# ./auditconfig -getpinfo 104485
audit id = abc(666)
process preselection mask = lo(0x1000,0x1000)
terminal id (maj,min,host) = 235,197121,elbow(172.146.89.77)
audit session id = 2000
```

`-setaudit` *audit-ID preselect_flags term-ID session-ID* [*cmd*]
Execute shell or *cmd* with the specified audit characteristics.

`-setauid` *audit-ID* [*cmd*]
Execute shell or *cmd* with the specified *audit–ID*.

`-setclass` *event audit_flag*[,*audit_flag* . . .]
Map the kernel event *event* to the classes specified by *audit_flags*. *event* is an event number or name. An *audit_flag* is a two character string representing an audit class. See `audit_control(4)` for further information. If `perzone` is not set, this option is valid only in the global zone.

`-setkaudit` *IP-address_type IP_address*
Set IP address of machine to specified values. *IP-address_type* is `ipv6` or `ipv4`.

If `perzone` is not set, this option is valid only in the global zone.

`-setkmask` *audit_flags*
Set non-attributes selection flags of machine.

If `perzone` is not set, this option is valid only in the global zone.

`-setpmask` *pid flags*
Set the preselection mask of the specified process. `flags` is the ASCII representation of the flags similar to that in `audit_control(4)`.

If `perzone` is not set, this option is valid only in the global zone.

`-setpolicy` [+|-]*policy_flag*[,*policy_flag* ...]
Set the kernel audit policy. A policy *policy_flag* is literal strings that denotes an audit policy. A prefix of + adds the policies specified to the current audit policies. A prefix of - removes the policies specified from the current audit policies. No policies can be set from a local zone unless the `perzone` policy is first set from the global zone. The following are the valid policy flag strings (`auditconfig -lspolicy` also lists the current valid audit policy flag strings):

`all`	Include all policies that apply to the current zone.
`ahlt`	Panic is called and the system dumps core if an asynchronous audit event occurs that cannot be delivered because the audit queue has reached the high-water mark or because there are insufficient resources to construct an audit record. By default, records are dropped and a count is kept of the number of dropped records.
`arge`	Include the `execv(2)` system call environment arguments to the audit record. This information is not included by default.
`argv`	Include the `execv(2)` system call parameter arguments to the audit record. This information is not included by default.

`cnt`	Do not suspend processes when audit resources are exhausted. Instead, drop audit records and keep a count of the number of records dropped. By default, process are suspended until audit resources become available.
`group`	Include the supplementary group token in audit records. By default, the group token is not included.
`none`	Include no policies. If used in other than the global zone, the `ahlt` and `perzone` policies are not changed.
`path`	Add secondary path tokens to audit record. These are typically the pathnames of dynamically linked shared libraries or command interpreters for shell scripts. By default, they are not included.
`perzone`	Maintain separate configuration, queues, and logs for each zone and execute a separate version of `auditd(1M)` for each zone.
`public`	Audit public files. By default, read-type operations are not audited for certain files which meet *public* characteristics: owned by root, readable by all, and not writable by all.
`trail`	Include the trailer token in every audit record. By default, the trailer token is not included.
`seq`	Include the sequence token as part of every audit record. By default, the sequence token is not included. The sequence token attaches a sequence number to every audit record.
`windata_down`	Include in an audit record any downgraded data moved between windows. This policy is available only if the system is configured with Trusted Extensions. By default, this information is not included.
`windata_up`	Include in an audit record any upgraded data moved between windows. This policy is available only if the system is configured with Trusted Extensions. By default, this information is not included.
`zonename`	Include the `zonename` token as part of every audit record. By default, the `zonename` token is not included. The `zonename` token gives the name of the zone from which the audit record was generated.

`-setqbufsz` *buffer_size*
Set the audit queue write buffer size (bytes).

`-setqctrl` *hiwater lowater bufsz interval*
Set the audit queue write buffer size (bytes), hiwater audit record count, lowater audit record count, and wakeup interval (ticks). Valid within a local zone only if `perzone` is set.

`-setqdelay` *interval*
Set the audit queue wakeup interval (ticks). This determines the interval at which the kernel pokes the audit queue, to write audit records to the audit trail. Valid within a local zone only if `perzone` is set.

`-setqhiwater` *hiwater*
Set the number of undelivered audit records in the audit queue at which audit record generation blocks. Valid within a local zone only if `perzone` is set.

`-setqlowater` *lowater*
Set the number of undelivered audit records in the audit queue at which blocked auditing processes unblock. Valid within a local zone only if `perzone` is set.

`-setsmask` *asid flags*
Set the preselection mask of all processes with the specified audit session ID. Valid within a local zone only if `perzone` is set.

`-setstat`
Reset audit statistics counters. Valid within a local zone only if `perzone` is set.

`-setumask` *auid flags*
Set the preselection mask of all processes with the specified audit ID. Valid within a local zone only if `perzone` is set.

Examples

EXAMPLE 1 Using `auditconfig`

The following is an example of an `auditconfig` program:

```
#
# map kernel audit event number 10 to the "fr" audit class
#
% auditconfig -setclass 10 fr

#
# turn on inclusion of exec arguments in exec audit records
#
% auditconfig -setpolicy +argv
```

Exit Status

`0` Successful completion.

`1` An error occurred.

Files

`/etc/security/audit_event`	Stores event definitions used in the audit system.
`/etc/security/audit_class`	Stores class definitions used in the audit system.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed

See Also audit(1M), auditd(1M), auditstat(1M), bsmconv(1M), praudit(1M), auditon(2), execv(2), audit_class(4), audit_control(4), audit_event(4), attributes(5), audit_binfile(5)

See the section on Solaris Auditing in *System Administration Guide: Security Services*.

Notes If plugin output is selected using audit_control(4), the behavior of the system with respect to the -setpolicy +cnt and the -setqhiwater options is modified slightly. If -setpolicy +cnt is set, data will continue to be sent to the selected plugin, even though output to the binary audit log is stopped, pending the freeing of disk space. If -setpolicy –cnt is used, the blocking behavior is as described under OPTIONS, above. The value set for the queue high water mark is used within auditd as the default value for its queue limits unless overridden by means of the qsize attribute as described in audit_control(4).

The auditconfig options that modify or display process-based information are not affected by the perzone policy. Those that modify system audit data such as the terminal id and audit queue parameters are valid only in the global zone, unless the perzone policy is set. The display of a system audit reflects the local zone if perzone is set. Otherwise, it reflects the settings of the global zone.

The -setcond option has been removed. Use audit(1M) to enable or disable auditing.

The -getfsize and -setfsize options have been removed. Use audit_binfile(5) p_fsize to set the audit file size.

Name auditd – audit daemon

Synopsis `/usr/sbin/auditd`

Description The audit daemon, `auditd`, controls the generation and location of audit trail files and the generation of syslog messages based on the definitions in `audit_control(4)`. If auditing is enabled, `auditd` reads the `audit_control(4)` file to do the following:

- reads the path to a library module for realtime conversion of audit data into syslog messages;
- reads other parameters specific to the selected plugin or plugins;
- obtains a list of directories into which audit files can be written;
- obtains the percentage limit for how much space to reserve on each filesystem before changing to the next directory.

`audit(1M)` is used to control `auditd`. It can cause `auditd` to:

- close the current audit file and open a new one;
- close the current audit file, re-read `/etc/security/audit_control` and open a new audit file;
- close the audit trail and terminate auditing.

Auditing Conditions The audit daemon invokes the program `audit_warn(1M)` under the following conditions with the indicated options:

`audit_warn soft` *pathname*
: The file system upon which *pathname* resides has exceeded the minimum free space limit defined in `audit_control(4)`. A new audit trail has been opened on another file system.

`audit_warn allsoft`
: All available file systems have been filled beyond the minimum free space limit. A new audit trail has been opened anyway.

`audit_warn hard` *pathname*
: The file system upon which *pathname* resides has filled or for some reason become unavailable. A new audit trail has been opened on another file system.

`audit_warn allhard` *count*
: All available file systems have been filled or for some reason become unavailable. The audit daemon will repeat this call to `audit_warn` at intervals of at least twenty seconds until space becomes available. *count* is the number of times that `audit_warn` has been called since the problem arose.

`audit_warn ebusy`
: There is already an audit daemon running.

`audit_warn tmpfile`
: The file `/etc/security/audit/audit_tmp` exists, indicating a fatal error.

`audit_warn nostart`
The internal system audit condition is `AUC_FCHDONE`. Auditing cannot be started without rebooting the system.

`audit_warn auditoff`
The internal system audit condition has been changed to not be `AUC_AUDITING` by someone other than the audit daemon. This causes the audit daemon to exit.

`audit_warn postsigterm`
An error occurred during the orderly shutdown of the auditing system.

`audit_warn getacdir`
There is a problem getting the directory list from `/etc/security/audit/audit_control`.

The audit daemon will hang in a sleep loop until this file is fixed.

Files `/etc/security/audit/audit_control`

`/etc/security/audit/audit_data`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also `audit(1M)`, `audit_warn(1M)`, `bsmconv(1M)`, `praudit(1M)`, `auditon(2)`, `auditsvc(2)`, `audit.log(4)`, `audit_control(4)`, `audit_data(4)`, `attributes(5)`

See the section on Solaris Auditing in *System Administration Guide: Security Services*.

Notes The functionality described in this man page is available only if the Solaris Auditing feature has been enabled. See `bsmconv(1M)` for more information.

`auditd` is loaded in the global zone at boot time if auditing is enabled. See `bsmconv(1M)`.

If the audit policy `perzone` is set, `auditd` runs in each zone, starting automatically when the local zone boots. If a zone is running when the `perzone` policy is set, auditing must be started manually in local zones. It is not necessary to reboot the system or the local zone to start auditing in a local zone. `auditd` can be started with "`/usr/sbin/audit -s`" and will start automatically with future boots of the zone.

When `auditd` runs in a local zone, the configuration is taken from the local zone's `/etc/security` directory's files: `audit_control`, `audit_class`, `audit_user`, `audit_startup`, and `audit_event`.

Configuration changes do not affect audit sessions that are currently running, as the changes do not modify a process's preselection mask. To change the preselection mask on a running process, use the –`setpmask` option of the `auditconfig` command (see `auditconfig(1M)`). If the user logs out and logs back in, the new configuration changes will be reflected in the next audit session.

Name auditreduce – merge and select audit records from audit trail files

Synopsis

```
auditreduce  [options] [audit-trail-file]...
```

Description `auditreduce` allows you to select or merge records from audit trail files. Audit files can be from one or more machines.

The merge function merges together audit records from one or more input audit trail files into a single output file. The records in an audit trail file are assumed to be sorted in chronological order (oldest first) and this order is maintained by `auditreduce` in the output file.

Unless instructed otherwise, `auditreduce` will merge the entire audit trail, which consists of all the audit trail files in the directory structure *audit_root_dir*/*/files (see `audit_control(4)` for details of the structure of the audit root). Unless specified with the `-R` or `-S` option, *audit_root_dir* defaults to `/etc/security/audit`. By using the file selection options it is possible to select some subset of these files, or files from another directory, or files named explicitly on the command line.

The select function allows audit records to be selected on the basis of numerous criteria relating to the record's content (see `audit.log(4)` for details of record content). A record must meet all of the *record-selection-option* criteria to be selected.

Audit Trail Filename Format

Any audit trail file not named on the command line must conform to the audit trail filename format. Files produced by the audit system already have this format. Output file names produced by `auditreduce` are in this format. It is:

start-time`.` *end-time*`.` *suffix*

where *start-time* is the 14-character timestamp of when the file was opened, *end-time* is the 14-character timestamp of when the file was closed, and *suffix* is the name of the machine which generated the audit trail file, or some other meaningful suffix (for example, `all`, if the file contains a combined group of records from many machines). The *end-time* can be the literal string `not_terminated`, to indicate that the file is still being written to by the audit system. Timestamps are of the form *yyyymmddhhmmss* (year, month, day, hour, minute, second). The timestamps are in Greenwich Mean Time (GMT).

Options

File Selection Options

The file selection options indicate which files are to be processed and certain types of special treatment.

`-A`

All of the records from the input files will be selected regardless of their timestamp. This option effectively disables the `-a`, `-b`, and `-d` options. This is useful in preventing the loss of records if the `-D` option is used to delete the input files after they are processed. Note, however, that if a record is *not* selected due to another option, then `-A` will not override that.

`-C`
Only process complete files. Files whose filename *end-time* timestamp is `not_terminated` are not processed (such a file is currently being written to by the audit system). This is useful in preventing the loss of records if `-D` is used to delete the input files after they are processed. It does not apply to files specified on the command line.

`-D` *suffix*
Delete input files after they are read if the entire run is successful. If `auditreduce` detects an error while reading a file, then that file is not deleted. If `-D` is specified, `-A`, `-C` and `-O` are also implied. *suffix* is given to the `-O` option. This helps prevent the loss of audit records by ensuring that all of the records are written, only complete files are processed, and the records are written to a file before being deleted. Note that if both `-D` and `-O` are specified in the command line, the order of specification is significant. The *suffix* associated with the latter specification is in effect.

`-M` *machine*
Allows selection of records from files with *machine* as the filename suffix. If `-M` is not specified, all files are processed regardless of suffix. `-M` can also be used to allow selection of records from files that contain combined records from many machines and have a common suffix (such as `all`).

`-N`
Select objects in *new mode*.This flag is off by default, thus retaining backward compatibility. In the existing, *old mode*, specifying the `-e`, `-f`, `-g`, `-r`, or `-u` flags would select not only actions taken with those IDs, but also certain objects owned by those IDs. When running in *new mode*, only actions are selected. In order to select objects, the `-o` option must be used.

`-O` *suffix*
Direct output stream to a file in the current `audit_root_dir` with the indicated suffix. *suffix* can alternatively contain a full pathname, in which case the last component is taken as the suffix, ahead of which the timestamps will be placed, ahead of which the remainder of the pathname will be placed. If the `-O` option is not specified, the output is sent to the standard output. When `auditreduce` places timestamps in the filename, it uses the times of the first and last records in the merge as the *start-time* and *end-time*.

`-Q`
Quiet. Suppress notification about errors with input files.

`-R` *pathname*
Specify the pathname of an alternate audit root directory *audit_root_dir* to be *pathname*. Therefore, rather than using `/etc/security/audit/*/`files by default, *pathname*`/*/`files will be examined instead.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-S` *server*
This option causes `auditreduce` to read audit trail files from a specific location (server directory). *server* is normally interpreted as the name of a subdirectory of the audit root, therefore `auditreduce` will look in *audit_root_dir*/*server*/files for the audit trail files. But if *server* contains any '`/`' characters, it is the name of a specific directory not necessarily contained in the audit root. In this case, *server*/files will be consulted. This option allows archived files to be manipulated easily, without requiring that they be physically located in a directory structure like that of `/etc/security/audit`.

`-V`
Verbose. Display the name of each file as it is opened, and how many records total were written to the output stream.

Record Selection Options

The record selection options listed below are used to indicate which records are written to the output file produced by `auditreduce`.

Multiple arguments of the same type are not permitted.

`-a` *date-time*
Select records that occurred at or after *date-time*. The *date-time* argument is described under `Option Arguments`, below. *date-time* is in local time. The `-a` and `-b` options can be used together to form a range.

`-b` *date-time*
Select records that occurred before *date-time*.

`-c` *audit-classes*
Select records by audit class. Records with events that are mapped to the audit classes specified by *audit-classes* are selected. Audit class names are defined in `audit_class(4)`. The *audit-classes* can be a comma separated list of `audit` *flags* like those described in `audit_control(4)`. Using the `audit` *flags,* one can select records based upon success and failure criteria.

`-d` *date-time*
Select records that occurred on a specific day (a 24-hour period beginning at 00:00:00 of the day specified and ending at 23:59:59). The day specified is in local time. The time portion of the argument, if supplied, is ignored. Any records with timestamps during that day are selected. If any hours, minutes, or seconds are given in *time,* they are ignored. `-d` can not be used with `-a` or `-b`.

`-e` *effective-user*
Select records with the specified *effective-user.*

`-f` *effective-group*
Select records with the specified *effective-group.*

`-g` *real-group*
Select records with the specified *real-group.*

`-j` *subject-ID*
Select records with the specified *subject-ID* where *subject-ID* is a process ID.

`-l` *label*
Select records with the specified label (or label range), as explained under "Option Arguments," below. This option is available only if the system is configured with the Solaris Trusted Extensions feature.

`-m` *event*
Select records with the indicated *event*. The *event* is the literal string or the *event* number.

`-o` *object_type=objectID_value*
Select records by object type. A match occurs when the record contains the information describing the specified *object_type* and the object ID equals the value specified by *objectID_value*. The allowable object types and values are as follows:

file=*pathname*
Select records containing file system objects with the specified pathname, where pathname is a comma separated list of regular expressions. If a regular expression is preceded by a tilde (~), files matching the expression are excluded from the output. For example, the option `file=~/usr/openwin,/usr,/etc` would select all files in `/usr` or `/etc` except those in `/usr/openwin`. The order of the regular expressions is important because auditreduce processes them from left to right, and stops when a file is known to be either selected or excluded. Thus the option `file= /usr, /etc, ~/usr/openwin` would select all files in `/usr` and all files in `/etc`. Files in `/usr/openwin` are not excluded because the regular expression `/usr` is matched first. Care should be given in surrounding the *pathname* with quotes so as to prevent the shell from expanding any tildes.

filegroup=*group*
Select records containing file system objects with *group* as the owning group.

fileowner=*user*
Select records containing file system objects with *user* as the owning user.

msgqid=*ID*
Select records containing message queue objects with the specified *ID* where *ID* is a message queue ID.

msgqgroup=*group*
Select records containing message queue objects with *group* as the owning or creating group.

msgqowner=*user*
Select records containing message queue objects with *user* as the owning or creating user.

pid=*ID*
Select records containing process objects with the specified *ID* where *ID* is a process ID. Process are objects when they are receivers of signals.

procgroup=*group*
Select records containing process objects with *group* as the real or effective group.

procowner=*user*
Select records containing process objects with *user* as the real or effective user.

semid=*ID*
Select records containing semaphore objects with the specified *ID* where *ID* is a semaphore ID.

semgroup=*group*
Select records containing semaphore objects with *group* as the owning or creating group.

semowner=*user*
Select records containing semaphore objects with *user* as the owning or creating user.

shmid=*ID*
Select records containing shared memory objects with the specified *ID* where *ID* is a shared memory ID.

shmgroup=*group*
Select records containing shared memory objects with *group* as the owning or creating group.

shmowner=*user*
Select records containing shared memory objects with *user* as the owning or creating user.

sock=*port_number*|*machine*
Select records containing socket objects with the specified *port_number* or the specified *machine* where *machine* is a machine name as defined in `hosts(4)`.

`-r` *real-user*
Select records with the specified *real-user*.

`-u` *audit-user*
Select records with the specified *audit-user*.

`-z` *zone-name*
Select records from the specified zone name. The zone name selection is case-sensitive.

When one or more *filename* arguments appear on the command line, only the named files are processed. Files specified in this way need not conform to the audit trail filename format. However, `-M`, `-S`, and `-R` must not be used when processing named files. If the *filename* is "–" then the input is taken from the standard input.

Option Arguments

audit-trail-file
An audit trail file as defined in `audit.log(4)`. An audit trail file not named on the command line must conform to the audit trail file name format. Audit trail files produced as output of `auditreduce` are in this format as well. The format is:

start-time . *end-time* . *suffix*

start-time is the 14 character time stamp denoting when the file was opened. *end-time* is the 14 character time stamp denoting when the file was closed. *end-time* can also be the literal string `not_terminated`, indicating the file is still be written to by the audit daemon or the file was not closed properly (a system crash or abrupt halt occurred). *suffix* is the name of the machine that generated the audit trail file (or some other meaningful suffix; for example, `all` would be a good suffix if the audit trail file contains a combined group of records from many machines).

date-time

The *date-time* argument to `-a`, `-b`, and `-d` can be of two forms: An absolute *date-time* takes the form:

yyyymmdd [*hh* [*mm* [*ss*]]]

where *yyyy* specifies a year (with 1970 as the earliest value), *mm* is the month (01-12), `dd` is the day (01-31), *hh* is the hour (00-23), *mm* is the minute (00-59), and *ss* is the second (00-59). The default is 00 for *hh*, *mm* and *ss*.

An offset can be specified as: +*n* `d`|`h`|`m`| `s` where *n* is a number of units, and the tags `d`, `h`, `m`, and `s` stand for days, hours, minutes and seconds, respectively. An offset is relative to the starting time. Thus, this form can only be used with the `-b` option.

event

The literal string or ordinal event number as found in `audit_event(4)`. If *event* is not found in the `audit_event` file it is considered invalid.

group

The literal string or ordinal group ID number as found in `group(4)`. If *group* is not found in the `group` file it is considered invalid. *group* can be negative.

label

The literal string representation of a MAC label or a range of two valid MAC labels. To specify a range, use "*x*;*y*" where *x* and *y* are valid MAC labels. Only those records that are fully bounded by *x* and *y* will be selected. If *x* or *y* is omitted, the default uses `ADMIN_LOW` or `ADMIN_HIGH`, respectively. Note that quotes must be used when specifying a range.

pathname

A regular expression describing a pathname.

user

The literal username or ordinal user ID number as found in `passwd(4)`. If the username is not found in the `passwd` file it is considered invalid. *user* can be negative.

Examples

EXAMPLE 1 The `auditreduce` command.

`praudit(1M)` is available to display audit records in a human-readable form.

This will display the entire audit trail in a human-readable form:

```
% auditreduce | praudit
```

If all the audit trail files are being combined into one large file, then deleting the original files could be desirable to prevent the records from appearing twice:

```
% auditreduce -V -D /etc/security/audit/combined/all
```

This displays what user `milner` did on April 13, 1988. The output will be displayed in a human-readable form to the standard output:

```
% auditreduce -d 19880413 -u milner | praudit
```

The above example might produce a large volume of data if `milner` has been busy. Perhaps looking at only login and logout times would be simpler. The `-c` option will select records from a specified class:

```
% auditreduce -d 19880413 -u milner -c lo | praudit
```

To see `milner`'s login/logout activity for April 13, 14, and 15 the following is used. The results are saved to a file in the current working directory. Note that the name of the output file will have `milnerlo` as the *suffix*, with the appropriate timestamp prefixes. Note that the long form of the name is used for the `-c` option:

```
% auditreduce -a 19880413 -b +3d -u milner -c login_logout -O milnerlo
```

To follow `milner`'s movement about the file system on April 13, 14, and 15 the `chdir` record types could be viewed. Note that in order to get the same time range as the above example we needed to specify the `-b` time as the day after our range. This is because `19880416` defaults to midnight of that day, and records before that fall on `0415`, the end-day of the range.

```
% auditreduce -a 19880413 -b 19880416 -u milner -m AUE_CHDIR | praudit
```

In this example the audit records are being collected in summary form (the login/logout records only). The records are being written to a summary file in a different directory than the normal audit root to prevent the selected records from existing twice in the audit root.

```
% auditreduce -d 19880330 -c lo -O /etc/security/audit_summary/logins
```

If activity for user ID 9944 has been observed, but that user is not known to the system administrator, then the following example will search the entire audit trail for any records generated by that user. `auditreduce` will query the system as to the current validity of ID 9944, and display a warning message if it is not currently active:

```
% auditreduce -O /etc/security/audit_suspect/user9944 -u 9944
```

To get an audit log of only the global zone:

EXAMPLE 1 The `auditreduce` command. *(Continued)*

```
% auditreduce -z global
```

Files

`/etc/security/audit/`*server*`/files/*`
location of audit trails, when stored

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`bsmconv(1M)`, `praudit(1M)`, `audit.log(4)`, `audit_class(4)`, `audit_control(4)`, `group(4)`, `hosts(4)`, `passwd(4)`, `attributes(5)`

See the section on Solaris Auditing in *System Administration Guide: Security Services*.

Diagnostics

`auditreduce` displays error messages if there are command line errors and then exit. If there are fatal errors during the run `auditreduce` displays an explanatory message and exit. In this case the output file might be in an inconsistent state (no trailer or partially written record) and `auditreduce` displays a warning message before exiting. Successful invocation returns `0` and unsuccessful invocation returns `1`.

Since `auditreduce` might be processing a large number of input files, it is possible that the machine-wide limit on open files will be exceeded. If this happens, `auditreduce` displays a message to that effect, give information on how many file there are, and exit.

If `auditreduce` displays a record's timestamp in a diagnostic message, that time is in local time. However, when filenames are displayed, their timestamps are in GMT.

Bugs

Conjunction, disjunction, negation, and grouping of record selection options should be allowed.

Notes

The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See `bsmconv(1M)` for more information.

The `-z` option should be used only if the audit policy `zonename` is set. If there is no zonename token, then no records will be selected.

Name audit_startup – audit subsystem initialization script

Synopsis

```
/etc/security/audit_startup
```

Description The `audit_startup` script is used to initialize the audit subsystem before the audit deamon is started. This script is configurable by the system administrator, and currently consists of a series of `auditconfig(1M)` commands to set the system default policy, and download the initial event to class mapping.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Obsolete Committed

See Also `auditconfig(1M)`, `auditd(1M)`, `bsmconv(1M)`, `attributes(5)`

See the section on Solaris Auditing in *System Administration Guide: Security Services*.

Notes The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See `bsmconv(1M)` for more information.

This command is Obsolete and may be removed and replaced with equivalent functionality in a future release of Solaris.

Name auditstat – display kernel audit statistics

Synopsis `auditstat [-c` *count*`] [-h` *numlines*`] [-i` *interval*`] [-n] [-v]`

Description `auditstat` displays kernel audit statistics. The fields displayed are as follows:

`aud` The total number of audit records processed by the `audit(2)` system call.

`ctl` This field is obsolete.

`drop` The total number of audit records that have been dropped. Records are dropped according to the kernel audit policy. See `auditon(2)`, AUDIT_CNT policy for details.

`enq` The total number of audit records put on the kernel audit queue.

`gen` The total number of audit records that have been constructed (not the number written).

`kern` The total number of audit records produced by user processes (as a result of system calls).

`mem` The total number of Kbytes of memory currently in use by the kernel audit module.

`nona` The total number of non-attributable audit records that have been constructed. These are audit records that are not attributable to any particular user.

`rblk` The total number of times that `auditsvc(2)` has blocked waiting to process audit data.

`tot` The total number of Kbytes of audit data written to the audit trail.

`wblk` The total number of times that user processes blocked on the audit queue at the high water mark.

`wrtn` The total number of audit records written. The difference between `enq` and `wrtn` is the number of outstanding audit records on the audit queue that have not been written.

Options

`-c` *count* Display the statistics a total of *count* times. If *count* is equal to zero, statistics are displayed indefinitely. A time interval must be specified.

`-h` *numlines* Display a header for every *numlines* of statistics printed. The default is to display the header every 20 lines. If *numlines* is equal to zero, the header is never displayed.

`-i` *interval* Display the statistics every *interval* where *interval* is the number of seconds to sleep between each collection.

`-n` Display the number of kernel audit events currently configured.

`-v` Display the version number of the kernel audit module software.

Exit Status `auditstat` returns `0` upon success and `1` upon failure.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `auditconfig(1M)`, `praudit(1M)`, `bsmconv(1M)`, `audit(2)`, `auditon(2)`, `auditsvc(2)`, `attributes(5)`

Notes The functionality described in this man page is available only if the Basic Security Module (BSM) has been enabled. See `bsmconv(1M)` for more information.

Name audit_warn – audit daemon warning script

Synopsis `/etc/security/audit_warn [`*option* `[`*arguments*`]]`

Description The `audit_warn` utility processes warning or error messages from the audit daemon. When a problem is encountered, the audit daemon, `auditd(1M)` calls `audit_warn` with the appropriate arguments. The *option* argument specifies the error type.

The system administrator can specify a list of mail recipients to be notified when an audit_warn situation arises by defining a mail alias called `audit_warn` in `aliases(4)`. The users that make up the `audit_warn` alias are typically the `audit` and `root` users.

Options The following options are supported:

`allhard` *count* — Indicates that the hard limit for all filesystems has been exceeded *count* times. The default action for this option is to send mail to the `audit_warn` alias only if the *count* is `1`, and to write a message to the machine console every time. It is recommended that mail *not* be sent every time as this could result in a the saturation of the file system that contains the mail spool directory.

`allsoft` — Indicates that the soft limit for all filesystems has been exceeded. The default action for this option is to send mail to the `audit_warn` alias and to write a message to the machine console.

`auditoff` — Indicates that someone other than the audit daemon changed the system audit state to something other than `AUC_AUDITING`. The audit daemon will have exited in this case. The default action for this option is to send mail to the `audit_warn` alias and to write a message to the machine console.

`ebusy` — Indicates that the audit daemon is already running. The default action for this option is to send mail to the `audit_warn` alias and to write a message to the machine console.

`getacdir` *count* — Indicates that there is a problem getting the directory list or plugin list from `audit_control(4)`. The audit daemon will hang in a sleep loop until the file is fixed. The default action for this option is to send mail to the `audit_warn` alias only if *count* is `1`, and to write a message to the machine console every time. It is recommended that mail *not* be sent every time as this could result in a the saturation of the file system that contains the mail spool directory.

`hard` *filename*
: Indicates that the hard limit for the file has been exceeded. The default action for this option is to send mail to the `audit_warn` alias and to write a message to the machine console.

`nostart`
: Indicates that auditing could not be started. The default action for this option is to send mail to the `audit_warn` alias and to write a message to the machine console. Some administrators may prefer to modify `audit_warn` to reboot the system when this error occurs.

`plugin` *name error count text*
: Indicates that an error occurred during execution of the `auditd` plugin *name*. The default action for this option is to send mail to the `audit_warn` alias only if *count* is 1, and to write a message to the machine console every time. (Separate counts are kept for each error type.) It is recommended that mail not be sent every time as this could result in the saturation of the file system that contains the mail spool directory. The *text* field provides the detailed error message passed from the plugin. The *error* field is one of the following strings:

  `load_error`
  : Unable to load the plugin *name*.

  `sys_error`
  : The plugin *name* is not executing due to a system error such as a lack of resources.

  `config_error`
  : No plugins loaded (including the binary file plugin, `audit_binfile(5)`) due to configuration errors in `audit_control(4)`. The name string is `--` to indicate that no plugin name applies.

  `retry`
  : The plugin *name* reports it has encountered a temporary failure. For example, the `audit_binfree.so` plugin uses `retry` to indicate that all directories are full.

  `no_memory`
  : The plugin *name* reports a failure due to lack of memory.

  `invalid`
  : The plugin *name* reports it received an invalid input.

  `failure`
  : The plugin *name* has reported an error as described in *text*.

`postsigterm`	Indicates that an error occurred during the orderly shutdown of the audit daemon. The default action for this option is to send mail to the `audit_warn` alias and to write a message to the machine console.
`soft` *filename*	Indicates that the soft limit for *filename* has been exceeded. The default action for this option is to send mail to the `audit_warn` alias and to write a message to the machine console.
`tmpfile`	Indicates that the temporary audit file already exists indicating a fatal error. The default action for this option is to send mail to the `audit_warn` alias and to write a message to the machine console.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr
Interface Stability	Evolving

The interface stability is evolving. The file content is unstable.

See Also `audit(1M)`, `auditd(1M)`, `bsmconv(1M)`, `aliases(4)`, `audit.log(4)`, `audit_control(4)`, `attributes(5)`

See the section on Solaris Auditing in *System Administration Guide: Security Services*.

Notes This functionality is available only if the Solaris Auditing feature has been enabled. See `bsmconv(1M)` for more information.

If the audit policy `perzone` is set, the `/etc/security/audit_warn` script for the local zone is used for notifications from the local zone's instance of `auditd`. If the `perzone` policy is not set, all `auditd` errors are generated by the global zone's copy of `/etc/security/audit_warn`.

Name automount – install automatic mount points

Synopsis

```
/usr/sbin/automount  [-t duration] [-v]
```

Description The `automount` utility installs `autofs` mount points and associates an automount map with each mount point. It starts the `automountd`(1M) daemon if it finds any non-trivial entries in either local or distributed automount maps and if the daemon is not already running. The `autofs` file system monitors attempts to access directories within it and notifies the `automountd`(1M) daemon. The daemon uses the map to locate a file system, which it then mounts at the point of reference within the `autofs` file system. A map can be assigned to an `autofs` mount using an entry in the `/etc/auto_master` map or a direct map.

If the file system is not accessed within an appropriate interval (`10` minutes by default), the `automountd` daemon unmounts the file system.

The file `/etc/auto_master` determines the locations of all `autofs` mount points. By default, this file contains three entries:

```
# Master map for automounter
#
+auto_master
/net          -hosts    -nosuid
/home         auto_home
```

The `+auto_master` entry is a reference to an external NIS or NIS+ master map. If one exists, then its entries are read as if they occurred in place of the `+auto_master` entry. The remaining entries in the master file specify a directory on which an `autofs` mount will be made followed by the automounter map to be associated with it. Optional mount options may be supplied as an optional third field in the each entry. These options are used for any entries in the map that do not specify mount options explicitly. The `automount` command is usually run without arguments. It compares the entries `/etc/auto_master` with the current list of `autofs` mounts in `/etc/mnttab` and adds, removes or updates `autofs` mounts to bring the `/etc/mnttab` up to date with the `/etc/auto_master`. At boot time it installs all `autofs` mounts from the master map. Subsequently, it may be run to install `autofs` mounts for new entries in the master map or the direct map, or to perform unmounts for entries that have been removed from these maps.

Automount with Solaris Trusted Extensions If a system is configured with Solaris Trusted Extensions, additional processing is performed to facilitate multilevel home directory access. A list of zones whose labels are dominated by the current zone is generated and default `auto_home` automount maps are generated if they do not currently exist. These automount maps are named `auto_home_`<*zonename*>, where <*zonename*> is the name of each zone's lower-level zone. An `autofs` mount of each such `auto_home` map is then performed, regardless of whether it is explicitly or implicitly listed in the master map. Instead of `autofs` mounting the standard `auto_home` map, the zone uses an `auto_home` file appended with its own zone name. Each zone's `auto_home` map is uniquely named so that it can be maintained and shared by all zones using a common name server.

By default, the home directories of lower-level zones are mounted read-only under `/zone/<`*zonename*`>/export/home` when each zone is booted. The default `auto_home_<`*zonename*`>` automount map specifies that path as the source directory for an `lofs` remount onto `/zone/<`*zonename*`>/home/<`*username*`>`. For example, the file `auto_home_public`, as generated from a higher level zone would contain:

+auto_home_public

* -fstype=lofs :/zone/public/export/home/&

When a home directory is referenced and the name does not match any other keys in the `auto_home_public` map, it will match this loopback mount specification. If this loopback match occurs and the name corresponds to a valid user whose home directory does not exist in the public zone, the directory is automatically created on behalf of the user.

Options The following options are supported:

`-t` *duration*	Specifies a *duration*, in seconds, that a file system is to remain mounted when not in use. The default is `10` minutes.
`-v`	Verbose mode. Notifies of `autofs` mounts, unmounts, or other non-essential information.

Usage

Map Entry Format A simple map entry (mapping) takes the form:

`key [ -`*mount-options* `] `*location* `. . .`

where `key` is the full pathname of the directory to mount when used in a direct map, or the simple name of a subdirectory in an indirect map. *mount-options* is a comma-separated list of `mount` options, and *location* specifies a file system from which the directory may be mounted. In the case of a simple NFS mount, the options that can be used are as specified in `mount_nfs(1M)`, and *location* takes the form:

host`:` *pathname*

host is the name of the host from which to mount the file system, and *pathname* is the absolute pathname of the directory to mount.

Options to other file systems are documented on the other `mount_*` reference manual pages, for example, `mount_cachefs(1M)`.

Replicated File Systems Multiple *location* fields can be specified for replicated NFS file systems, in which case `automount` and the kernel will each try to use that information to increase availability. If the read-only flag is set in the map entry, `automountd` mounts a list of locations that the kernel may use, sorted by several criteria. Only locations available at mount time will be mounted, and thus be available to the kernel. When a server does not respond, the kernel will switch to an alternate server. The sort ordering of `automount` is used to determine how the next server is

chosen. If the read-only flag is not set, `automount` will mount the best single location, chosen by the same sort ordering, and new servers will only be chosen when an unmount has been possible, and a remount is done. Servers on the same local subnet are given the strongest preference, and servers on the local net are given the second strongest preference. Among servers equally far away, response times will determine the order if no weighting factors (see below) are used.

If the list includes server locations using both the NFS Version 2 Protocol and the NFS Version 3 Protocol, `automount` will choose only a subset of the server locations on the list, so that all entries will be the same protocol. It will choose servers with the NFS Version 3 Protocol so long as an NFS Version 2 Protocol server on a local subnet will not be ignored. See the *System Administration Guide: IP Services* for additional details.

If each *location* in the list shares the same *pathname* then a single *location* may be used with a comma-separated list of hostnames:

```
hostname,hostname . . . : pathname
```

Requests for a server may be weighted, with the weighting factor appended to the server name as an integer in parentheses. Servers without a weighting are assumed to have a value of zero (most likely to be selected). Progressively higher values decrease the chance of being selected. In the example,

```
man -ro alpha,bravo,charlie(1),delta(4) : /usr/man
```

hosts `alpha` and `bravo` have the highest priority; host `delta` has the lowest.

Server proximity takes priority in the selection process. In the example above, if the server `delta` is on the same network segment as the client, but the others are on different network segments, then `delta` will be selected; the weighting value is ignored. The weighting has effect only when selecting between servers with the same network proximity. The automounter always selects the localhost over other servers on the same network segment, regardless of weighting.

In cases where each server has a different export point, the weighting can still be applied. For example:

```
man -ro alpha:/usr/man  bravo,charlie(1):/usr/share/man
     delta(3):/export/man
```

A mapping can be continued across input lines by escaping the NEWLINE with a backslash (\) Comments begin with a number sign (#) and end at the subsequent NEWLINE.

Map Key Substitution

The ampersand (&) character is expanded to the value of the `key` field for the entry in which it occurs. In this case:

```
jane sparcserver : /home/&
```

the & expands to `jane`.

Wildcard Key The asterisk (*) character, when supplied as the `key` field, is recognized as the catch-all entry. Such an entry will match any key not previously matched. For instance, if the following entry appeared in the indirect map for `/config`:

```
*         & : /export/config/&
```

this would allow automatic mounts in `/config` of any remote file system whose location could be specified as:

```
hostname : /export/config/hostname
```

Note that the wildcard key does not work in conjunction with the `-browse` option.

Variable Substitution Client specific variables can be used within an `automount` map. For instance, if `$HOST` appeared within a map, `automount` would expand it to its current value for the client's host name. Supported variables are:

`ARCH`	The application architecture is derived from the output of `uname -m`	The architecture name. For example, `sun4` on a sun4u machine.
`CPU`	The output of `uname -p`	The processor type. For example, “sparc”
`HOST`	The output of `uname -n`	The host name. For example, “biggles”
`OSNAME`	The output of `uname -s`	The OS name. For example, “SunOS”
`OSREL`	The output of `uname -r`	The OS release name. For example “5.3”
`OSVERS`	The output of `uname -v`	The OS version. For example, “beta1.0”
`NATISA`	The output of `isainfo -n`	The native instruction set architecture for the system. For example, “sparcv9”

If a reference needs to be protected from affixed characters, you can surround the variable name with curly braces (`{ }`).

Multiple Mounts

A multiple mount entry takes the form:

```
key [-mount-options] [ [mountpoint] [-mount-options] location. . . ] . . .
```

The initial /[*mountpoint*] is optional for the first mount and mandatory for all subsequent mounts. The optional *mountpoint* is taken as a pathname relative to the directory named by `key`. If *mountpoint* is omitted in the first occurrence, a *mountpoint* of / (root) is implied.

Given an entry in the indirect map for `/src`

```
beta     -ro\
  /           svr1,svr2:/export/src/beta  \
  /1.0        svr1,svr2:/export/src/beta/1.0  \
  /1.0/man    svr1,svr2:/export/src/beta/1.0/man
```

All offsets must exist on the server under `beta`. `automount` will automatically mount `/src/beta`, `/src/beta/1.0`, and `/src/beta/1.0/man`, as needed, from either `svr1` or `svr2`, whichever host is nearest and responds first.

Other File System Types

The automounter assumes NFS mounts as a default file system type. Other file system types can be described using the `fstype` mount option. Other mount options specific to this file system type can be combined with the `fstype` option. The location field must contain information specific to the file system type. If the location field begins with a slash, a colon character must be prepended, for instance, to mount a CD file system:

```
cdrom -fstype=hsfs,ro   : /dev/sr0
```

or to perform an `autofs` mount:

```
src -fstype=autofs    auto_src
```

Use this procedure only if you are not using Volume Manager.

Mounts using CacheFS are most useful when applied to an entire map as map defaults. The following entry in the master map describes cached home directory mounts. It assumes the default location of the cache directory, `/cache`.

```
/home auto_home -fstype=cachefs,backfstype=nfs
```

See the `NOTES` section for information on option inheritance.

Indirect Maps

An indirect map allows you to specify mappings for the subdirectories you wish to mount under the `directory` indicated on the command line. In an indirect map, each `key` consists of a simple name that refers to one or more file systems that are to be mounted as needed.

Direct Maps

Entries in a direct map are associated directly with `autofs` mount points. Each `key` is the full pathname of an `autofs` mount point. The direct map as a whole is not associated with any single directory.

Direct maps are distinguished from indirect maps by the `/-` key. For example:

```
# Master map for automounter
#
+auto_master
/net            -hosts          -nosuid,nobrowse
/home           auto_home       -nobrowse
/-              auto_direct
```

Included Maps

The contents of another map can be included within a map with an entry of the form

+mapname

If *mapname* begins with a slash, it is assumed to be the pathname of a local file. Otherwise, the location of the map is determined by the policy of the name service switch according to the entry for the automounter in `/etc/nsswitch.conf`, such as

```
automount: files nis
```

If the name service is `files`, then the name is assumed to be that of a local file in `/etc`. If the key being searched for is not found in the included map, the search continues with the next entry.

Special Maps

There are two special maps available: `-hosts` and `-null`. The `-hosts` map is used with the `/net` directory and assumes that the map key is the hostname of an NFS server. The `automountd` daemon dynamically constructs a map entry from the server's list of exported file systems. References to a directory under `/net/hermes` will refer to the corresponding directory relative to `hermes` root.

The `-null` map cancels a previous map for the directory indicated. This is most useful in the `/etc/auto_master` for cancelling entries that would otherwise be inherited from the `+auto_master` include entry. To be effective, the `-null` entries must be inserted before the included map entry.

Executable Maps

Local maps that have the execute bit set in their file permissions will be executed by the automounter and provided with a key to be looked up as an argument. The executable map is expected to return the content of an automounter map entry on its stdout or no output if the entry cannot be determined. A direct map cannot be made executable.

Configuration and the auto_master Map

When initiated without arguments, `automount` consults the master map for a list of `autofs` mount points and their maps. It mounts any `autofs` mounts that are not already mounted, and unmounts `autofs` mounts that have been removed from the master map or direct map.

The master map is assumed to be called `auto_master` and its location is determined by the name service switch policy. Normally the master map is located initially as a local file `/etc/auto_master`.

Browsing

The `automountd` supports the browsability of indirect maps. This allows all of the potential mount points to be visible, whether or not they are mounted. The `-nobrowse` option can be added to any indirect `autofs` map to disable browsing. For example:

```
/net     -hosts      -nosuid,nobrowse
/home    auto_home
```

In this case, any *hostnames* would only be visible in `/net` after they are mounted, but all potential mount points would be visible under `/home`. The `-browse` option enables browsability of `autofs` file systems. This is the default for all indirect maps.

The `-browse` option does not work in conjunction with the wildcard key.

Restricting Mount Maps

Options specified for a map are used as the default options for all the entries in that map. They are ignored when map entries specify their own mount options.

In some cases, however, it is desirable to force `nosuid`, `nodevices`, `nosetuid`, or `noexec` for a complete mount map and its submounts. This can be done by specifying the additional mount option, `-restrict`.

```
/home     auto_home       -restrict,nosuid,hard
```

The `-restrict` option forces the inheritance of all the restrictive options `nosuid`, `nodevices`, `nosetuid`, and `noexec` as well as the restrict option itself. In this particular example, the `nosuid` and `restrict` option are inherited but the `hard` option is not. The `restrict` option also prevents the execution of "executable maps" and is enforced for auto mounts established by programs with fewer than all privileges available in their zone.

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

Files

`/etc/auto_master`	Master automount map.
`/etc/auto_home`	Map to support automounted home directories.
`/etc/default/autofs`	Supplies default values for parameters for `automount` and `automountd`. See `autofs(4)`.
`/etc/nsswitch.conf`	Name service switch configuration file. See `nsswitch.conf(4)`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`isainfo(1)`, `ls(1)`, `svcs(1)`, `uname(1)`, `automountd(1M)`, `mount(1M)`, `mount_cachefs(1M)`, `mount_nfs(1M)`, `svcadm(1M)`, `autofs(4)`, `attributes(5)`, `nfssec(5)`, `smf(5)`

System Administration Guide: IP Services

Notes `autofs` mount points must not be hierarchically related. `automount` does not allow an `autofs` mount point to be created within another `autofs` mount.

Since each direct map entry results in a new `autofs` mount such maps should be kept short.

Entries in both direct and indirect maps can be modified at any time. The new information is used when `automountd` next uses the map entry to do a mount.

New entries added to a master map or direct map will not be useful until the automount command is run to install them as new `autofs` mount points. New entries added to an indirect map may be used immediately.

As of the Solaris 2.6 release, a listing (see `ls(1)`) of the `autofs` directory associated with an indirect map shows all potential mountable entries. The attributes associated with the potential mountable entries are temporary. The real file system attributes will only be shown once the file system has been mounted.

Default mount options can be assigned to an entire map when specified as an optional third field in the master map. These options apply only to map entries that have no mount options. Note that map entities with options override the default options, as at this time, the options do not concatenate. The concatenation feature is planned for a future release.

When operating on a map that invokes an NFS mount, the default number of retries for the automounter is 0, that is, a single mount attempt, with no retries. Note that this is significantly different from the default (10000) for the `mount_nfs(1M)` utility.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same.

The `automount` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/filesystem/autofs:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name automountd – autofs mount/unmount daemon

Synopsis

```
automountd [-Tvn] [-D name=value]
```

Description `automountd` is an RPC server that answers file system mount and unmount requests from the `autofs` file system. It uses local files or name service maps to locate file systems to be mounted. These maps are described with the `automount(1M)` command.

If `automount` finds any non-trivial entries in either the local or distributed automount maps and if the daemon is not running already, the `automountd` daemon is automatically invoked by `automount(1M)`. `automountd` enables the `svc:/network/nfs/nlockmgr` service (`lockd(1M)`), and the `svc:/network/nfs/status` service (`statd(1M)`), if NFS mounts need to be done.

At startup, the `automountd` daemon is invoked as is invoked as the `system/filesystem/autofs:default` service. See NOTES.

Options The following options are supported:

`-D` *name=value*	Assign *value* to the indicated `automount` map substitution variable. These assignments cannot be used to substitute variables in the master map `auto_master`.
`-n`	Turn off browsing for all `autofs` mount points. This option overrides the `-browse` `autofs` map option on the local host.
`-T`	Trace. Expand each RPC call and display it on the standard output.
`-v`	Verbose. Log status messages to the console.

Usage See `largefile(5)` for the description of the behavior of `automountd` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/auto_master`	Master map for automounter.
`/etc/default/autofs`	Supplies default values for parameters for `automount` and `automountd`. See `autofs(4)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `svcs(1)`, `automount(1M)`, `svcadm(1M)`, `autofs(4)`, `attributes(5)`, `largefile(5)`, `smf(5)`

Notes The `automountd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/filesystem/autofs
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using svcs(1). If it is disabled, it is enabled by automount(1M) unless the application/auto_enable property is set to false.

Name autopush – configures lists of automatically pushed STREAMS modules

Synopsis

```
autopush -f filename

autopush -g -M major -m minor

autopush -r -M major -m minor
```

Description The `autopush` command configures the list of modules to be automatically pushed onto the stream when a device is opened. It can also be used to remove a previous setting or get information on a setting.

Options The following options are supported:

`-f` *filename*
: Sets up the `autopush` configuration for each driver according to the information stored in *filename*. An `autopush` file consists of lines of four or more fields, separated by spaces as shown below:

  major minor last-minor module1 module2 . . . module8

  The first field is a string that specifies the *major* device name, as listed in the `/kernel/drv` directory. The next two fields are integers that specify the *minor* device number and *last-minor* device number. The fields following represent the names of modules. If *minor* is `–1`, then all minor devices of a major driver specified by *major* are configured, and the value for *last-minor* is ignored. If *last-minor* is `0`, then only a single minor device is configured. To configure a range of minor devices for a particular major, *minor* must be less than *last-minor*.

  The remaining fields list the names of modules to be automatically pushed onto the stream when opened, along with the position of an optional anchor. The maximum number of modules that can be pushed is eight. The modules are pushed in the order they are specified. The optional special character sequence `[anchor]` indicates that a STREAMS anchor should be placed on the stream at the module previously specified in the list; it is an error to specify more than one anchor or to have an anchor first in the list.

  A nonzero exit status indicates that one or more of the lines in the specified file failed to complete successfully.

`-g`
: Gets the current configuration setting of a particular *major* and *minor* device number specified with the `-M` and `-m` options respectively and displays the autopush modules associated with it. It will also return the starting minor device number if the request corresponds to a setting of a range (as described with the `-f` option).

`-m` *minor*
: Specifies the minor device number.

`-M` *major*
: Specifies the major device number.

`-r` Removes the previous configuration setting of the particular *major* and *minor* device number specified with the `-M` and `-m` options respectively. If the values of *major* and *minor* correspond to a previously established setting of a range of minor devices, where *minor* matches the first minor device number in the range, the configuration would be removed for the entire range.

Exit Status The following exit values are returned:

0 Successful completion.

non-zero An error occurred.

Examples **EXAMPLE 1** Using the `autopush` command.

The following example gets the current configuration settings for the *major* and *minor* device numbers as indicated and displays the `autopush` modules associated with them for the character-special device `/dev/term/a`:

```
example# autopush -g -M 29 -m 0
Major     Minor     Lastminor     Modules
  29        0         1           ldterm  ttcompat
```

Files `/etc/iu.ap`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `bdconfig(1M)`, `ttymon(1M)`, `attributes(5)`, `ldterm(7M)`, `sad(7D)`, `streamio(7I)`, `ttcompat(7M)`

STREAMS Programming Guide

Name bart – basic audit reporting tool

Synopsis

```
/usr/bin/bart create [ -n] [-R root_directory]
     [-r rules_file | -]

/usr/bin/bart create [-n] [-R root_directory] -I
     [file_name]...

/usr/bin/bart compare [-i attribute ] [-p]
     [-r rules_file | -] control-manifest test-manifest
```

Description `bart`(1M) is a tool that performs a file-level check of the software contents of a system.

You can also specify the files to track and the types of discrepancies to flag by means of a rules file, `bart_rules`. See `bart_rules`(4).

The `bart` utility performs two basic functions:

`bart create` The manifest generator tool takes a file-level *snapshot* of a system. The output is a catalog of file attributes referred to as a *manifest*. See `bart_manifest`(4).

You can specify that the list of files be cataloged in three ways. Use `bart create` with no options, specify the files by name on the command line, or create a rules file with directives that specify which the files to monitor. See `bart_rules`(4).

By default, the manifest generator catalogs all attributes of all files in the root (`/`) file system. File systems mounted on the root file system are cataloged only if they are of the same type as the root file system.

For example, `/`, `/usr`, and `/opt` are separate UFS file systems. `/usr` and `/opt` are mounted on `/`. Therefore, all three file systems are cataloged. However, `/tmp`, also mounted on `/`, is not cataloged because it is a TMPFS file system. Mounted CD-ROMs are not cataloged since they are HSFS file systems.

`bart compare` The report tool compares two manifests. The output is a list of per-file attribute discrepancies. These discrepancies are the differences between two manifests: a control manifest and a test manifest.

A discrepancy is a change to any attribute for a given file cataloged by both manifests. A new file or a deleted file in a manifest is reported as a discrepancy.

The reporting mechanism provides two types of output: verbose and programmatic. Verbose output is localized and presented on multiple lines, while programmatic output is more easily parsable by other programs. See `OUTPUT`.

By default, the report tool generates verbose output where all discrepancies are reported except for modified directory timestamps (`dirmtime` attribute).

To ensure consistent and accurate comparison results, *control-manifest* and *test-manifest* must be built with the same rules file.

Use the rules file to ignore specified files or subtrees when you generate a manifest or compare two manifests. Users can compare manifests from different perspectives by re-running the `bart compare` command with different rules files.

Options The following options are supported:

`-i` *attribute ...* Specify the file attributes to be ignored globally. Specify attributes as a comma separated list.

This option produces the same behavior as supplying the file attributes to a global `IGNORE` keyword in the rules file. See `bart_rules`(4).

`-I` [*file_name...*] Specify the input list of files. The file list can be specified at the command line or read from standard input.

`-n` Prevent computation of content signatures for all regular files in the file list.

`-p` Display manifest comparison output in "programmatic mode," which is suitable for programmatic parsing. The output is not localized.

`-r` *rules_file* Use *rules_file* to specify which files and directories to catalog, and to define which file attribute discrepancies to flag. If *rules_file* is `-`, then the rules are read from standard input. See `bart_rules`(4) for the definition of the syntax.

`-R` *root_directory* Specify the root directory for the manifest. All paths specified by the rules, and all paths reported in the manifest, are relative to *root_directory*.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones`(5).

Operands bart allows quoting of operands. This is particularly important for white-space appearing in subtree and subtree modifier specifications.

The following operands are supported:

control-manifest Specify the manifest created by `bart create` on the control system.

test-manifest Specify the manifest created by `bart create` on the test system.

Output The `bart create` and `bart compare` commands write output to standard output, and write error messages to standard error.

The `bart create` command generates a system manifest. See `bart_manifest`(4).

When the `bart compare` command compares two system manifests, it generates a list of file differences. By default, the comparison output is localized. However, if the `-p` option is specified, the output is generated in a form that is suitable for programmatic manipulation.

Default Format

filename
attribute `control:`*xxxx* `test:`*yyyy*

filename Name of the file that differs between *control-manifest* and *test-manifest*. For file names that contain embedded whitespace or newline characters, see `bart_manifest`(4).

attribute The name of the file attribute that differs between the manifests that are compared. *xxxx* is the attribute value from *control-manifest*, and *yyyy* is the attribute value from *test-manifest*. When discrepancies for multiple attributes occur for the same file, each difference is noted on a separate line.

The following attributes are supported:

acl	ACL attributes for the file. For a file with ACL attributes, this field contains the output from `acltotext()`.
all	All attributes.
contents	Checksum value of the file. This attribute is only specified for regular files. If you turn off context checking or if checksums cannot be computed, the value of this field is `-`.
dest	Destination of a symbolic link.
devnode	Value of the device node. This attribute is for character device files and block device files only.
dirmtime	Modification time in seconds since 00:00:00 UTC, January 1, 1970 for directories.
gid	Numerical group ID of the owner of this entry.
lnmtime	Creation time for links.
mode	Octal number that represents the permissions of the file.
mtime	Modification time in seconds since 00:00:00 UTC, January 1, 1970 for files.
size	File size in bytes.
type	Type of file.

uid Numerical user ID of the owner of this entry.

The following default output shows the attribute differences for the `/etc/passwd` file. The output indicates that the `size`, `mtime`, and `contents` attributes have changed.

```
/etc/passwd:
  size  control:74  test:81
  mtime  control:3c165879  test:3c165979
  contents  control:daca28ae0de97afd7a6b91fde8d57afa
test:84b2b32c4165887355317207b48a6ec7
```

Programmatic Format *filename attribute control-val test-val* [*attribute control-val test-val*]*

filename Same as *filename* in the default format.

attribute control-val test-val A description of the file attributes that differ between the control and test manifests for each file. Each entry includes the attribute value from each manifest. See `bart_manifest`(4) for the definition of the attributes.

Each line of the programmatic output describes all attribute differences for a single file.

The following programmatic output shows the attribute differences for the `/etc/passwd` file. The output indicates that the `size`, `mtime`, and `contents` attributes have changed.

```
/etc/passwd size 74 81 mtime 3c165879 3c165979
contents daca28ae0de97afd7a6b91fde8d57afa 84b2b32c4165887355317207b48a6ec7
```

Exit Status

Manifest Generator The manifest generator returns the following exit values:

`0` Success

`1` Non-fatal error when processing files; for example, permission problems

`>1` Fatal error; for example, invalid command-line options

Report Tool The report tool returns the following exit values:

`0` No discrepancies reported

`1` Discrepancies found

`>1` Fatal error executing comparison

Examples

EXAMPLE 1 Creating a Default Manifest Without Computing Checksums

The following command line creates a default manifest, which consists of all files in the `/` file system. The `-n` option prevents computation of checksums, which causes the manifest to be generated more quickly.

EXAMPLE 1 Creating a Default Manifest Without Computing Checksums *(Continued)*

```
bart create -n
```

EXAMPLE 2 Creating a Manifest for a Specified Subtree

The following command line creates a manifest that contains all files in the `/home/nickiso` subtree.

```
bart create -R /home/nickiso
```

EXAMPLE 3 Creating a Manifest by Using Standard Input

The following command line uses output from the `find`(1) command to generate the list of files to be cataloged. The `find` output is used as input to the `bart create` command that specifies the `-I` option.

```
find /home/nickiso -print | bart create -I
```

EXAMPLE 4 Creating a Manifest by Using a Rules File

The following command line uses a rules file, `rules`, to specify the files to be cataloged.

```
bart create -r rules
```

EXAMPLE 5 Comparing Two Manifests and Generating Programmatic Output

The following command line compares two manifests and produces output suitable for parsing by a program.

```
bart compare -p manifest1 manifest2
```

EXAMPLE 6 Comparing Two Manifests and Specifying Attributes to Ignore

The following command line compares two manifests. The `dirmtime`, `lnmtime`, and `mtime` attributes are not compared.

```
bart compare -i dirmtime,lnmtime,mtime manifest1 manifest2
```

EXAMPLE 7 Comparing Two Manifests by Using a Rules File

The following command line uses a rules file, `rules`, to compare two manifests.

```
bart compare -r rules manifest1 manifest2
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbart

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also cksum(1), digest(1), find(1), bart_manifest(4), bart_rules(4), attributes(5)

Notes The file attributes of certain system libraries can be temporarily altered by the system as it boots. To avoid triggering false warnings, you should compare manifests only if they were both created with the system in the same state; that is, if both were created in single-user or both in multi-user.

Name bdconfig – configures the bd (buttons and dials) stream

Synopsis

```
bdconfig  [startup] [off] [on] [term] [status] [verbose]
```

Description The `bdconfig` utility is responsible for configuring the autopush facility and defining to the system what serial device to use for the `bd` stream.

Options If no options are given, then an interactive mode is assumed. In this mode the current status is presented along with this usage line, and a series of interactive questions asked to determine the user's desires.

Root privilege is required to change the configuration. The status option does not require root privilege. `bdconfig` can be installed as a setuid root program.

The non-interactive options below can be given in any order.

term — Specify to the system the serial device for `bd` use. This option implies the `on` option unless the `off` option is present.

`iff` — Reconfigure the configured term for tty use.

`on` — Reconfigure the configured term for `bd` use. If *term* has not been previously specified, interactive questions are asked to determine the user's desires.

`startup` — Configure as was last configured before the system went down. This option is used by the startup script, and precludes the use of the `on`, `off`, and *term* options. This option implies non-interactive mode.

`status` — Emit the current configuration in terms of the words used as options: `off`, `on`, `/dev/term/a`, `/dev/term/b`, and so forth. This option implies non interactive mode.

`verbose` — `bdconfig` describes what it finds and what it is doing.

Exit Status The `bdconfig` utility returns `0` on success, `1` on general error, and `2` on argument error.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdialh

See Also `autopush(1M)`, `attributes(5)`, `x_buttontest(6)`, `x_dialtest(6)`, `bd(7M)`, `sad(7D)`, `streamio(7I)`

Notes All `bdconfig` does is configure the AUTOPUSH facility. `bdconfig` does not actually manipulate the serial port or stream in any way. Only the first open of a dismantled stream will see the effects of a previously run `bdconfig`.

The `bdconfig` utility is silent except for error messages unless:

a) invoked with no args: status / usage line emitted

b) interactive modes are invoked as described above

c) the verbose option is used

Bugs The interface does not support more than one dialbox and one buttonbox, both of which must be on the same serial device.

There should be a library routine to read, parse, and validate records in the `iu.ap` file, so that `bdconfig` could return to the appropriate record in `iu.ap` as the default configuration.

Name boot – start the system kernel or a standalone program

Synopsis

SPARC
```
boot [OBP names] [file] [-aLV] [-F object] [-D default-file]
     [-Z dataset] [boot-flags] [—] [client-program-args]
```

x86
```
kernel$ /platform/i86pc/kernel/$ISADIR/unix [boot-args]
     [-B prop=val [,val...]]
```

Description Bootstrapping is the process of loading and executing a standalone program. For the purpose of this discussion, bootstrapping means the process of loading and executing the bootable operating system. Typically, the standalone program is the operating system kernel (see `kernel(1M)`), but any standalone program can be booted instead. On a SPARC-based system, the diagnostic monitor for a machine is a good example of a standalone program other than the operating system that can be booted.

If the standalone is identified as a dynamically-linked executable, `boot` will load the interpreter (linker/loader) as indicated by the executable format and then transfer control to the interpreter. If the standalone is statically-linked, it will jump directly to the standalone.

Once the kernel is loaded, it starts the UNIX system, mounts the necessary file systems (see `vfstab(4)`), and runs `/sbin/init` to bring the system to the "initdefault" state specified in `/etc/inittab`. See `inittab(4)`.

SPARC Bootstrap Procedure On SPARC based systems, the bootstrap procedure on most machines consists of the following basic phases.

After the machine is turned on, the system firmware (in PROM) executes power-on self-test (POST). The form and scope of these tests depends on the version of the firmware in your system.

After the tests have been completed successfully, the firmware attempts to autoboot if the appropriate flag has been set in the non-volatile storage area used by the firmware. The name of the file to load, and the device to load it from can also be manipulated.

These flags and names can be set using the `eeprom(1M)` command from the shell, or by using PROM commands from the `ok` prompt after the system has been halted.

The second level program is either a fileystem-specific boot block (when booting from a disk), or `inetboot` or `wanboot` (when booting across the network).

Network Booting

Network booting occurs in two steps: the client first obtains an IP address and any other parameters necessary to permit it to load the second-stage booter. The second-stage booter in turn loads the boot archive from the boot device.

An IP address can be obtained in one of three ways: RARP, DHCP, or manual configuration, depending on the functions available in and configuration of the PROM. Machines of the `sun4u` and `sun4v` kernel architectures have DHCP-capable PROMs.

The boot command syntax for specifying the two methods of network booting are:

```
boot net:rarp
boot net:dhcp
```

The command:

```
boot net
```

without a `rarp` or `dhcp` specifier, invokes the default method for network booting over the network interface for which `net` is an alias.

The sequence of events for network booting using RARP/`bootparams` is described in the following paragraphs. The sequence for DHCP follows the RARP/`bootparams` description.

When booting over the network using RARP/`bootparams`, the PROM begins by broadcasting a reverse ARP request until it receives a reply. When a reply is received, the PROM then broadcasts a TFTP request to fetch the first block of `inetboot`. Subsequent requests will be sent to the server that initially answered the first block request. After loading, `inetboot` will also use reverse ARP to fetch its IP address, then broadcast `bootparams` RPC calls (see `bootparams(4)`) to locate configuration information and its root file system. `inetboot` then loads the boot archive by means of NFS and transfers control to that archive.

When booting over the network using DHCP, the PROM broadcasts the hardware address and kernel architecture and requests an IP address, boot parameters, and network configuration information. After a DHCP server responds and is selected (from among potentially multiple servers), that server sends to the client an IP address and all other information needed to boot the client. After receipt of this information, the client PROM examines the name of the file to be loaded, and will behave in one of two ways, depending on whether the file's name appears to be an HTTP URL. If it does not, the PROM downloads `inetboot`, loads that file into memory, and executes it. `inetboot` loads the boot archive, which takes over the machine and releases `inetboot`. Startup scripts then initiate the DHCP agent (see `dhcpagent(1M)`), which implements further DHCP activities.

If the file to be loaded is an HTTP URL, the PROM will use HTTP to load the referenced file. If the client has been configured with an HMAC SHA-1 key, it will check the integrity of the loaded file before proceeding to execute it. The file is expected to be the `wanboot` binary. The WAN boot process can be configured to use either DHCP or NVRAM properties to discover the install server and router and the proxies needed to connect to it. When `wanboot` begins executing, it determines whether sufficient information is available to it to allow it to proceed. If any necessary information is missing, it will either exit with an appropriate error or bring up a command interpreter and prompt for further configuration information. Once `wanboot` has

obtained the necessary information, it loads the boot loader into memory by means of HTTP. If an encryption key has been installed on the client, `wanboot` will verify the boot loader's signature and its accompanying hash. Presence of an encryption key but no hashing key is an error.

The `wanboot` boot loader can communicate with the client using either HTTP or secure HTTP. If the former, and if the client has been configured with an HMAC SHA-1 key, the boot loader will perform an integrity check of the root file system. Once the root file system has been loaded into memory (and possibly had an integrity check performed), the boot archive is transferred from the server. If provided with a `boot_logger` URL by means of the `wanboot.conf(4)` file, `wanboot` will periodically log its progress.

Not all PROMs are capable of consuming URLs. You can determine whether a client is so capable using the `list-security-keys` OBP command (see `monitor(1M)`).

WAN booting is not currently available on the x86 platform.

The `wanboot` Command Line

When the client program is `wanboot`, it accepts `client-program-args` of the form:

`boot ... -o` *opt1*`[,`*opt2*`[,...]]`

where each option may be an action:

`dhcp`
Require `wanboot` to obtain configuration parameters by means of DHCP.

`prompt`
Cause `wanboot` to enter its command interpreter.

<*cmd*>
One of the interpreter commands listed below.

...or an assignment, using the interpreter's parameter names listed below.

The `wanboot` Command Interpreter

The `wanboot` command interpreter is invoked by supplying a `client-program-args` of "`-o prompt`" when booting. Input consists of single commands or assignments, or a comma-separated list of commands or assignments. The configuration parameters are:

`host-ip`
IP address of the client (in dotted-decimal notation)

`router-ip`
IP address of the default router (in dotted-decimal notation)

`subnet-mask`
subnet mask (in dotted-decimal notation)

`client-id`
DHCP client identifier (a quoted ASCII string or hex ASCII)

`hostname`
hostname to request in DHCP transactions (ASCII)

`http-proxy`
HTTP proxy server specification (IPADDR[:PORT])

The key names are:

`3des`
the triple DES encryption key (48 hex ASCII characters)

`aes`
the AES encryption key (32 hex ASCII characters)

`sha1`
the HMAC SHA-1 signature key (40 hex ASCII characters)

Finally, the URL or the WAN boot CGI is referred to by means of:

`bootserver`
URL of WAN boot's CGI (the equivalent of OBP's `file` parameter)

The interpreter accepts the following commands:

`help`
Print a brief description of the available commands

var=*val*
Assign *val* to *var*, where *var* is one of the configuration parameter names, the key names, or `bootserver`.

var=
Unset parameter *var*.

`list`
List all parameters and their values (key values retrieved by means of OBP are never shown).

`prompt`
Prompt for values for unset parameters. The name of each parameter and its current value (if any) is printed, and the user can accept this value (press Return) or enter a new value.

`go`
Once the user is satisfied that all values have been entered, leave the interpreter and continue booting.

`exit`
Quit the boot interpreter and return to OBP's `ok` prompt.

Any of these assignments or commands can be passed on the command line as part of the `-o` options, subject to the OBP limit of 128 bytes for boot arguments. For example, `-o list,go` would simply list current (default) values of the parameters and then continue booting.

iSCSI Boot

iSCSI boot is supported on both x86 and SPARC.

iSCSI Boot on x86

For iSCSI boot on x86, the host being booted must be equipped with NIC(s) capable of iBFT (iSCSI Boot Firmware Table) or have the mainboard's BIOS be iBFT-capable. iBFT, defined in the Advanced Configuration and Power Interface (ACPI) 3.0b specification, specifies a block of information that contains various parameters that are useful to the iSCSI Boot process.

Firmware implementing iBFT presents an iSCSI disk in the BIOS during startup as a bootable device by establishing the connection to the iSCSI target. The rest of the process of iSCSI booting is the same as booting from a local disk.

To configure the iBFT properly, users need to refer to the documentation from their hardware vendors.

iSCSI Boot on SPARC

iSCSI boot on SPARC is supported with OpenBoot level 4.31 and above, and does not require a specific NIC.

The boot command in OpenBoot takes a series of keywords to identify the destination iSCSI target, following the *keyword=value* format. The complete form of the iSCSI boot command is:

```
boot net:iscsi-target-ip=t-ip,iscsi-target-name=name
     host-ip=h-ip[,router-ip=r-ip]
     [,subnet-mask=m-ip]
     [,iscsi-port=port]
     [,iscsi-lun=lun]
     [,iscsi-partition=partition]
```

The descriptions of the preceding keywords are as follows:

`host-ip`	IP address of booting host.
`router-ip`	IP address of routing gateway.
`subnet-mask`	Subnet mask of `host-ip`.
`iscsi-target-ip`	IP address of iSCSI target storing OS.
`iscsi-target-name`	Name of iSCSI target storing OS.
`iscsi-partition`	Partition containing the bootable root.
`iscsi-port`	IP port of the target.
`iscsi-lun`	LUN to be booted off on target.

The values of `iscsi-target-ip`, `route-ip`, and `subnet-mask` are in standard, IPv4 dotted-decimal format; for example, `255.255.255.0` for `subnet-mask`. IPv6 is not supported in the current OpenBoot implementation.

The value of `iscsi-port`, a decimal number, is in the range of 1 to 65535.

The value of `iscsi-lun` is in the format of a dashed hexadecimal LUN, *ffff-ffff-ffff-ffff*. Please refer to section 5 of RFC 4173 for details. Leading zeroes and trailing dashes can be excluded, thus, `3`, for example, is equivalent to `0003-0000-0000-0000`.

The value of `iscsi-partition` is one ASCII character, used to specify the root partition. Most commonly, it is `a`.

The value of `iscsi-target-name` is in the format of a string, as specified by RFC 3720 and RFC 3722.

Two security keys are added to provide CHAP authentication on the target side. These are:

`chap-user` CHAP name

`chap-password` CHAP secret

Currently these two keys can be set with the command `set-ascii-security-key` at the Open Boot PROM (`ok`) prompt. For example:

```
ok set-ascii-security-key chap-user chap name
ok set-ascii-security-key chap-password chap password
```

Bi-directional authentication is not yet supported. These two variables can be changed only under the Open Boot PROM prompt.

RFC 4173 is supported, to retrieve iSCSI boot information from a DHCP server. The DHCP server must specify the Root Path option for the booting client, after which the client can do an iSCSI boot by means of the simple command:

```
boot net:dhcp
```

Currently the key `boot-device` is used to retrieve the physical boot device path during iSCSI boot. This key is setup during the Solaris installation. A manually modified key value might break iSCSI boot.

Booting from Disk

When booting from disk, the OpenBoot PROM firmware reads the boot blocks from blocks 1 to 15 of the partition specified as the boot device. This standalone booter usually contains a file system-specific reader capable of reading the boot archive.

If the pathname to the standalone is relative (does not begin with a slash), the second level boot will look for the standalone in a platform-dependent search path. This path is guaranteed to contain `/platform/`*platform-name*. Many SPARC platforms next search the

platform-specific path entry `/platform/`*hardware-class-name*. See `filesystem(5)`. If the pathname is absolute, `boot` will use the specified path. The `boot` program then loads the standalone at the appropriate address, and then transfers control.

Once the boot archive has been transferred from the boot device, Solaris can initialize and take over control of the machine. This process is further described in the "Boot Archive Phase," below, and is identical on all platforms.

If the filename is not given on the command line or otherwise specified, for example, by the `boot-file` NVRAM variable, `boot` chooses an appropriate default file to load based on what software is installed on the system and the capabilities of the hardware and firmware.

The path to the kernel must not contain any whitespace.

Booting from ZFS

Booting from ZFS differs from booting from UFS in that, with ZFS, a device specifier identifies a storage pool, not a single root file system. A storage pool can contain multiple bootable datasets (that is, root file systems). Therefore, when booting from ZFS, it is not sufficient to specify a boot device. One must also identify a root file system within the pool that was identified by the boot device. By default, the dataset selected for booting is the one identified by the pool's `bootfs` property. This default selection can be overridden by specifying an alternate bootable dataset with the `-Z` option.

Boot Archive Phase

The boot archive contains a file system image that is mounted using an in-memory disk. The image is self-describing, specifically containing a file system reader in the boot block. This file system reader mounts and opens the RAM disk image, then reads and executes the kernel contained within it. By default, this kernel is in:

```
/platform/‘uname -i‘/kernel/unix
```

If booting from ZFS, the pathnames of both the archive and the kernel file are resolved in the root file system (that is, dataset) selected for booting as described in the previous section.

The initialization of the kernel continues by loading necessary drivers and modules from the in-memory filesystem until I/O can be turned on and the root filesystem mounted. Once the root filesystem is mounted, the in-memory filesystem is no longer needed and is discarded.

OpenBoot PROM `boot` Command Behavior

The OpenBoot `boot` command takes arguments of the following form:

```
ok boot [device-specifier] [arguments]
```

The default `boot` command has no arguments:

```
ok boot
```

If no *device-specifier* is given on the `boot` command line, OpenBoot typically uses the *boot-device* or *diag-device* `NVRAM` variable. If no optional *arguments* are given on the command

line, OpenBoot typically uses the *boot-file* or *diag-file* `NVRAM` variable as default `boot` arguments. (If the system is in diagnostics mode, *diag-device* and *diag-file* are used instead of *boot-device* and *boot-file*).

arguments may include more than one string. All *argument* strings are passed to the secondary booter; they are not interpreted by OpenBoot.

If any *arguments* are specified on the `boot` command line, then neither the *boot-file* nor the *diag-file* `NVRAM` variable is used. The contents of the `NVRAM` variables are not merged with command line arguments. For example, the command:

```
ok boot -s
```

ignores the settings in both *boot-file* and *diag-file*; it interprets the string "`-s`" as *arguments*. `boot` will not use the contents of *boot-file* or *diag-file*.

With older PROMs, the command:

```
ok boot net
```

took no arguments, using instead the settings in *boot-file* or *diag-file* (if set) as the default file name and arguments to pass to boot. In most cases, it is best to allow the `boot` command to choose an appropriate default based upon the system type, system hardware and firmware, and upon what is installed on the root file system. Changing *boot-file* or *diag-file* can generate unexpected results in certain circumstances.

This behavior is found on most OpenBoot 2.x and 3.x based systems. Note that differences may occur on some platforms.

The command:

ok **`boot cdrom`**

...also normally takes no arguments. Accordingly, if *boot-file* is set to the 64-bit kernel filename and you attempt to boot the installation CD or DVD with `boot cdrom`, boot will fail if the installation media contains only a 32-bit kernel.

Because the contents of *boot-file* or *diag-file* can be ignored depending on the form of the `boot` command used, reliance upon *boot-file* should be discouraged for most production systems.

When executing a WAN boot from a local (CD or DVD) copy of wanboot, one must use:

ok **`boot cdrom -F wanboot - install`**

Modern PROMs have enhanced the network boot support package to support the following syntax for arguments to be processed by the package:

[*protocol*,] [*key*=*value*,]*

All arguments are optional and can appear in any order. Commas are required unless the argument is at the end of the list. If specified, an argument takes precedence over any default values, or, if booting using DHCP, over configuration information provided by a DHCP server for those parameters.

protocol, above, specifies the address discovery protocol to be used.

Configuration parameters, listed below, are specified as *key=value* attribute pairs.

`tftp-server`
IP address of the TFTP server

`file`
file to download using TFTP or URL for WAN boot

`host-ip`
IP address of the client (in dotted-decimal notation)

`router-ip`
IP address of the default router

`subnet-mask`
subnet mask (in dotted-decimal notation)

`client-id`
DHCP client identifier

`hostname`
hostname to use in DHCP transactions

`http-proxy`
HTTP proxy server specification (IPADDR[:PORT])

`tftp-retries`
maximum number of TFTP retries

`dhcp-retries`
maximum number of DHCP retries

The list of arguments to be processed by the network boot support package is specified in one of two ways:

- As arguments passed to the package's `open` method, or
- arguments listed in the NVRAM variable `network-boot-arguments`.

Arguments specified in `network-boot-arguments` will be processed only if there are no arguments passed to the package's `open` method.

Argument Values

protocol specifies the address discovery protocol to be used. If present, the possible values are `rarp` or `dhcp`.

If other configuration parameters are specified in the new syntax and style specified by this document, absence of the *protocol* parameter implies manual configuration.

If no other configuration parameters are specified, or if those arguments are specified in the positional parameter syntax currently supported, the absence of the *protocol* parameter causes the network boot support package to use the platform-specific default address discovery protocol.

Manual configuration requires that the client be provided its IP address, the name of the boot file, and the address of the server providing the boot file image. Depending on the network configuration, it might be required that `subnet-mask` and `router-ip` also be specified.

If the *protocol* argument is not specified, the network boot support package uses the platform-specific default address discovery protocol.

`tftp-server` is the IP address (in standard IPv4 dotted-decimal notation) of the TFTP server that provides the file to download if using TFTP.

When using DHCP, the value, if specified, overrides the value of the TFTP server specified in the DHCP response.

The TFTP RRQ is unicast to the server if one is specified as an argument or in the DHCP response. Otherwise, the TFTP RRQ is broadcast.

file specifies the file to be loaded by TFTP from the TFTP server, or the URL if using HTTP. The use of HTTP is triggered if the file name is a URL, that is, the file name starts with `http:` (case-insensitive).

When using RARP and TFTP, the default file name is the ASCII hexadecimal representation of the IP address of the client, as documented in a preceding section of this document.

When using DHCP, this argument, if specified, overrides the name of the boot file specified in the DHCP response.

When using DHCP and TFTP, the default file name is constructed from the root node's `name` property, with commas (,) replaced by periods (.).

When specified on the command line, the filename must not contain slashes (`/`).

The format of URLs is described in RFC 2396. The HTTP server must be specified as an IP address (in standard IPv4 dotted-decimal notation). The optional port number is specified in decimal. If a port is not specified, port 80 (decimal) is implied.

The URL presented must be “safe-encoded”, that is, the package does not apply escape encodings to the URL presented. URLs containing commas must be presented as a quoted string. Quoting URLs is optional otherwise.

`host-ip` specifies the IP address (in standard IPv4 dotted-decimal notation) of the client, the system being booted. If using RARP as the address discovery protocol, specifying this argument makes use of RARP unnecessary.

If DHCP is used, specifying the `host-ip` argument causes the client to follow the steps required of a client with an "Externally Configured Network Address", as specified in RFC 2131.

`router-ip` is the IP address (in standard IPv4 dotted-decimal notation) of a router on a directly connected network. The router will be used as the first hop for communications spanning networks. If this argument is supplied, the router specified here takes precedence over the preferred router specified in the DHCP response.

`subnet-mask` (specified in standard IPv4 dotted-decimal notation) is the subnet mask on the client's network. If the subnet mask is not provided (either by means of this argument or in the DHCP response), the default mask appropriate to the network class (Class A, B, or C) of the address assigned to the booting client will be assumed.

`client-id` specifies the unique identifier for the client. The DHCP client identifier is derived from this value. Client identifiers can be specified as:

- The ASCII hexadecimal representation of the identifier, or
- a quoted string

Thus, `client-id="openboot"` and `client-id=6f70656e626f6f74` both represent a DHCP client identifier of 6F70656E626F6F74.

Identifiers specified on the command line must must not include slash (`/`) or spaces.

The maximum length of the DHCP client identifier is 32 bytes, or 64 characters representing 32 bytes if using the ASCII hexadecimal form. If the latter form is used, the number of characters in the identifier must be an even number. Valid characters are 0-9, a-f, and A-F.

For correct identification of clients, the client identifier must be unique among the client identifiers used on the subnet to which the client is attached. System administrators are responsible for choosing identifiers that meet this requirement.

Specifying a client identifier on a command line takes precedence over any other DHCP mechanism of specifying identifiers.

`hostname` (specified as a string) specifies the hostname to be used in DHCP transactions. The name might or might not be qualified with the local domain name. The maximum length of the hostname is 255 characters.

Note – The `hostname` parameter can be used in service environments that require that the client provide the desired hostname to the DHCP server. Clients provide the desired hostname to the DHCP server, which can then register the hostname and IP address assigned to the client with DNS.

`http-proxy` is specified in the following standard notation for a host:

host `[":"` *port*`]`

...where *host* is specified as an IP ddress (in standard IPv4 dotted-decimal notation) and the optional *port* is specified in decimal. If a port is not specified, port 8080 (decimal) is implied.

`tftp-retries` is the maximum number of retries (specified in decimal) attempted before the TFTP process is determined to have failed. Defaults to using infinite retries.

`dhcp-retries` is the maximum number of retries (specified in decimal) attempted before the DHCP process is determined to have failed. Defaults to of using infinite retries.

x86 Bootstrap Procedure

On x86 based systems, the bootstrapping process consists of two conceptually distinct phases, kernel loading and kernel initialization. Kernel loading is implemented in GRUB (GRand Unified Bootloader) using the BIOS ROM on the system board, and BIOS extensions in ROMs on peripheral boards. The BIOS loads GRUB, starting with the first physical sector from a hard disk, DVD, or CD. If supported by the ROM on the network adapter, the BIOS can also download the `pxegrub` binary from a network boot server. Once GRUB is located, it executes a command in a menu to load the `unix` kernel and a pre-constructed boot archive containing kernel modules and data.

If the device identified by GRUB as the boot device contains a ZFS storage pool, the `menu.lst` file used to create the GRUB menu will be found in the dataset at the root of the pool's dataset hierarchy. This is the dataset with the same name as the pool itself. There is always exactly one such dataset in a pool, and so this dataset is well-suited for pool-wide data such as the `menu.lst` file. After the system is booted, this dataset is mounted at */poolname* in the root file system.

There can be multiple bootable datasets (that is, root file systems) within a pool. By default, the file system in which file name entries in a `menu.lst` file are resolved is the one identified by the pool's `bootfs` property (see `zpool(1M)`). However, a `menu.lst` entry can contain a `bootfs` command, which specifies an alternate dataset in the pool. In this way, the `menu.lst` file can contain entries for multiple root file systems within the pool.

Kernel initialization starts when GRUB finishes loading the boot archive and hands control over to the `unix` binary. At this point, GRUB becomes inactive and no more I/O occurs with the boot device. The Unix operating system initializes, links in the necessary modules from the boot archive and mounts the root file system on the real root device. At this point, the kernel regains storage I/O, mounts additional file systems (see `vfstab(4)`), and starts various operating system services (see `smf(5)`).

Enabling Automatic Rebooting (x86)

The Solaris operating system supports an `smf(5)` property that enables a system to automatically reboot from the current boot device, to recover from conditions such as an out-of-date boot archive.

The service `svc:/system/boot-config:default` contains the boolean property `auto-reboot-safe`, which is set to `false` by default. Setting it to `true` communicates that both the system's BIOS and default GRUB menu entry are set to boot from the current boot device. The value of this property can be changed using `svccfg(1M)` and `svcadm(1M)`. For example, to set `auto-reboot-safe` to enable automatic rebooting, enter a command such as:

```
example# svccfg -s svc:/system/boot-config:default \
      setprop config/auto-reboot-safe = true
```

Most systems are configured for automatic reboot from the current boot device. However, in some instances, automatic rebooting to an unknown operating system might produce undesirable results. For these instances, the `auto-reboot-safe` property allows you to specify the behavior you want.

Failsafe Mode

A requirement of booting from a root filesystem image built into a boot archive then remounting root onto the actual root device is that the contents of the boot archive and the root filesystem must be consistent. Otherwise, the proper operation and integrity of the machine cannot be guaranteed.

The term "consistent" means that all files and modules in the root filesystem are also present in the boot archive and have identical contents. Since the boot strategy requires first reading and mounting the boot archive as the first-stage root image, all unloadable kernel modules and initialization derived from the contents of the boot archive are required to match the real root filesystem. Without such consistency, it is possible that the system could be running with a kernel module or parameter setting applied to the root device before reboot, but not yet updated in the root archive. This inconsistency could result in system instability or data loss.

Once the root filesystem is mounted, and before relinquishing the in-memory filesystem, Solaris performs a consistency verification against the two file systems. If an inconsistency is detected, Solaris suspends the normal boot sequence and falls back to failsafe mode. Correcting this state requires the administrator take one of two steps. The recommended procedure is to reboot to the failsafe archive and rebuild the boot archive. This ensures that a known kernel is booted and functioning for the archive rebuild process. Alternatively, the administrator can elect to clear the inconsistent boot archive service state and continue system bring-up if the inconsistency is such that correct system operation will not be impaired. See `svcadm(1M)`.

If the boot archive service is cleared and system bring-up is continued (the second alternative above), the system may be running with unloadable kernel drivers or other modules that are out-of-date with respect to the root filesystem. As such, correct system operation may be compromised.

To ensure that the boot archive is consistent, the normal system shutdown process, as initiated by `reboot(1M)` and `shutdown(1M)`, checks for and applies updates to the boot archive at the conclusion of the `umountall(1M)` milestone.

An update to any kernel file, driver, module or driver configuration file that needs to be included in the boot archive after the `umountall` service is complete will result in a failed boot archive consistency check during the next boot. To avoid this, it is recommended to always shut down a machine cleanly.

If an update is required to the kernel after completion of the `umountall` service, the administrator may elect to rebuild the archive by invoking:

```
# bootadm update-archive
```

Failsafe Boot Archive

The failsafe archive can be used to boot the machine at any time for maintenance or troubleshooting. The failsafe boot archive is installed on the machine, sourced from the miniroot archive. Booting the failsafe archive causes the machine to boot using the in-memory filesystem as the root device.

SPARC

The SPARC failsafe archive is:

```
/platform/‘uname -i‘/failsafe
```

...and can be booted as follows:

```
ok boot [device-specifier] -F failsafe
```

If a user wishes to boot a failsafe archive from a particular ZFS bootable dataset, this can be done as follows:

```
ok boot [device-specifier] -Z dataset -F failsafe
```

x86

The x86 failsafe archive is:

```
/boot/x86.miniroot-safe
```

...and can be booted by selecting the `Solaris failsafe` item from the GRUB menu.

Options

SPARC

The following SPARC options are supported:

`-a`
The boot program interprets this flag to mean `ask me`, and so it prompts for the name of the standalone. The `'-a'` flag is then passed to the standalone program.

`-D` *default-file*
Explicitly specify the *default-file*. On some systems, `boot` chooses a dynamic default file, used when none is otherwise specified. This option allows the *default-file* to be explicitly set and can be useful when booting `kmdb(1)` since, by default, `kmdb` loads the default-file as exported by the `boot` program.

`-F` *object*
Boot using the named object. The object must be either an ELF executable or bootable object containing a boot block. The primary use is to boot the failsafe or `wanboot` boot archive.

`-L`
List the bootable datasets within a ZFS pool. You can select one of the bootable datasets in the list, after which detailed instructions for booting that dataset are displayed. Boot the selected dataset by following the instructions. This option is supported only when the boot device contains a ZFS storage pool.

`-V`
Display verbose debugging information.

boot-flags
The boot program passes all *boot-flags* to `file`. They are not interpreted by `boot`. See the `kernel(1M)` and `kmdb(1)` manual pages for information about the options available with the default standalone program.

client-program-args
The `boot` program passes all *client-program-args* to *file*. They are not interpreted by `boot`.

file
Name of a standalone program to `boot`. If a filename is not explicitly specified, either on the `boot` command line or in the *boot-file* NVRAM variable, `boot` chooses an appropriate default filename.

OBP names
Specify the open boot prom designations. For example, on Desktop SPARC based systems, the designation `/sbus/esp@0,800000/sd@3,0:a` indicates a SCSI disk (sd) at target 3, lun0 on the SCSI bus, with the esp host adapter plugged into slot 0.

`-Z` *dataset*
Boot from the root file system in the specified ZFS dataset.

x86 The following x86 options are supported:

`-B` *prop=val...*
One or more property-value pairs to be passed to the kernel. Multiple property-value pairs must be separated by a comma. Use of this option is the equivalent of the command: `eeprom` *prop=val*. See `eeprom(1M)` for available properties and valid values.

If the root file system corresponding to this menu entry is a ZFS dataset, the menu entry needs the following option added:

```
-B $ZFS-BOOTFS
```

boot-args
The boot program passes all *boot-args* to `file`. They are not interpreted by `boot`. See `kernel(1M)` and `kmdb(1)` for information about the options available with the kernel.

`/platform/i86pc/kernel/$ISADIR/unix`
Name of the kernel to boot. When using the `kernel$` token, `$ISADIR` expands to `amd64` on 64-bit machines, and a null string on other machines. As a result of this dereferencing, this path expands to the proper kernel for the machine.

x86 Boot Sequence Details

After a PC-compatible machine is turned on, the system firmware in the BIOS ROM executes a power-on self test (POST), runs BIOS extensions in peripheral board ROMs, and invokes software interrupt INT 19h, Bootstrap. The INT 19h handler typically performs the standard PC-compatible boot, which consists of trying to read the first physical sector from the first diskette drive, or, if that fails, from the first hard disk. The processor then jumps to the first byte of the sector image in memory.

x86 Primary Boot

The first sector on a floppy disk contains the master boot block (GRUB `stage1`). The stage 1 is responsible for loading GRUB `stage2`. Now GRUB is fully functional. It reads and executes the menu file `/boot/grub/menu.lst`. A similar sequence occurs for DVD or CD boot, but the master boot block location and contents are dictated by the El Torito specification. The El Torito boot also leads to `strap.com`, which in turn loads `boot.bin`.

The first sector on a hard disk contains the master boot block, which contains the master boot program and the FDISK table, named for the PC program that maintains it. The master boot finds the active partition in the FDISK table, loads its first sector (GRUB `stage1`), and jumps to its first byte in memory. This completes the standard PC-compatible hard disk boot sequence. If GRUB `stage1` is installed on the master boot block (see the `-m` option of `installgrub(1M)`), then `stage2` is loaded directly from the Solaris `FDISK` partition regardless of the active partition.

An x86 FDISK partition for the Solaris software begins with a one-cylinder boot slice, which contains GRUB `stage1` in the first sector, the standard Solaris disk label and volume table of contents (VTOC) in the second and third sectors, and GRUB `stage2` in the fiftieth and subsequent sectors. The area from sector 4 to 49 might contain boot blocks for older versions of Solaris. This makes it possible for multiple Solaris releases on the same FDISK to coexist. When the FDISK partition for the Solaris software is the active partition, the master boot program (`mboot`) reads the partition boot program in the first sector into memory and jumps to it. It in turn reads GRUB `stage2` program into memory and jumps to it. Once the GRUB menu is displayed, the user can choose to boot an operating system on a different partition, a different disk, or possibly from the network.

For network booting, the supported method is Intel's Preboot eXecution Environment (PXE) standard. When booting from the network using PXE, the system or network adapter BIOS uses DHCP to locate a network bootstrap program (`pxegrub`) on a boot server and reads it using Trivial File Transfer Protocol (TFTP). The BIOS executes the `pxegrub` by jumping to its first byte in memory. The `pxegrub` program downloads a menu file and presents the entries to user.

x86 Kernel Startup

The kernel startup process is independent of the kernel loading process. During kernel startup, console I/O goes to the device specified by the `console` property.

When booting from UFS, the root device is specified by the `bootpath` property, and the root file system type is specified by the `fstype` property. These properties should be setup by the Solaris Install/Upgrade process in `/boot/solaris/bootenv.rc` and can be overridden with the `-B` option, described above (see the `eeprom(1M)` man page).

When booting from ZFS, the root device is specified by a boot parameter specified by the `-B $ZFS-BOOTFS` parameter on either the `kernel` or `module` line in the GRUB menu entry. This value (as with all parameters specified by the `-B` option) is passed by GRUB to the kernel.

If the console properties are not present, console I/O defaults to `screen` and `keyboard`. The root device defaults to `ramdisk` and the file system defaults to `ufs`.

Examples

SPARC **EXAMPLE 1** To Boot the Default Kernel In Single-User Interactive Mode

To boot the default kernel in single-user interactive mode, respond to the `ok` prompt with one of the following:

```
boot -as

boot disk3 -as
```

EXAMPLE 2 Network Booting with WAN Boot-Capable PROMs

To illustrate some of the subtle repercussions of various boot command line invocations, assume that the `network-boot-arguments` are set and that `net` is devaliased as shown in the commands below.

In the following command, device arguments in the device alias are processed by the device driver. The network boot support package processes arguments in `network-boot-arguments`.

```
boot net
```

The command below results in no device arguments. The network boot support package processes arguments in `network-boot-arguments`.

```
boot net:
```

The command below results in no device arguments. `rarp` is the only network boot support package argument. `network-boot-arguments` is ignored.

```
boot net:rarp
```

In the command below, the specified device arguments are honored. The network boot support package processes arguments in `network-boot-arguments`.

```
boot net:speed=100,duplex=full
```

EXAMPLE 3 Using `wanboot` with Older PROMs

The command below results in the `wanboot` binary being loaded from DVD or CD, at which time `wanboot` will perform DHCP and then drop into its command interpreter to allow the user to enter keys and any other necessary configuration.

```
boot cdrom -F wanboot -o dhcp,prompt
```

x86 (32–bit) **EXAMPLE 4** To Boot the Default Kernel In 32–bit Single-User Interactive Mode

To boot the default kernel in single-user interactive mode, edit the GRUB kernel command line to read:

```
kernel /platform/i86pc/kernel/unix -as
```

x86 (64–bit Only) **EXAMPLE 5** To Boot the Default Kernel In 64–bit Single-User Interactive Mode

To boot the default kernel in single-user interactive mode, edit the GRUB kernel command line to read:

```
kernel /platform/i86pc/kernel/amd64/unix -as
```

EXAMPLE 6 Switching Between 32-bit and 64-bit Kernels on 64-bit x86 Platform

To be able to boot both 32-bit and 64-bit kernels, add entries for both kernels to `/boot/grub/menu.lst`, and use the `set-menu` subcommand of `bootadm(1M)` to switch. See `bootadm(1M)` for an example of the `bootadm set-menu`.

Files

`/platform/`*platform-name*`/ufsboot`
: Second-level program to boot from a disk, DVD, or CD

`/etc/inittab`
: Table in which the `initdefault` state is specified

`/sbin/init`
: Program that brings the system to the `initdefault` state

64-bit SPARC Only `/platform/`*platform-name*`/kernel/sparcv9/unix`
: Default program to boot system.

x86 Only `/boot`
: Directory containing boot-related files.

`/boot/grub/menu.lst`
: Menu of bootable operating systems displayed by GRUB.

`/platform/i86pc/kernel/unix`
: 32–bit kernel.

64–bit x86 Only `/platform/i86pc/kernel/amd64/unix`
: 64–bit kernel.

See Also `kmdb(1)`, `uname(1)`, `bootadm(1M)`, `eeprom(1M)`, `init(1M)`, `installboot(1M)`, `kernel(1M)`, `monitor(1M)`, `shutdown(1M)`, `svcadm(1M)`, `svccfg(1M)`, `umountall(1M)`, `zpool(1M)`, `uadmin(2)`, `bootparams(4)`, `inittab(4)`, `vfstab(4)`, `wanboot.conf(4)`, `attributes(5)`, `filesystem(5)`, `smf(5)`

RFC 903, *A Reverse Address Resolution Protocol*, `http://www.ietf.org/rfc/rfc903.txt`

RFC 2131, *Dynamic Host Configuration Protocol*, `http://www.ietf.org/rfc/rfc2131.txt`

RFC 2132, *DHCP Options and BOOTP Vendor Extensions*, `http://www.ietf.org/rfc/rfc2132.txt`

RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*, `http://www.ietf.org/rfc/rfc2396.txt`

System Administration Guide: Basic Administration

Sun Hardware Platform Guide

OpenBoot Command Reference Manual

Warnings The `boot` utility is unable to determine which files can be used as bootable programs. If the booting of a file that is not bootable is requested, the `boot` utility loads it and branches to it. What happens after that is unpredictable.

Notes *platform-name* can be found using the `-i` option of `uname(1)`. *hardware-class-name* can be found using the `-m` option of `uname(1)`.

The current release of the Solaris operating system does not support machines running an UltraSPARC-I CPU.

Name bootadm – manage bootability of GRUB-enabled operating system

Synopsis

```
/sbin/bootadm update-archive [-vn] [-R altroot [-p platform]]

/sbin/bootadm list-archive [-vn] [-R altroot [-p platform]]

 x86 only

/sbin/bootadm set-menu [-R altroot [-p platform]] key=value

/sbin/bootadm list-menu [-R altroot [-p platform]]
```

Description The `bootadm` command manages the boot archive and, with x86 boot environments, the GRUB (GRand Unified Bootloader) menu. The `update–archive` option provides a way for user to update the boot archive as a preventative measure or as part of a recovery procedure. The `set-menu` subcommand allows you to switch the `auto-boot` timeout and default boot entry in the GRUB menu.

The `list-menu` subcommand displays the location of the GRUB menu and the current GRUB menu entries. While the typical location of the GRUB menu is `/boot/grub/menu.lst`, depending on the install method used the active GRUB menu might be located somewhere else. Use the `list-menu` subcommand to locate the active GRUB menu. For example, if a system was installed using Live Upgrade, the GRUB menu might not be located in the current boot environment. See the EXAMPLES section for typical output from the `list-menu` option.

Note that OpenBoot PROM (OBP)-based machines, such as SPARC systems, do not use GRUB and have no boot menu manageable by `bootadm`.

The `bootadm` command determines dynamically the options supported by the image to be managed, so that `bootadm` invoked on one platform can be used to manage diskless clients of a different platform type.

Subcommands The `bootadm` command has the following subcommands:

`update-archive`
: Updates current boot archive if required. Applies to both SPARC and x86 platforms.

`list-archive`
: Lists the files and directories to be included in the boot archive. Applies to both SPARC and x86 platforms.

`set-menu`
: Maintain the GRUB menu. The current GRUB menu is `boot/grub/menu.lst`, relative to root. Do not depend on this location, because it is subject to change. Applies to x86 platforms only.

`list-menu`
: Lists the location of the active GRUB menu, as well as the current GRUB menu entries. This includes the autoboot-timeout, the default entry number, and the title of each entry. Applies to x86 platforms only.

Options The `bootadm` command has the following options:

`-v`
In an `update-archive` operation, stale files are displayed on stderr.

`-n`
In an `update-archive` operation, archive content is checked but not updated.

`-p` *platform*
The platform, or machine hardware class, of the client. The platform type can only be specified together with `-R`, and is generally useful only for managing a diskless client where the client is of a different platform class than the server. Platform must be one of `i86pc`, `sun4u`, or `sun4v`.

`-R` *altroot*
Operation is applied to an alternate root path.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

key=value
Possible values are:

`default=`*entrynum*
The item number (for example, 0, 1, or 2) in the GRUB menu designating the operating system to boot when the timer expires.

`timeout=`*seconds*
The number of seconds before the operating system designated by the default item number is booted. If the value is –1, auto boot is disabled.

Examples

EXAMPLE 1 Updating the Current Boot Archive

The following command updates the current boot archive:

```
# bootadm update-archive
```

EXAMPLE 2 Updating the Boot Archive on an Alternate Root

The following command updates the boot archive on an alternate root:

```
# bootadm update-archive -R /a
```

EXAMPLE 3 Listing Installed OS Instances

The following command lists the installed operating system instances in a GRUB menu:

```
# bootadm list-menu

default=0
timeout=10
(0) Solaris10
```

EXAMPLE 3 Listing Installed OS Instances *(Continued)*

```
(1) Solaris10 Failsafe
(2) Linux
```

EXAMPLE 4 Switching Default Boot Entry

The following command refers to the menu displayed in the previous example. The user selects Linux (item 2).

```
# bootadm set-menu default=2
```

EXAMPLE 5 Listing GRUB Menu Entries and Location of GRUB Menu

The following command lists the GRUB menu entries and the location of the GRUB menu:

```
# bootadm list-menu
The location for the active GRUB menu is: /stubboot/boot/grub/menu.lst
default 0
timeout 10
0 Solaris10
1 Solaris10 failsafe
2 Linux
```

EXAMPLE 6 Displaying Location of GRUB Menu

The following command displays the location of the GRUB menu:

```
# bootadm list-menu
The location for the active GRUB menu is: /dev/dsk/c0t1d0s0 (not mounted)
The filesystem type of the menu device is <ufs>
default 2
timeout 10
0 c0t1d0s3
1 c0t1d0s3 failsafe
2 Solaris10
3 Solaris10 failsafe
```

In this example, the active GRUB menu is located on a device which is *not* mounted. To access the GRUB menu, mount the device and access the GRUB menu at <*mountpoint*>`/boot/grub/menu.lst`.

Exit Status

The following exit values are returned:

`0`
: The command completed successfully.

`1`
: The command exited due to an error.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed

See Also boot(1M), installgrub(1M), attributes(5)

Consult the GRUB home page, under:

http://www.gnu.org/

Name bootconfchk – verify the integrity of a network boot configuration file

Synopsis

```
/usr/sbin/bootconfchk [bootconf-file]
```

Description The `bootconfchk` command checks that the file specified is a valid network boot configuration file as described in `wanboot.conf(4)`.

Any discrepancies are reported on standard error.

Exit Status

`0` Successful completion.

`1` An error occurred.

`2` Usage error.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbsup
Interface Stability	Evolving

See Also `wanboot.conf(4)`, `attributes(5)`

Name bsmconv, bsmunconv – enable or disable Solaris Auditing

Synopsis

```
/etc/security/bsmconv [rootdir]...

/etc/security/bsmunconv [rootdir]...
```

Description The `bsmconv` and `bsmunconv` scripts are used to enable or disable the BSM features on a Solaris system. The optional argument *rootdir* is a list of one or more root directories of diskless clients that have already been configured. See `smdiskless(1M)`.

To enable or disable BSM on a diskless client, a server, or a stand-alone system, logon as super-user to the system being converted and use the `bsmconv` or `bsmunconv` commands without any options.

To enable or disable BSM on a diskless client from that client's server, logon to the server as super-user and use `bsmconv`, specifying the root directory of each diskless client you wish to affect. For example, the command:

```
myhost# bsmconv /export/root/client1 /export/root/client2
```

enables BSM on the two machines named `client1` and `client2`. While the command:

```
myhost# bsmconv
```

enables BSM only on the machine called `myhost`. It is no longer necessary to enable BSM on both the server and its diskless clients.

After running `bsmconv` the system can be configured by editing the files in `/etc/security`. Each diskless client has its own copy of configuration files in its root directory. You might want to edit these files before rebooting each client.

Following the completion of either script, the affected system(s) should be rebooted to allow the auditing subsystem to come up properly initialized.

Files The following files are created by `bsmconv`:

`/etc/security/device_maps`	Administrative file defining the mapping of device special files to allocatable device names.
`/etc/security/device_allocate`	Administrative file defining parameters for device allocation.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr
Interface Stability	Obsolete Committed

See Also auditconfig(1M), auditd(1M), audit_startup(1M), audit.log(4), audit_control(4), attributes(5)

See the section on Solaris Auditing in *System Administration Guide: Security Services*.

Notes bsmconv and bsmunconv are not valid in a non-global zone.

These commands are Obsolete and may be removed and replaced with equivalent functionality in a future release of Solaris.

Name bsmrecord – display Solaris audit record formats

Synopsis

```
/usr/sbin/bsmrecord [-d] [ [-a] | [-e string] | [-c class] |
      [-i id] | [-p programname] | [-s systemcall] | [-h]]
```

Description The `bsmrecord` utility displays the event ID, audit class and selection mask, and record format for audit record event types defined in `audit_event(4)`. You can use `bsmrecord` to generate a list of all audit record formats, or to select audit record formats based on event class, event name, generating program name, system call name, or event ID.

There are two output formats. The default format is intended for display in a terminal window; the optional HTML format is intended for viewing with a web browser.

Tokens contained in square brackets (`[ ]`) are optional and might not be present in every record.

Options The following options are supported:

`-a`
List all audit records.

`-c` *class*
List all audit records selected by *class*. *class* is one of the two-character class codes from the file `/etc/security/audit_class`.

`-d`
Debug mode. Display number of audit records that are defined in `audit_event`, the number of classes defined in `audit_class`, any mismatches between the two files, and report which defined events do not have format information available to `bsmrecord`.

`-e` *string*
List all audit records for which the event ID label contains the string *string*. The match is case insensitive.

`-h`
Generate the output in HTML format.

`-i` *id*
List the audit records having the numeric event ID *id*.

`-p` *programname*
List all audit records generated by the program *programname*, for example, audit records generated by a user-space program.

`-s` *systemcall*
List all audit records generated by the system call *systemcall*, for example, audit records generated by a system call.

The `-p` and `-s` options are different names for the same thing and are mutually exclusive. The `-a` option is ignored if any of `-c`, `-e`, `-i`, `-p`, or `-s` are given. Combinations of `-c`, `-e`, `-i`, and either `-p` or `-s` are ANDed together.

Examples

EXAMPLE 1 Displaying an Audit Record with a Specified Event ID

The following example shows how to display the contents of a specified audit record.

```
% bsmrecord -i 6152
  terminal login
  program     /usr/sbin/login      see login(1)
              /usr/dt/bin/dtlogin  See dtlogin
  event ID    6152                 AUE_login
  class       lo                   (0x00001000)
      header
      subject
      [text]                       error message
      return
```

EXAMPLE 2 Displaying an Audit Record with an Event ID Label that Contains a Specified String

The following example shows how to display the contents of a audit record with an event ID label that contains the string `login`.

```
# bsmrecord -e login
terminal login
  program     /usr/sbin/login      see login(1)
              /usr/dt/bin/dtlogin  See dtlogin
  event ID    6152                 AUE_login
  class       lo                   (0x00001000)
      header
      subject
      [text]                       error message
      return

rlogin
  program     /usr/sbin/login      see login(1) - rlogin
  event ID    6155                 AUE_rlogin
  class       lo                   (0x00001000)
      header
      subject
      [text]                       error message
      return
```

Exit Status

`0`
: Successful operation

`non-zero`
: Error

Files

`/etc/security/audit_class`
: Provides the list of valid classes and the associated audit mask.

`/etc/security/audit_event`
Provides the numeric event ID, the literal event name, and the name of the associated system call or program.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr
CSI	Enabled
Interface Stability	Obsolete Uncommitted

See Also `auditconfig(1M)`, `praudit(1M)`, `audit.log(4)`, `audit_class(4)`, `audit_event(4)`, `attributes(5)`

See the section on Solaris Auditing in *System Administration Guide: Security Services*.

Diagnostics If unable to read either of its input files or to write its output file, `bsmrecord` shows the name of the file on which it failed and exits with a non-zero return.

If no options are provided, if an invalid option is provided, or if both `-s` and `-p` are provided, an error message is displayed and `bsmrecord` displays a usage message then exits with a non-zero return.

Notes This command is Obsolete and may be removed and replaced with equivalent functionality in a future release of Solaris.

If `/etc/security/audit_event` has been modified to add user-defined audit events, `bsmrecord` displays the record format as `undefined`.

The audit records displayed by `bsmrecord` are the core of the record that can be produced. Various audit policies and optional tokens, such as those shown below, might also be present.

The following is a list of `praudit(1M)` token names with their descriptions.

`group`
Present if the `group` audit policy is set.

`sensitivity label`
Present when Trusted Extensions is enabled and represents the label of the subject or object with which it is associated. The `mandatory_label` token is noted in the basic audit record where a label is explicitly part of the record.

`sequence`
Present when the `seq` audit policy is set.

`trailer`
Present when the `trail` audit policy is set.

`zone`

The name of the zone generating the record when the `zonename` audit policy is set. The `zonename` token is noted in the basic audit record where a zone name is explicitly part of the record.

Name busstat – report bus-related performance statistics

Synopsis

```
busstat -e device-inst | -h | -l

busstat [-a] [-n]
     [-w  device-inst [,pic0=event,picn=event ]]...
     [-r device-inst]... [interval  [count]]
```

Description `busstat` provides access to the bus-related performance counters in the system. These performance counters allow for the measurement of statistics like hardware clock cycles, bus statistics including DMA and cache coherency transactions on a multiprocessor system. Each bus device that supports these counters can be programmed to count a number of events from a specified list. Each device supports one or more Performance Instrumentation Counters (PIC) that are capable of counting events independently of each other.

Separate events can be selected for each PIC on each instance of these devices. `busstat` summarizes the counts over the last interval seconds, repeating forever. If a count is given, the statistics are repeated count times.

Only root users can program these counters. Non-root users have the option of reading the counters that have been programmed by a root user.

The default value for the *interval* argument is 1 second, and the default *count* is unlimited.

The devices that export these counters are highly platform-dependent and the data may be difficult to interpret without an in-depth understanding of the operation of the components that are being measured and of the system they reside in.

Options The following options are supported:

`-a` — Display absolute counter values. The default is `delta` values.

`-e` *device-inst* — Display the list of events that the specified device supports for each pic.

Specify *device-inst* as device (`name`) followed by an optional instance number. If an instance number is specified, the events for that instance are displayed. If no instance number is specified, the events for the first instance of the specified device are displayed.

`-h` — Print a usage message.

`-l` — List the devices in the system which support performance counters.

`-n` — Do not display a title in the output. The default is to display titles.

-r *device-inst*	Read and display all pic values for the specified device Specify *device-inst* as *device* (`name`) followed by *instance number*, if specifying an instance number of a device whose counters are to be read and displayed. If all instances of this device are to be read, use *device* (`name`) without an instance number. All pic values will be sampled when using the `-r` option.
-w *device-inst* [,pic0=*event*] [,pic*n*=*event*]	Program (write) the specified devices to count the specified events. Write access to the counters is restricted to root users only. Non-root users can use `-r` option. Specify *device-inst* as *device* (`name`) followed by an optional *instance number*. If specifying an instance number of a device to program these events on. If all instances of this device are to be programmed the same, then use *device* without an instance number. Specify an event to be counted for a specified pic by providing a comma separated list of `pic`*n*=*event* values. The `-e` option displays all valid event names for each device. Any devices that are programmed will be sampled every interval seconds and repeated count times. It is recommended that the interval specified is small enough to ensure that counter wraparound will be detected. The rate at which counters wraparound varies from device to device. If a user is programming events using the `-w` option and `busstat` detects that another user has changed the events that are being counted, the tool will terminate as the programmed devices are now being controlled by another user. Only one user can be programming a device instance at any one time. Extra devices can be sampled using the `-r` option. Using multiple instances of the `-w` option on the same command line, with the same *device-inst* specifying a different list of events for the pics will give the effect of multiplexing for that device. `busstat` will

switch between the list of events for that device every interval seconds. Event can be a string representing the event name, or even a number representing the bit pattern to be programmed into the Performance Control Register (PCR). This assumes explicit knowledge of the meaning of the control register bits for a device. The number can be specified in hexadecimal, decimal, or octal, using the usual conventions of `strtol(3C)`.

Exit Status The following exit values are returned:

0 Successful completion.

1 An error occurred.

2 Another user is writing to the same devices.

Examples

SPARC Only **EXAMPLE 1** Programming and monitoring the Address Controller counters

In this example, `ac0` refers to the Address Controller instance 0. The counters are programmed to count Memory Bank stalls on an Ultra Enterprise system at 10 second intervals with the values displayed in absolute form instead of deltas.

```
# busstat -a -w ac0,pic0=mem_bank0_stall,pic1=mem_bank1_stall 10
time  dev   event0            pic0   event1             pic1
10    ac0   mem_bank0_stall   1234   mem_bank1_stall    5678
20    ac0   mem_bank0_stall   5678   mem_bank1_stall   12345
30    ac0   mem_bank0_stall  12345   mem_bank1_stall   56789
...
```

For a complete list of the supported events for a device, use the `-e` option.

EXAMPLE 2 Programming and monitoring the counters on all instances of the Address Controller

In this example, `ac` refers to all `ac` instances. This example programs all instances of the Address Controller counters to `count_clock` cycles and `mem_bank0_rds` at 2 second intervals, 100 times, displaying the values as deltas.

```
# busstat -w ac,pic0=clock_cycles,pic1=mem_bank0_rds 2 100
time  dev     event0          pic0           event1          pic1
2     ac0     clock_cycles    167242902      mem_bank0_rds   3144
2     ac1     clock_cycles    167254476      mem_bank0_rds   1392
4     ac0     clock_cycles    168025190      mem_bank0_rds   40302
4     ac1     clock_cycles    168024056      mem_bank0_rds   40580
...
```

EXAMPLE 3 Monitoring the events being counted

This example monitors the events that are being counted on the sbus1 device, 100 times at 1 second intervals. It suggests that a root user has changed the events that `sbus1` was counting to be `dvma_tlb_misses` and interrupts instead of `pio_cycles`.

```
% busstat -r sbus0 1 100

time    dev     event0              pic0       event1          pic1
1       sbus1   pio_cycles          2321       pio_cycles      2321
2       sbus1   pio_cycles          48         pio_cycles      48
3       sbus1   pio_cycles          49         pio_cycles      49
4       sbus1   pio_cycles          2281       pio_cycles      2281
5       sbus1   dvma_tlb_misses     0          interrupts      0
6       sbus1   dvma_tlb_misses     6          interrupts      2
7       sbus1   dvma_tlb_misses     8          interrupts      11
...
```

EXAMPLE 4 Event Multiplexing

This example programs `ac0` to alternate between counting (clock cycles, `mem_bank0_rds`) and (`addr_pkts`, `data_pkts`) at 2 second intervals while also monitoring what `ac1` is counting :

It shows the expected output of the above `busstat` command. Another root user on the machine has changed the events that this user had programmed and `busstat` has detected this and terminates the command with a message.

```
# busstat -w ac0,pic0=clock_cycles,pic1=mem_bank0_rds \
          -w ac0,pic0=addr_pkts,pic1=data_pkts \
          -r ac1 2

time    dev     event0          pic0            event1          pic1
2       ac0     addr_pkts       12866           data_pkts       17015
2       ac1     rio_pkts        385             rio_pkts        385
4       ac0     clock_cycles    168018914       mem_bank0_rds   2865
4       ac1     rio_pkts        506             rio_pkts        506
6       ac0     addr_pkts       144236          data_pkts       149223
6       ac1     rio_pkts        522             rio_pkts        522
8       ac0     clock_cycles    168021245       mem_bank0_rds   2564
8       ac1     rio_pkts        387             rio_pkts        387
10      ac0     addr_pkts       144292          data_pkts       159645
10      ac1     rio_pkts        506             rio_pkts        506
12      ac0     clock_cycles    168020364       mem_bank0_rds   2665
12      ac1     rio_pkts        522             rio_pkts        522
busstat: events changed (possibly by another busstat).
#
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also iostat(1M), mpstat(1M), vmstat(1M), strtol(3C), attributes(5)

Name cachefsd – CacheFS daemon

Synopsis

```
/usr/lib/fs/cachefs/cachefsd
```

Description The `cachefsd` server implements features of the cache filesystem (CacheFS). It is invoked at boot time and run if the `/` (root) and `/usr` filesystems are being cached. If `/usr` is being cached, `cachefsd` is invoked by `inetd(1M)` from `inetd.conf(4)`. At run time, `cachefsd` is invoked by the `inetd` mechanism in response to an RPC request from a user command such as `mount_cachefs(1M)`.

The `cachefsd` server supports the "disconnected mode" of CacheFS. In this mode, a user can continue to read and, depending on the option selected, write to files in a cached filesystem when the NFS server for the cached files is not available.

The `cachefsd` daemon performs the following functions in support of the CacheFS:

- Implements the connection policy. The daemon determines whether the NFS server backing the cache is connected or disconnected from the cache, or is in transition from the connected or disconnected states.
- Implements "log rolling," wherein the daemon monitors a disconnected NFS server for reconnection. After such a server returns to a connected state, `cachefsd` rolls any local changes to cached files (kept in a log) back to the server.
- Manages "packing," wherein `cachefsd` makes a best effort to ensure that files in a user-specified list are available in the cache in disconnected mode.
- Supports user interfaces by supplying statistics, reporting conflicts between the cache and the back filesystem, and supporting a list of files for packing.

The running of `cachefsd` is required for the disconnected mode of CacheFS.

Options The following options are supported:

`-r` Used for invoking `cachefsd` for the `/` filesystem.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `cachefspack(1M)`, `cfsadmin(1M)`, `mount_cachefs(1M)`, `inetd.conf(4)`, `attributes(5)`

System Administration Guide: Basic Administration

Name cachefslog – Cache File System logging

Synopsis `cachefslog [-f` *logfile* `| -h]` *cachefs_mount_point*

Description The `cachefslog` command displays where CacheFS statistics are being logged. Optionally, it sets where CacheFS statistics are being logged, or it halts logging for a cache specified by *cachefs_mount_point*. The *cachefs_mount_point* argument is a mount point of a cache file system. All file systems cached under the same cache as *cachefs_mount_point* will be logged.

Options The following options are supported. You must be super-user to use the `-f` and `-h` options.

`-f` *logfile* Specify the log file to be used.

`-h` Halt logging.

Operands *cachefs_mount_point* A mount point of a cache file system.

Usage See `largefile(5)` for the description of the behavior of `cachefslog` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Checking the Logging of a directory.

The example below checks if the directory `/home/sam` is being logged:

```
example% cachefslog /home/sam
not logged: /home/sam
```

EXAMPLE 2 Changing the *logfile*.

The example below changes the *logfile* of `/home/sam` to `/var/tmp/samlog`:

```
example# cachefslog -f /var/tmp/samlog /home/sam
/var/tmp/samlog: /home/sam
```

EXAMPLE 3 Verifying the change of a *logfile*.

The example below verifies the change of the previous example:

```
example% cachefslog /home/sam
/var/tmp/samlog: /home/sam
```

EXAMPLE 4 Halting the logging of a directory.

The example below halts logging for the `/home/sam` directory:

```
example# cachefslog -h /home/sam
not logged: /home/sam
```

Exit Status The following exit values are returned:

`0` success

non-zero an error has occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also cachefsstat(1M), cachefswssize(1M), cfsadmin(1M), attributes(5), largefile(5)

Diagnostics `Invalid path` It is illegal to specify a path within a cache file system.

Name cachefspack – pack files and file systems in the cache

Synopsis

```
cachefspack [-h] [-i | -p | -u] [-f packing-list]
     [-U cache-directory] [file]...
```

Description The `cachefspack` utility is used to set up and maintain files in the cache. This utility affords greater control over the cache, ensuring that the specified files are in the cache whenever possible.

`cachefspack` does not pack files when the backfileystem type for the `cachefs` mount is NFSv4. This is because only pass-through support is available for `cachefs` with NFSv4.

Options The following options are supported:

`-f` *packing-list*	Specify a file containing a list of files and directories to be packed. Options within subdirectories and files can also be specified. The format and rules governing *packing-list* are described on the `packingrules(4)` manual page. Directories are packed recursively. Symlinks that match a regular expression on a `LIST` command are followed. Symlinks encountered while recursively processing directories are not followed.
`-h`	Help. Print a brief summary of all the options.
`-i`	View information about the packed files.
`-p`	Pack the file or files specified by `file`. This is the default behavior.
`-u`	Unpack the file or files specified by `file`.
`-U` *cache-directory*	Unpack all files in the specified cache directory.

Operands The following operands are supported:

`file` A path name of a file to be packed or unpacked.

Usage See `largefile(5)` for the description of the behavior of `cachefspack` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples

EXAMPLE 1 Packing a File in the Cache

The following example packs the file `projects` in the cache:

```
% cachefspack -p projects
```

EXAMPLE 2 Packint Files in the Cache

The following example packs the files `projects`, `updates`, and `master_plan` in the cache:

```
% cachefspack -p projects updates master_plan
```

EXAMPLE 3 Unpacking a File From the Cache

The following example unpacks the file `projects` from the cache:

```
% cachefspack -u projects
```

EXAMPLE 4 Unpacking Files From the Cache

The following example unpacks the files `projects`, `updates`, and `master_plan` from the cache:

```
% cachefspack -u projects updates master_plan
```

EXAMPLE 5 Unpacking All Files From in a Cache Directory

The following example unpacks all files in the cache directory `cache1`:

```
% cachefspack -U /cache/cache1
```

EXAMPLE 6 Using a Packing List

The following example illustrates the use of a packing list to specify files to be packed in the cache.

The contents of `lists.pkg` are as follows:

```
IGNORE SCCS BASE /src/junk LIST *.c LIST *.h
```

The following command packs all files in the `/src/junk` directory which have `.c` and `.h` extensions, and do contain the string SCCS in the file's path name:

```
% cachefspack -f lists.pkg
```

Exit Status

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`cfsadmin(1M)`, `mount_cachefs(1M)`, `packingrules(4)`, `attributes(5)`, `largefile(5)`

Name cachefsstat – Cache File System statistics

Synopsis

```
/usr/bin/cachefsstat [-z] [path]...
```

Description The `cachefsstat` command displays statistical information about the cache file system mounted on *path*. The statistical information includes cache hits and misses, consistency checking, and modification operations. If *path* is not specified, all mounted cache file systems are used.

`cachefsstat` can also be used to reinitialize this information (see `-z` option).

The statistical information has the following format:

<*cache hit rate*>
<*consistency checks*>
<*modifies*>

where:

hit rate	The percentage of cache hits over the total number of attempts, followed by the actual numbers of hits and misses.
consistency checks	The number of consistency checks performed, followed by the number that passed, and the number that failed.
modifies	The number of modify operations, including writes, creates, etc.

Options The following option is supported:

`-z` Zero (reinitialize) statistics. Execute `cachefsstat -z` before executing `cachefsstat` again to gather statistics on the cache performance. This option can only be use by the superuser. The statistics printed reflect those just before the statistics are reinitialized.

Usage See `largefile(5)` for the description of the behavior of `cachefsstat` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Using `cachefsstat`

The following example shows the `cachefsstat` command run on file system `/test`:

```
example# cachefsstat /test
  /test
       cache hit rate:      100% (0 hits, 0 misses)
       consistency checks:    0  (0 pass, 0 fail)
       modifies:              0
garbage collection:           0
```

Exit Status The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also cachefslog(1M), cachefswssize(1M), cfsadmin(1M), attributes(5), largefile(5)

Name cachefswssize – determine working set size for cachefs

Synopsis `cachefswssize` *logfile*

Description The `cachefswssize` command displays the workspace size determined from *logfile*. This includes the amount of cache space needed for each filesystem that was mounted under the cache, as well as a total.

Usage See `largefile(5)` for the description of the behavior of `cachefswssize` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** A sample output of `cachefswssize`.

```
example% cachefswssize /var/tmp/samlog
```

```
/home/sam
                              end size:                   10688k
                              high water size:            10704k

/foo
                              end size:                   128k
                              high water size:            128k

/usr/dist
                              end size:                   1472k
                              high water size:            1472k

total for cache
                              initial size:               110960k
                              end size:                   12288k
                              high water size:            12304k
```

Exit Status The following exit values are returned:

`0` success

non-zero an error has occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also cachefslog(1M), cachefsstat(1M), cfsadmin(1M), attributes(5), largefile(5)

Diagnostics

`problems were encountered writing log file` — There were problems encountered when the kernel was writing the logfile. The most common problem is running out of disk space.

`invalid log file` — The logfile is not a valid logfile or was created with a newer version of Solaris than the one where `cachefswssize` is running.

Name captoinfo – convert a termcap description into a terminfo description

Synopsis

```
captoinfo [-1] [-v]... [-V] [-w width] filename...
```

Description `captoinfo` looks in *filename* for `termcap` descriptions. For each one found, an equivalent `terminfo` description is written to standard output, along with any comments found. A description which is expressed as relative to another description (as specified in the `termcap` `tc` = `field`) is reduced to the minimum superset before being displayed.

If no *filename* is given, then the environment variable `TERMCAP` is used for the filename or entry. If `TERMCAP` is a full pathname to a file, only the terminal whose name is specified in the environment variable `TERM` is extracted from that file. If the environment variable `TERMCAP` is not set, then the file `/usr/share/lib/termcap` is read.

Options

–1	Display the fields one to a line. Otherwise, the fields are printed several to a line, with a maximum width of 60 characters.
`-v`	Display tracing information on the standard error as the program runs. Specifying additional `-v` options displays more detailed information.
`-V`	Display the version of the program in use on the standard error and then exit.
`-w` *width*	Change the output to *width* characters.

Files

`/usr/share/lib/terminfo/?/*`	compiled terminal description database
`/usr/share/lib/termcap`	

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `infocmp(1M)`, `curses(3CURSES)`, `terminfo(4)`, `attributes(5)`

Notes `captoinfo` should be used to convert `termcap` entries to `terminfo` entries because the `termcap` database may not be supplied in future releases.

Name catman – create the formatted files for the reference manual

Synopsis

```
/usr/bin/catman [-c] [-n] [-p] [-t] [-w] [-M directory]
     [-T macro-package] [sections]
```

Description The `catman` utility creates the preformatted versions of the on-line manual from the `nroff(1)` or `sgml(5)` input files. This feature allows easy distribution of the preformatted manual pages among a group of associated machines (for example, with `rdist(1)`), since it makes the directories of preformatted manual pages self-contained and independent of the unformatted entries.

`catman` also creates the `windex` database file in the directories specified by the `MANPATH` or the `-M` option. The `windex` database file is a three column list consisting of a keyword, the reference page that the keyword points to, and a line of text that describes the purpose of the utility or interface documented on the reference page. Each keyword is taken from the comma separated list of words on the `NAME` line before the '–' (dash). The reference page that the keyword points to is the first word on the `NAME` line. The text after the – on the `NAME` line is the descriptive text in the third column. The `NAME` line must be immediately preceded by the page heading line created by the `.TH` macro (see `NOTES` for required format).

Each manual page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, `catman` recreates the `windex` database.

If a manual page is a *shadow* page, that is, it sources another manual page for its contents, a symbolic link is made in the `cat`*x* or `fmt`*x* directory to the appropriate preformatted manual page.

Shadow files in an unformatted nroff source file are identified by the first line being of the form `.so man`*x*`/yyy.`*x*.

Shadow files in the SGML sources are identified by the string `SHADOW_PAGE`. The file entity declared in the shadow file identifies the file to be sourced.

Options The following options are supported:

`-c` Create unformatted nroff source files in the appropriate `man` subdirectories from the SGML sources. This option will overwrite any existing file in the `man` directory of the same name as the SGML file.

`-n` Do not create (or recreate) the `windex` database. If the `-n` option is specified, the `windex` database is not created and the `apropos`, `whatis`, `man -f`, and `man -k` commands will fail.

`-p` Print what would be done instead of doing it.

`-t` Create `troff`ed entries in the appropriate `fmt` subdirectories instead of `nroff`ing into the `cat` subdirectories.

`-w` Only create the `windex` database that is used by `whatis(1)` and the `man(1)` `-f` and `-k` options. No manual reformatting is done.

`-M` *directory* — Update manual pages located in the specified *directory*, (`/usr/share/man` by default). If the `-M` option is specified, the directory argument must not contain a ',' (comma), since a comma is used to delineate section numbers. See `man(1)`.

`-T` *macro-package* — Use *macro-package* in place of the standard manual page macros, (`man(5)` by default).

Operands

The following operand is supported:

sections — If there is one parameter not starting with a '–', it is taken to be a space separated list of manual sections to be processed by `catman`. If this operand is specified, only the manual sections in the list will be processed. For example,

```
catman 1 2 3
```

only updates manual sections `1`, `2`, and `3`. If specific sections are not listed, all sections in the `man` directory specified by the environment variable `MANPATH` are processed.

Environment Variables

`TROFF` — The name of the formatter to use when the `-t` flag is given. If not set, `troff(1)` is used.

`MANPATH` — A colon-separated list of directories that are processed by `catman` and `man(1)`. Each directory can be followed by a comma-separated list of sections. If set, its value overrides `/usr/share/man` as the default directory search path, and the `man.cf` file as the default section search path. The `-M` and `-s` flags, in turn, override these values.

Files

`/usr/share/man`	default manual directory location
`/usr/share/man/man*/*.*`	raw nroff input files
`/usr/share/man/sman*/*.*`	raw SGML input files
`/usr/share/man/cat*/*.*`	preformatted `nroff`ed manual pages
`/usr/share/man/fmt*/*.*`	preformatted `troff`ed manual pages
`/usr/share/man/windex`	table of contents and keyword database
`/usr/lib/makewhatis`	command script to make `windex` database
`/usr/share/lib/tmac/an`	default macro package

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdoc

ATTRIBUTE TYPE	ATTRIBUTE VALUE
CSI	Enabled

See Also apropos(1), man(1), nroff(1), rdist(1), rm(1), troff(1), whatis(1), attributes(5), man(5), sgml(5)

Diagnostics

`man?/xxx.? (.so'ed from man?/yyy.?): No such file or directory`
The file outside the parentheses is missing, and is referred to by the file inside them.

`target of .so in man?/xxx.? must be relative to /usr/man`
`catman` only allows references to filenames that are relative to the directory `/usr/man`.

`opendir:man?: No such file or directory`
A harmless warning message indicating that one of the directories `catman` normally looks for is missing.

`*.*: No such file or directory`
A harmless warning message indicating `catman` came across an empty directory.

Warnings If a user, who has previously run `catman` to install the `cat*` directories, upgrades the operating system, the entire `cat*` directory structure should be removed prior to running `catman`. See `rm(1)`.

Do not re-run `catman` to re-build the `whatis` database unless the complete set of `man*` directories is present. `catman` builds this `windex` file based on the `man*` directories.

Notes To generate a valid windex index file, `catman` has certain requirements. Within the individual man page file, `catman` requires two macro lines to have a specific format. These are the `.TH` page heading line and the `.SH NAME` line.

The `.TH` macro requires at least the first three arguments, that is, the filename, section number, and the date. The `.TH` line starts off with the `.TH` macro, followed by a space, the man page filename, a single space, the section number, another single space, and the date. The date should appear in double quotes and is specified as "day month year," with the month always abbreviated to the first three letters (Jan, Feb, Mar, and so forth).

The `.SH NAME` macro, also known as the `NAME` line, must immediately follow the `.TH` line, with nothing in between those lines. No font changes are permitted in the `NAME` line. The `NAME` line is immediately followed by a line containing the man page filename; then shadow page names, if applicable, separated by commas; a dash; and a brief summary statement. These elements should all be on one line; no carriage returns are permitted.

An example of proper coding of these lines is:

```
.TH nismatch 1M "10 Apr 1998"
.SH NAME
nismatch, nisgrep \- utilities for searching NIS+ tables
```

Name cfgadm – configuration administration

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -c function ap_id...

/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -x hardware_function ap_id...

/usr/sbin/cfgadm [-v] [-a] [-s listing_options]
     [-o hardware_options] [-l [ap_id | ap_type]]

/usr/sbin/cfgadm [-v] [-o hardware_options] -t ap_id...

/usr/sbin/cfgadm [-v] [-o hardware_options] -h
     [ap_id | ap_type]
```

Description The `cfgadm` command provides configuration administration operations on dynamically reconfigurable hardware resources. These operations include displaying status, (`-l`), initiating testing, (`-t`), invoking configuration state changes, (`-c`), invoking hardware specific functions, (`-x`), and obtaining configuration administration help messages (`-h`). Configuration administration is performed at *attachment points*, which are places where system software supports dynamic reconfiguration of hardware resources during continued operation of Solaris.

Configuration administration makes a distinction between hardware resources that are physically present in the machine and hardware resources that are configured and visible to Solaris. The nature of configuration administration functions are hardware specific, and are performed by calling hardware specific libraries.

Configuration administration operates on an *attachment point*. Hardware resources located at attachment points can or can not be physically replaceable during system operation, but are dynamically reconfigurable by way of the configuration administration interfaces.

An attachment point defines two unique elements, which are distinct from the hardware resources that exist beyond the attachment point. The two elements of an attachment point are a *receptacle* and an *occupant*. Physical insertion or removal of hardware resources occurs at attachment points and results in a receptacle gaining or losing an occupant. Configuration administration supports the physical insertion and removal operations as well as other configuration administration functions at an attachment point.

Attachment points have associated state and condition information. The configuration administration interfaces provide control for transitioning attachment point states. A receptacle can exist in one of three states: `empty`, `disconnected` or `connected`, while an occupant can exist in one of two states: `configured` or `unconfigured`.

A receptacle can provide the `empty` state, which is the normal state of a receptacle when the attachment point has no occupants. A receptacle can also provide the `disconnected` state if it has the capability of isolating its occupants from normal system access. Typically this state is used for various hardware specific testing prior to bringing the occupant's resources into full

use by the system, or as a step in preparing an occupant for physical removal or reconfiguration. A receptacle in the disconnected state isolates its occupant from the system as much as its hardware allows, but can provide access for testing and setup. A receptacle must provide the `connected` state, which allows normal access to hardware resources contained on any occupants. The connected state is the normal state of a receptacle that contains an occupant and that is not currently undergoing configuration administration operations.

The hardware resources contained on an occupant in the `unconfigured` state are not represented by normal Solaris data structures and are thus not available for use by Solaris. Operations allowed on an unconfigured occupant are limited to configuration administration operations. The hardware resources of an occupant in the `configured` state are represented by normal Solaris data structures and thus some or all of those hardware resources can be in use by Solaris. All occupants provide both the `configured` and `unconfigured` states,

An attachment point can be in one of five conditions: `unknown`, `ok`, `failing`, `failed`, or `unusable`. An attachment point can enter the system in any condition depending upon results of power-on tests and non-volatile record keeping.

An attachment point with an occupant in the `configured` state is in one of four conditions: `unknown`, `ok`, `failing`, or `failed`. If the condition is not `failing` or `failed` an attachment point can change to `failing` during the course of operation if a hardware dependent recoverable error threshold is exceeded. If the condition is not `failed` an attachment point can change to `failed` during operation as a result of an unrecoverable error.

An attachment point with an occupant in the `unconfigured` state can be in any of the defined conditions. The condition of an attachment point with an `unconfigured` occupant can decay from `ok` to `unknown` after a machine dependent time threshold. Initiating a test function changes the attachment point's condition to `ok`, `failing` or `failed` depending on the outcome of the test. An attachment point that does not provide a test function can leave the attachment point in the `unknown` condition. If a test is interrupted, the attachment point's condition can be set to the previous condition, `unknown` or `failed`. An attachment point in the `unknown`, `ok`, `failing`, or `failed` conditions can be re-tested.

An attachment point can exist in the `unusable` condition for a variety of reasons, such as inadequate power or cooling for the receptacle, an occupant that is unidentifiable, unsupported, incorrectly configured, etc. An attachment point in the `unusable` condition can never be used by the system. It typically remains in this condition until the physical cause is remedied.

An attachment point also maintains busy information that indicates when a state change is in progress or the condition is being reevaluated.

Attachment points are referred to using hardware specific identifiers (*ap_id*s) that are related to the type and location of the attachment points in the system device hierarchy. An *ap_id* can not be ambiguous, it must identify a single attachment point. Two types of *ap_id*

specifications are supported: physical and logical. A physical *ap_id* contains a fully specified pathname, while a logical *ap_id* contains a shorthand notation that identifies an attachment point in a more user-friendly way.

For example, an attachment point representing a system's backplane slot number `7` could have a physical *ap_id* of `/devices/central/fhc/sysctrl:slot7` while the logical *ap_id* could be `system:slot7`. Another example, the third receptacle on the second PCI I/O bus on a system could have a logical *ap_id* of `pci2:plug3`.

Attachment points may also be created dynamically. A dynamic attachment point is named relative to a base attachment point which is present in the system. `ap_id`s for dynamic attachment points consist of a base component followed by two colons (`::`) and a dynamic component. The base component is the base attachment point `ap_id`. The dynamic component is hardware specific and generated by the corresponding hardware specific library.

For example, consider a base attachment point, which represents a SCSI HBA, with the physical `ap_id` `/devices/sbus@1f,0/SUNW,fas@e,8800000:scsi` and logical `ap_id` `c0` . A disk attached to this SCSI HBA could be represented by a dynamic attachment point with logical `ap_id` `c0::dsk/c0t0d0` where `c0` is the base component and `dsk/c0t0d0` is the hardware specific dynamic component. Similarly the physical `ap_id` for this dynamic attachment point would be: `/devices/sbus@1f,0/SUNW,fas@e,8800000:scsi::dsk/c0t0d0`

An *ap_type* is a partial form of a logical `ap_id` that can be ambiguous and not specify a particular attachment point. An *ap_type* is a substring of the portion of the logical `ap_id` up to but not including the colon (`:`) separator. For example, an *ap_type* of `pci` would show all attachment points whose logical *ap_id*s begin with `pci`.

The use of *ap_types* is discouraged. The new select sub-option to the `-s` option provides a more general and flexible mechanism for selecting attachment points. See `OPTIONS`.

The `cfgadm` command interacts primarily with hardware dependent functions contained in hardware specific libraries and thus its behavior is hardware dependent.

For each configuration administration operation a service interruption can be required. Should the completion of the function requested require a noticeable service interruption to interactive users, a prompt is output on the standard error output for confirmation on the standard input before the function is started. Confirmation can be overridden using the `-y` or `-n` options to always answer yes or no respectively. Hardware specific options, such as test level, are supplied as sub-options using the `-o` option.

Operations that change the state of the system configuration are audited by the system log daemon `syslogd(1M)`.

The arguments for this command conform to the `getopt(3C)` and `getsubopt(3C)` syntax convention.

Options The following options are supported:

`-a` Specifies that the `-l` option must also list dynamic attachment points.

`-c`*function* Performs the state change *function* on the attachment point specified by *ap_id.*

Specify *function* as `insert`, `remove`, `disconnect`, `connect`, `configure` or `unconfigure`. These functions cause state transitions at the attachment point by calling hardware specific library routines and are defined in the following list.

`insert` Performs operations that allows the user to manually insert an occupant or to activate a hardware supplied mechanism that performs the physical insertion. `insert` can have hardware specific side effects that temporarily suspend activity in portions of the system. In such cases the hardware specific library generates appropriate warning messages and informs the user of any special considerations or procedures unique to that hardware. Various hardware specific errors can cause this function to fail and set the receptacle condition to `unusable`.

`remove` Performs operations that allow the user to manually remove an occupant or to activate a hardware supplied mechanism to perform the physical removal. `remove` can have hardware specific side effects that temporarily suspend activity in portions of the system. In such cases the hardware specific library generates appropriate warning messages and informs the user of any special considerations or procedures unique to that hardware. Various hardware specific errors can cause this function to fail and set the receptacle condition to unusable.

`disconnect` Performs hardware specific operations to put a receptacle in the disconnected state, which can prevent an occupant from operating in a normal fashion through the receptacle.

`connect` Performs hardware specific operations to put the receptacle in the `connected` state, which allows an occupant to operate in a normal fashion through the receptacle.

`configure` Performs hardware specific operations that allow an occupant's hardware resources to be usable by Solaris. Occupants that are configured are part of the system configuration and are available for manipulation by Solaris device manipulation maintenance commands (eg: `psradm`(1M), `mount`(1M), `ifconfig`(1M)).

`unconfigure` Performs hardware specific operations that logically remove an occupant's hardware resources from the system. The occupant must currently be configured and its hardware resources must not be in use by Solaris.

State transition functions can fail due to the condition of the attachment point or other hardware dependent considerations. All state change *function*s in the direction of adding resources, (`insert`, `connect` and `configure`) are passed onto the hardware specific library when the attachment point is in the `ok` or `unknown` condition. All other conditions require the use of the force option to allow these *function*s to be passed on to the hardware specific library. Attachment point condition does not prevent a hardware specific library being called for related to the removal (`remove`, `disconnect` and `unconfigure`), of hardware resources from the system. Hardware specific libraries can reject state change *function*s if the attachment point is in the `unknown` condition.

The condition of an attachment point is not necessarily changed by the state change functions, however errors during state change operations can change the attachment point condition. An attempt to override a condition and force a state change that would otherwise fail can be made by specifying the force option (`-f`). Hardware specific safety and integrity checks can prevent the force option from having any effect.

`-f` Forces the specified action to occur. Typically, this is a hardware dependent override of a safety feature. Forcing a state change operation can allow use of the hardware resources of occupant that is not in the `ok` or `unknown` conditions, at the discretion of any hardware dependent safety checks.

`-h` [*ap_id* | *ap_type* . . .]

Prints out the help message text. If *ap_id* or *ap_type* is specified, the help routine of the hardware specific library for the attachment point indicated by the argument is called.

`-l` [*ap_id* | *ap_type* . . .]

Lists the state and condition of attachment points specified. Attachment points can be filtered by using the `-s` option and `select` sub-option. Invoking `cfgadm` without one of the action options is equivalent to `-l` without an argument. The format of the list display is controlled by the `-v` and `-s` options. When the `-a` option is specified attachment points are dynamically expanded.

`-n`

Suppress any interactive confirmation and assume that the answer is *no*. If neither `-n` or `-y` is specified, interactive confirmation is obtained through the standard error output and the standard input. If either of these standard channels does not correspond to a terminal (as determined by `isatty(3C)`) then the `-n` option is assumed.

`-o`*hardware_options*

Supplies hardware specific options to the main command option. The format and content of the hardware option string is completely hardware specific. The option string *hardware_options* conforms to the `getsubopt(3C)` syntax convention.

`-s`*listing_options*

Supplies listing options to the list (`-l`) command. *listing_options* conforms to the `getsubopt(3C)` syntax convention. The sub-options are used to specify the attachment point selection criteria (`select`=*select_string*), the type of matching desired (`match`=*match_type*), order of listing (`sort`=*field_spec*), the data that is displayed (`cols`=*field_spec* and `cols2`=*field_spec*), the column delimiter (`delim`=*string*) and whether to suppress column headings (`noheadings`).

When the `select` sub-option is specified, only attachment points which match the specified criteria will be listed. The `select` sub-option has the following syntax:

```
cfgadm -s select=attr1(value1):attr2(value2)...
```

where an *attr* is one of `ap_id`, `class` or `type`. `ap_id` refers to the logical *ap_id* field, `class` refers to attachment point class and `type` refers to the type field. *value1*, *value2*, etc. are the corresponding values to be matched. The type of match can be specified by the `match` sub-option as follows:

```
cfgadm -s match=match_type,select=attr1(value1)...
```

where *match_type* can be either `exact` or `partial`. The default value is `exact`.

Arguments to the `select` sub-option can be quoted to protect them from the shell.

A *field_spec* is one or more *data-field*s concatenated using colon (`:`), as in *data-field*:*data-field*:*data-field*. A *data-field* is one of `ap_id`, `physid`, `r_state`, `o_state`, `condition`, `type`, `busy`, `status_time`, `status_time_p`, `class`, and `info`. The `ap_id` field output is the logical name for the attachment point, while the `physid` field contains the physical name. The `r_state` field can be `empty`, `disconnected` or `connected`. The `o_state` field can be `configured` or `unconfigured`. The `busy` field can be either `y` if the attachment point is busy, or `n` if it is not. The `type` and `info` fields are hardware specific. The `status_time` field provides the time at which either the `r_state`, `o_state`, or condition of the attachment point last changed. The `status_time_p` field is a parsable version of the `status_time` field. If an attachment point has an associated class, the `class` field lists the class name. If an attachment point does not have an associated class, the `class` field lists `none`.

The order of the fields in *field_spec* is significant: For the `sort` sub-option, the first field given is the primary sort key. For the `cols` and `cols2` sub-options, the fields are printed in the order requested. The order of sorting on a *data-field* can be reversed by placing a minus (–) before the *data-field* name within the *field_sec* for the `sort` sub-option. The default value for `sort` is `ap_id`. The defaults values for `cols` and `cols2` depend on whether the `-v` option is given: Without it `cols` is `ap_id:r_state:o_state:condition` and `cols2` is not set. With `-v` `cols` is `ap_id:r_state:o_state:condition:info` and `cols2` is `status_time:type:busy:physid:`. The default value for `delim` is a single space. The value of `delim` can be a string of arbitrary length. The delimiter cannot include comma (`,`) character, see `getsubopt(3C)`. These listing options can be used to create parsable output. See `NOTES`.

`-t`

Performs a test of one or more attachment points. The test function is used to re-evaluate the condition of the attachment point. Without a test level specifier in *hardware_options*, the fastest test that identifies hard faults is used.

More comprehensive tests are hardware specific and are selected using the *hardware_options*.

The results of the test is used to update the condition of the specified occupant to either `ok` if no faults are found, `failing` if recoverable faults are found or `failed` if any unrecoverable faults are found.

If a test is interrupted, the attachment point's condition can be restored to its previous value or set to `unknown` if no errors were found or `failing` if only recoverable errors were found or to `failed` if any unrecoverable errors were found. The attachment point should only be set to `ok` upon normal completion of testing with no errors.

`-v` Executes in verbose mode. For the `-c`, `-t` and `-x` options outputs a message giving the results of each attempted operation. Outputs detailed help information for the `-h` option. Outputs verbose information for each attachment point for the `-l` option.

`-x`*hardware_function* Performs hardware specific functions. Private hardware specific functions can change the state of a receptacle or occupant. Attachment point conditions can change as the result of errors encountered during private hardware specific functions. The format and content of the *hardware_function* string is completely hardware specific. The option string *hardware_function* conforms to the `getsubopt(3C)` syntax convention.

`-y` Suppresses any interactive confirmation and assume that the answer is `yes`.

Usage

The required privileges to use this command are hardware dependent. Typically, a default system configuration restricts all but the list option to the superuser.

Examples

EXAMPLE 1 Listing Attachment Points in the Device Tree

The following example lists all attachment points except dynamic attachment points.

```
example# cfgadm

  Ap_Id         Type        Receptacle      Occupant       Cond
system:slot0    cpu/mem     connected       configured     ok
system:slot1    sbus-upa    connected       configured     ok
system:slot2    cpu/mem     connected       configured     ok
system:slot3    unknown     connected       unconfigured   unknown
system:slot4    dual-sbus   connected       configured     failing
system:slot5    cpu/mem     connected       configured     ok
system:slot6    unknown     disconnected    unconfigured   unusable
```

EXAMPLE 1 Listing Attachment Points in the Device Tree *(Continued)*

```
system:slot7    unknown     empty           unconfigured   ok
c0              scsi-bus    connected       configured     unknown
c1              scsi-bus    connected       configured     unknown
```

EXAMPLE 2 Listing All Configurable Hardware Information

The following example lists all current configurable hardware information, including those represented by dynamic attachment points:

```
example# cfgadm -al

  Ap_Id            Type         Receptacle      Occupant        Cond
system:slot0       cpu/mem      connected       configured      ok
system:slot1       sbus-upa     connected       configured      ok
system:slot2       cpu/mem      connected       configured      ok
system:slot3       unknown      connected       unconfigured    unknown
system:slot4       dual-sbus    connected       configured      failing
system:slot5       cpu/mem      connected       configured      ok
system:slot6       unknown      disconnected    unconfigured    unusable
system:slot7       unknown      empty           unconfigured    ok
c0                 scsi-bus     connected       configured      unknown
c0::dsk/c0t14d0    disk         connected       configured      unknown
c0::dsk/c0t11d0    disk         connected       configured      unknown
c0::dsk/c0t8d0     disk         connected       configured      unknown
c0::rmt/0          tape         connected       configured      unknown
c1                 scsi-bus     connected       configured      unknown
```

EXAMPLE 3 Listing Selectively, Based on Attachment Point Attributes

The following example lists all attachment points whose class begins with `scsi`, `ap_id` begins with `c` and `type` field begins with `scsi`. The argument to the `-s` option is quoted to protect it from the shell.

```
example# cfgadm -s "match=partial,select=class(scsi):ap_id(c):type(scsi)"

Ap_Id          Type          Receptacle      Occupant          Cond
 c0          scsi-bus      connected       configured        unknown
 c1          scsi-bus      connected       configured        unknown
```

EXAMPLE 4 Listing Current Configurable Hardware Information in Verbose Mode

The following example lists current configurable hardware information for *ap-type* `system` in verbose mode:

```
example# cfgadm -v -l system
Ap_Id                     Receptacle Occupant   Condition Information
When         Type      Busy     Phys_Id
```

EXAMPLE 4 Listing Current Configurable Hardware Information in Verbose Mode *(Continued)*

```
system:slot1              connected  configured ok
Apr  4 23:50 sbus-upa  n        /devices/central/fhc/sysctrl:slot1
system:slot3              connected  configured ok        non-detachable
Apr 17 11:20 cpu/mem   n        /devices/central/fhc/sysctrl:slot3
system:slot5              connected  configured ok
Apr  4 23:50 cpu/mem   n        /devices/central/fhc/sysctrl:slot5
system:slot7              connected  configured ok
Apr  4 23:50 dual-sbus n        /devices/central/fhc/sysctrl:slot7
```

The `When` column represents the `status_time` field.

EXAMPLE 5 Testing Two Occupants Using the Hardware Specific `Extended` Test

The following example tests two occupants using the hardware specific `extended` test:

```
example# cfgadm -v -o extended -t system:slot3 system:slot5
Testing attachment point system:slot3 ...  ok
Testing attachment point system:slot5 ...  ok
```

EXAMPLE 6 Configuring an Occupant Using the Force Option

The following example configures an occupant in the `failing` state to the system using the force option:

```
example# cfgadm -f -c configure system:slot3
```

EXAMPLE 7 Unconfiguring an Occupant From the System

The following example unconfigures an occupant from the system:

```
example# cfgadm -c unconfigure system:slot4
```

EXAMPLE 8 Configuring an Occupant at an Attachment Point

The following example configures an occupant:

```
example# cfgadm -c configure c0::dsk/c0t0d0
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `cfgadm`: `LC_TIME`, `LC_MESSAGES`, `NLSPATH` and `TZ`.

`LC_MESSAGES` Determines how `cfgadm` displays column headings and error messages. Listing output data is not affected by the setting of this variable.

`LC_TIME` Determines how `cfgadm` displays human readable status changed time (`status_time`).

`TZ` Specifies the timezone used when converting the status changed time. This applies to both the human readable (`status_time`) and parsable

(`status_time_p`) formats.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

`2` Configuration administration not supported on specified target.

`3` Usage error.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `cfgadm_fp(1M)`, `cfgadm_ib(1M)`, `cfgadm_pci(1M)`,`cfgadm_sbd(1M)`, `cfgadm_scsi(1M)`, `cfgadm_usb(1M)`, `ifconfig(1M)`, `mount(1M)`, `prtdiag(1M)`, `psradm(1M)`, `syslogd(1M)`, `config_admin(3CFGADM)`, `getopt(3C)`, `getsubopt(3C)`, `isatty(3C)`, `attributes(5)`, `environ(5)`

Diagnostics Diagnostic messages appear on the standard error output. Other than options and usage errors, the following are diagnostic messages produced by this utility:

`cfgadm: Configuration administration not supported on`*ap_id*

`cfgadm: No library found for` *ap_id*

`cfgadm:` *ap_id*`is ambiguous`

`cfgadm:` *operation*`: Insufficient privileges`

`cfgadm: Attachment point is busy, try again`

`cfgadm: No attachment points with specified attributes found`

`cfgadm: System is busy, try again`

`cfgadm:` *operation*`: Operation requires a service interruption`

`cfgadm:` *operation*`: Data error:` *error_text*

`cfgadm:` *operation*`: Hardware specific failure:` *error_text*

See `config_admin(3CFGADM)` for additional details regarding error messages.

Notes Hardware resources enter the unconfigured pool in a hardware specific manner. This can occur at various times such as: system initialization or as a result of an unconfigure operation. An occupant that is in the `unconfigured` state is not available for use by the system until specific intervention occurs. This intervention can be manifested as an operator initiated command or it can be by way of an automatic configuring mechanism.

The listing option of the `cfgadm` command can be used to provide parsable input for another command, for example within a shell script. For parsable output, the `-s` option must be used to select the fields required. The `-s` option can also be used to suppress the column headings. The following fields always produce parsable output: `ap_id`, `physid`, `r_state`, `o_state`, `condition`, `busy` `status_time_p`, `class`, and `type`. Parsable output never has white-space characters embedded in the field value.

The following shell script fragment finds the first good `unconfigured` occupant of type `CPU`.

```
found=
cfgadm -l -s "noheadings,cols=ap_id:r_state:condition:type" | \
while read ap_id r_state cond type
do
     if [ "$r_state" = unconfigured -a "$cond" = ok -a "$type" = CPU ]
     then
          if [ -z "$found" ]
          then
               found=$ap_id
          fi
     fi
done
if [ -n "$found" ]
then
         echo "Found CPU $found"
fi
```

The format of the parsable time field (`status_time_p`) is *YYYYMMDDhhmmss*, giving the year, month, day, hour, minute and second in a form suitable for string comparison.

Reference should be made to the hardware specific documentation for details of System Configuration Administration support.

Name cfgadm_ac – EXX00 memory system administration

Synopsis

```
/usr/sbin/cfgadm [-c configure] [-f]
     [-o disable-at-boot | enable-at-boot ] ac#:bank# ...

/usr/sbin/cfgadm [-c unconfigure]
     [-o disable-at-bootp | enable-at-boot ] ac#:bank# ...

/usr/sbin/cfgadm [-v]
     [-o quick | normal | extended, [max_errors=#] ] -t ac#:bank#...

/usr/sbin/cfgadm -x relocate-test ac#:bank# ...

/usr/sbin/cfgadm [-l] -o disable-at-boot | enable-at-boot ac#:bank# ...
```

Description The ac hardware specific library `/usr/platform/sun4u/lib/cfgadm/cfgadm_ac.so.1` provides the functionality for configuring and unconfiguring memory banks on `E6X00`, `E5X00`, `E4X00` and `E3X00` systems as part of the Dynamic Reconfiguration of CPU/Memory boards using `cfgadm_sysctrl(1M)`.

Memory banks appear as attachment points in the device tree. For each CPU/Memory board, two attachment points are published, one for each bank on the board: `bank0` and `bank1`. If the bank is unpopulated, the receptacle state is empty. If the bank is populated, the receptacle state is connected. The receptacle state of a memory bank can never be disconnected. The occupant state of a connected memory bank can be configured or unconfigured. If the occupant state is configured, the memory is in use by Solaris, if unconfigured it is not.

Options Refer to `cfgadm`(1M) for complete descriptions of the command options.

The following options are supported:

`-c` configure | unconfigure

Change the occupant state. The `configure` argument ensures that the memory is initialized and adds the memory to the Solaris memory pool. The `unconfigure` argument removes the memory from use by Solaris. When a CPU/Memory board is to be removed from a system, both banks of memory must be unconfigured.

`cfgadm` refuses the `configure` operation if the memory on the board is marked `disabled-at-boot` (see `info` field), unless either the `-f` (force) option or the enable at boot flag, (`-o enable-at-boot`), is given. The configure operation takes a short time proportional to the size of memory that must be initialized.

`cfgadm` refuses the `unconfigure` operation if there is not enough uncommitted memory in the system (`VM viability` error) or if the bank to be unconfigured has memory that can't be removed (`non-relocatable`

pages error). The presence of non-relocatable pages is indicated by the word permanent in the info listing field. Removing memory from use by Solaris may take a significant time due to factors such as system load and how much paging to secondary storage is required. The unconfigure operation can be cancelled at any time and the memory returned to the fully configured state by interrupting the command invocation with a signal. The unconfigure operation self-cancels if no memory can be removed within a timeout period. The default timeout period of 60 seconds can be changed using the -o timeout=# option, with a value of 0 disabling the timeout.

`-f` Force option. Use this option to override the block on configuring a memory bank marked as disabled at boot in the non-volatile disabled-memory-list variable. See *Platform Notes:Sun Enterprise 6x00/5x00/4x00/3x00 Systems*

`-l` List option. This option is supported as described in cfgadm(1M).

The type field is always *memory*.

The *info* field has the following information for empty banks:

```
slot# empty
```

The slot# indicates the system slot into which the CPU/Memory board is inserted. For example, if this were slot11 the attachment point for use with cfgadm to manipulate the associated board would be sysctrl0:slot11. The info field has the following information for connected banks:

```
slot# sizeMb|sizeGb [(sizeMb|sizeGb used)] base 0x###
      [interleaved #-way] [disabled at boot] [permanent]
```

The size of the bank is given in Mb or Gb as appropriate. If the memory is less than completely used, the used size is reported. The physical base address is given in hexadecimal. If the memory bank is interleaved with some other bank, the interleave factor is reported. If the memory on the board is disabled at boot using the non-volatile disabled-memory-list

	variable, this is reported. If the bank has memory that cannot be removed this is reported as permanent.
`-o` disable-at-boot | enable-at-boot	These options allow the state of the non-volatile disabled-memory-list variable to be modified. These options can be used in conjunction with the issuing of a `-c` option or with the explicit or implied listing command, -l, if no command is required. Use of `-o enable-at-boot` with the `configure` command to override the block on configuring memory on a board in the disabled memory list.
`-o` extended | normal | quick	Use with the `-t` option to specify test level. The `normal` test level ensures that each memory cell stores both a `0` and a `1`, and checks that all cells are separately addressable. The `quick` test level only does the `0`s and `1`s test, and typically misses address line problems. The `extended` test uses patterns to test for adjacent cell interference problems. The default test level is `normal`. See `-t` option.
`-o` max_errors=#	Use with the `-t` option to specify the maximum number of allowed errors. If not specified, a default of `32` is assumed.
`-o` timeout=#	Use with the `unconfigure` command to set the self-cancelling timeout. The default value is `60` and the unit is seconds. A value of `0` means no timeout.
`-t`	Test an unconfigured bank of memory. Specify the test level using the `-o quick` | `normal` | `extended` option. `cfgadm` exits with a `0` (success) if the *test* was able to run on the memory bank. The result of the test is available in the condition for the attachment point.
`-v`	Verbose option. Use this option in combination with the `-t` option to display detailed progress and results of tests.
`-x` relocate-test	For all pages of memory in use on the specified memory bank, a relocation operation as used in the unconfigure command is attempted. The success of this operation does not guarantee that the bank can be unconfigured. Failure indicates that it probably cannot be unconfigured. This option is for test purposes only.

Operands The following operand is supported:

ac#:bank# The attachment points for memory banks are published by instances of the address controller (ac) driver (*ac#*). One instance of the ac driver is created for each system board, but only those instances associated with CPU/Memory boards publish the two bank attachment points, bank0 and bank1.

This form conforms to the logical `ap_id` specification given in `cfgadm`(1M). The corresponding physical `ap_id`s are listed in the `FILES` section.

The ac driver instance numbering has no relation to the slot number for the corresponding board. The full physical attachment point identifier has the slot number incorporated into it as twice the slot number in hexadecimal directly following the `fhc@` part.

Files

/devices/fhc@*,f8800000/ac@0,1000000:bank?	attachment points
/usr/platform/sun4u/lib/cfgadm/cfgadm_ac.so.1	hardware specific library file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkvm.u

See Also `cfgadm(1M)`, `cfgadm_sysctrl(1M)`, `config_admin(3CFGADM)`, `attributes(5)`

Sun Enterprise 6x00, 5x00, 4x00 and 3x00 Systems Dynamic Reconfiguration User's Guide

Platform Notes:Sun Enterprise 6x00/5x00/4x00/3x00 Systems

Notes Refer to the *Sun Enterprise 6x00, 5x00, 4x00 and 3x00 Systems Dynamic Reconfiguration User's Guide* for additional details regarding dynamic reconfiguration of EXX00 system CPU/Memory boards.

Name cfgadm_cardbus – cardbus hardware specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v]
     [-o hardware_options] -c function ap_id [ap_id]

/usr/sbin/cfgadm [-f] [-y | -n] [-v]
     [-o hardware_options] -x hardware_function ap_id
     [ap_id]

/usr/sbin/cfgadm [-v] [-s listing_options]
     [-o hardware_options] [-l [ap_id | ap_type]]

/usr/sbin/cfgadm [-v] [-o harware_options] -t ap_id [ap_id]

/usr/sbin/cfgadm [-v] [-o hardware_function] -h
     [ap_id| ap_type]
```

Description The CardBus slots in Solaris are hot plug capable. This capability is supported by the PCI hardware specific library `/usr/lib/cfgadm/pci.so.1` through the `cfgadm` command (see `cfgadm(1M)`).

The hot plug administrative models between CardBus, PCI, CompactPCI, and PCI Express operate the same fashion. Please refer to `cfgadm_pci(1M)` for the usage information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl

See Also `cfgadm(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`

System Administration Guide: Basic Administration

Name cfgadm_fp – driver specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm  [-f] [-n | -y] [-v] [-o hardware_options]
-c function ap_id [ap_id]

/usr/sbin/cfgadm  [-v] [-a] [-s listing_options] [-o hardware_options]
[-l [ap_id]]

/usr/sbin/cfgadm  [-v] [-o hardware_options] -h [ap_id]
```

Description The `fp` port driver plug-in `/usr/lib/cfgadm/fp.so.1` provides the functionality for Fibre Channel Fabric device node management through `cfgadm`(1M). `cfgadm` operates on attachment points. Attachment points are locations in the system where hardware resources can be dynamically reconfigured. Refer to `cfgadm`(1M) for additional details on attachment points.

For Fibre Channel Fabric device node management, each `fp` port node is represented by an attachment point in the device tree. In addition, each Fibre Channel device is represented by a dynamic attachment point. Attachment points are named through `ap_ids`. Two types of `ap_ids` are defined: logical and physical. The physical `ap_id` is based on the physical pathname. The logical `ap_id` is a shorter, more user-friendly name. For `fp` port nodes, the logical `ap_id` is the corresponding disk controller number. For example, `c0` is a typical logical `ap_id`.

Fibre Channel devices are named with a port World Wide Name (WWN). If a disk device is connected to controller `c0`, its `ap_id` can be:

```
c0::50020f2300006077
```

where `50020f2300006077` identifies the port WWN of a specific Fibre Channel device.

Each device on the Fibre Channel private loop port, Fabric port or public loop port is probed and made available to Solaris by default. Devices connected to the Fibre Channel Fabric port or public loop port can be made unavailable to Solaris by initiating an application or an end user operation. The operation is similar to the hot unplugging of devices by way of management user interfaces. Applications or users can use the `/usr/lib/cfgadm/fp.so.1` library to enable `libcfgadm` to provide interfaces to accomplish this task.

The list of currently connected Fabric devices is generated in the form of the attachment point.

A simple listing of attachment points in the system includes attachment points at `fp` port nodes but not Fibre Channel devices. The following example uses the `-a` flag to the list option (`-l`) to list Fibre Channel devices:

```
# cfgadm -l
Ap_Id                Type         Receptacle   Occupant     Condition
c0                   fc-fabric    connected    configured   unknown
c1                   fc-private   connected    configured   unknown
c2                   fc           connected    unconfigured unknown
sysctrl0:slot0       cpu/mem      connected    configured   ok
sysctrl0:slot1       sbus-upa     connected    configured   ok
```

The following example lists Fibre Channel devices connected to `fp` ports.

```
# cfgadm -al
Ap_Id                Type         Receptacle   Occupant     Condition
c0                   fc-fabric    connected    configured   unknown
c0::50020f2300006077 disk         connected    configured   unknown
c0::50020f23000063a9 disk         connected    configured   unknown
c0::50020f2300005f24 disk         connected    configured   unknown
c0::50020f2300006107 disk         connected    configured   unknown
c1                   fc-private   connected    configured   unknown
c1::220000203708b69c disk         connected    configured   unknown
c1::220000203708ba7d disk         connected    configured   unknown
c1::220000203708b8d4 disk         connected    configured   unknown
c1::220000203708b9b2 disk         connected    configured   unknown
c2                   fc           connected    unconfigured unknown
sysctrl0:slot0       cpu/mem      connected    configured   ok
sysctrl0:slot1       sbus-upa     connected    configured   ok
```

In this example, the `fc-fabric` type of `ap_id` `c0` indicates that the `fp` port is connected to Fabric. For an `fp` port with Fabric related type such as fc-fabric and fc-public, device node creation happens by default at the boot time and can be managed by the `cfgadm` configure and unconfigure operations. The `fc-private` type of `ap_id` `c1` indicates that `fp` port is connected to private-loop and device node creation happens by default as well. The `fc` type of `ap_id` `c2` indicates that nothing is attached to `fp` port `c2`. The `Type` field of a Fibre Channel device ap_id shows the SCSI device type of `LUN` `0` in the device.

A Fibre Channel device with multiple FCP SCSI LUNs is configured into Solaris and each `FCP SCSI LUN` is available as a Solaris device. Suppose that ap_ids `c0::50020f2300006077` and `c0::50020f23000063a9` represent Fibre Channel devices with multiple `FCP SCSI LUN`s.

The following example shows how to list ap_ids with `FCP SCSI LUN` information:

```
# cfgadm -al -o show_SCSI_LUN
Ap_Id                  Type         Receptacle   Occupant     Condition
c0                     fc-fabric    connected    configured   unknown
c0::50020f2300006077,0 disk         connected    configured   unknown
c0::50020f2300006077,1 disk         connected    configured   unknown
c0::50020f2300006077,2 disk         connected    configured   unknown
c0::50020f2300006077,3 disk         connected    configured   unknown
c0::50020f23000063a9,0 disk         connected    configured   unknown
c0::50020f23000063a9,1 disk         connected    configured   unknown
c0::50020f23000063a9,2 disk         connected    configured   unknown
c0::50020f23000063a9,3 disk         connected    configured   unknown
c0::50020f2300005f24,0 disk         connected    unconfigured unknown
c0::50020f2300005f24,1 disk         connected    unconfigured unknown
c0::50020f2300006107,0 disk         connected    unconfigured unknown
c0::50020f2300006107,1 disk         connected    unconfigured unknown
c1                     fc-private   connected    configured   unknown
```

```
c1::220000203708b69c,0 disk         connected    configured   unknown
c1::220000203708ba7d,0 disk         connected    configured   unknown
c1::220000203708b8d4,0 disk         connected    configured   unknown
c1::220000203708b9b2,0 disk         connected    configured   unknown
c2                     fc           connected    unconfigured unknown
```

In this example, the `ap_id` `c0::50020f2300006077,0` identifies the `FCP SCSI LUN` `0` of the Fibre Channel device which is represented by port `WWN` `50020f2300006077`. The Fibre Channel device is reported to have 4 `FCP SCSI LUN`s and they are all configured. `4 FCP SCSI LUN` level ap_ids associated with port `WWN` `50020f2300006077` are listed. The listing also displays `FCP SCSI LUN`s for unconfigured Fibre Channel devices. The Fibre Channel device represented by `c0::50020f2300005f24` is reported to have two `FCP SCSI LUN`s. The configure operation on `c0::50020f2300005f24` creates two Solaris devices. The `Type` field of `FCP SCSI LUN` level ap_ids show the `SCSI` device type of each `LUN`. When a Fibre Channel device has different device type `LUN`s, the `Type` field reflects that.

The receptacle and occupant state for attachment points at the `fp` port have the following meanings:

configured	One or more devices configured on the `fp` port
connected	`fp` port active
disconnected	`fp` port quiesced (IO activity is suspended)
empty	Not applicable
unconfigured	No devices configured on the `fp` port

The state for individual Fibre Channel devices on an `fp` port:

configured	Device is configured into Solaris and is available for use
connected	`fp` port to which the device is connected to is active
disconnected	`fp` port to which the device is attached is quiesced
unconfigured	Device is available to be configured

The `condition` field for attachment points at the `fp` port has the following meanings:

failed	An error condition has prevented the `fp` port from being able to detect the presence or type of a Fibre Channel connection.

The `condition` field for individual Fibre Channel devices on an `fp` port has the following meanings:

failed	An error is encountered while probing a device on Fabric.
failing	A device was configured on a host and its state as seen by Solaris appears to be normal (i.e., online) but it is either not

currently present or visible in the fabric or its presence could not be verified due to an error condition on the local port through which the device was configured.

unusable — A device has been configured on the host, but is currently offline or failed.

The unknown `condition` indicates that probing a device on Fabric completed without an error and the device state within Solaris host is normal if the device was configured previously. The internal condition of the device cannot be guaranteed.

Options `cfgadm` defines several types of operations in addition to listing (`-l`). These operations include invoking configuration state changes and obtaining configuration administration help messages (`-h`).

The following options are supported:

`-c` *function*

The following generic commands are defined for the `fp`-transport-specific library:

For Fibre Channel device attachment points on the `fc-fabric` type `fp` port attachment point, the following configuration state change operations are supported:

`configure` — Configure a connected Fibre Channel Fabric device to a host. When a Fibre Channel device is listed as an unknown type in the output of the list operation the device might not be configurable. No attempt is made to configure devices with unknown types. The force option (`-f`) can be used to force the `fp` port driver plug-in to make an attempt to configure any devices. Any errors in the process are reported. By default, each FCP SCSI LUN that is discovered on a Fibre channel Fabric device is configured. However, FCP SCSI LUNs that are specified in the "pwwn-lun-blacklist" property in the `fp.conf` file will remain unconfigured. The FCP SCSI LUN level listing reflects the state of such FCP SCSI LUNs. They stay in the "unconfigured" state after reboot or Solaris Dynamic Reconfiguration on the controller that they are connected through. Refer to `fp(7d)` for additional details on the "pwwn-lun-blacklist" property.

`unconfigure` — Unconfigure a Fibre Channel Fabric device from a host. This device stays unconfigured until the next reboot or Solaris Dynamic Reconfiguration on the controller that the device is connected, at which time all fabric devices are automatically enumerated. The default behavior may be changed through the use of the "manual_configuration_only" property in the `fp.conf` file. If the property is set, the device remains unconfigured after reboot. Refer to `fp(7d)` for additional details on the "manual_configuration_only" property.

For Fibre Channel private loop devices, the `configure` command returns success without doing any operation. The `unconfigure` command is not supported on the private loop devices. The private loop devices are configured by Solaris Fibre Channel drivers by default and are not managed through end user– or application-initiated operations. The "pwwn-lun-blacklist" property in the `fp.conf` file is applied to the private loop device in the same way it works on a Fabric device.

`-f`

Force the `configure` change state operation to occur irrespective of the `condition` or `type`. Refer to the above description of the `configure` change state operation.

`-h` *ap_id*

Obtain `fp`—transport-specific help. Specify any `fp` attachment point.

`-o` *hardware_options*

The following hardware options are supported.

show_SCSI_LUN	Lists ap_ids associated with each FCP SCSI LUN for discovered Fibre Channel devices when specified with the `list` option `-al`. Refer to the previously mentioned description and example of FCP SCSI LUN level listing. Device node creation is not supported on the FCP SCSI LUN level. See `NOTES`.

All Fibre Channel devices are available to Solaris by default. Enabling only a subset of Fabric devices available to Solaris by default can be accomplished by setting the property "manual_configuration_only" in `/kernel/drv/fp.conf` file. When "manual_configuration_only" in `fp.conf` is set, all Fabric devices are not available to Solaris unless an application or an end user had previously requested the device be configured into Solaris. The `configure` state-change command makes the device available to Solaris. After a successful `configure` operation on a Fabric device, the associated links are added to the `/dev` namespace. The `unconfigure` state-change command makes a device unavailable to Solaris.

When a Fibre Channel Fabric device is configured successfully to a host using the `-c configure` operation, its physical `ap_id` is stored in a repository. When a Fibre Channel Fabric device is unconfigured using the `-c unconfigure` operation, its physical `ap_id` is deleted from the same repository. All fabric devices are automatically enumerated by default and the repository is used only if the `fp.conf` "manual_configuration_only" property is set. Refer to `fp(7d)` for additional details on the "manual_configuration_only" property.

You can specify the following commands with the `-c` option to control the update behavior of the repository:

`force_update` For `configure`, the attachment point is unconditionally added to the repository; for `unconfigure`, the attachment point is unconditionally deleted.

`no_update` No update is made to the repository regardless of the operation.

These options should not be used for normal `configure` and `unconfigure` operations. See `WARNINGS`.

When a Fibre Channel device has multiple `FCP SCSI LUN`s configured and any Solaris device associated with its `FCP SCSI LUN` is in the unusable condition, the whole Fibre Channel device is reported as unusable. The following option with the `-c` unconfigure command removes only Solaris devices with the unusable condition for a Fibre Channel device.

`unusable_SCSI_LUN` For `unconfigure` operation, any offlined device nodes for a target device is removed.

`-s` *listing_options*
Refer to `cfgadm(1M)` for usage information.

`-t` *ap_id*
No test commands are available at present.

`-x` *hardware_function*
No hardware specific functions are available at present.

All other options have the same meaning as defined in the `cfgadm(1M)` man page.

Examples

EXAMPLE 1 Unconfiguring a Disk

The following command unconfigures a disk:

```
# cfgadm -c unconfigure c0::210000203708b606
```

EXAMPLE 2 Unconfigure all the Configured Disks under Single Attachment Point

The following command unconfigures all configured disks under the attachment point `c0`.

```
# cfgadm -c unconfigure c0
```

EXAMPLE 3 Configuring a Disk

The following command configures a disk:

```
# cfgadm -c configure c0::210000203708b606
```

EXAMPLE 4 Configure all the Unconfigured Disks under Single Attachment Point

The following command configures all unconfigured disks under the attachment point `c0`.

```
# cfgadm -c configure c0
```

EXAMPLE 5 Removing the Fibre Channel Fabric Device Attachment Point from Repository

The following command unconditionally removes the fibre channel fabric device attachment point from the Fabric device repository.

```
# cfgadm -c unconfigure -o force_update c0::210000203708b606
```

EXAMPLE 6 Removing Offlined Solaris Device Nodes for a Target Device

The following command removes offlined Solaris device nodes for a target device:

```
# cfgadm -c unconfigure -o unusable_SCSI_LUN c0::210000203708b606
```

Files

`/usr/lib/cfgadm/fp.so.1`
Hardware-specific library for Fibre Channel Fabric device node management.

`/etc/cfg/fp/fabric_WWN_map`
Repository of physical `ap_id`s of Fabric devices currently configured. It is used only to reconfigure those Fabric devices at boot time. This repository is only used when the "manual_configuration_only" `/kernel/drv/fp.conf` file is set.

`/etc/rcS.d/fdevattach`
Reconfigures Fabric device(s) of which physical `ap_id` is listed in `/etc/cfg/fp/fabric_WWN_map` on boot time.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcfpl, SUNWcfplx

See Also `svcs(1)`, `cfgadm(1M)`, `svcadm(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`, `smf(5)`, `fp(7d)`

Warnings Do not use hardware-specific options for the repository update under normal `configure`/`unconfigure` operations. The hardware-specific options are expected to be used when the node creation of a Fabric device fails at boot time and the error condition is considered to be permanent. The `unconfigure` command with `force_update` hardware-specific option unconditionally removes the attachment point of a failing Fabric device from the repository.

Notes For devices with unknown or no SCSI device type (for example, a Fibre Channel Host Bus Adapter), the `configure` operation might not be applicable.

The `configure` and `unconfigure` commands operate on the Fibre Channel device level which is represented by port WWN ap_id. If a Fibre Channel device has multiple FCP SCSI LUNs configured, the `configure` command on the associated port WWN ap_id results in creating a Solaris device for each `FCP SCSI LUN` unless it is specified in the "pwwn-lun-blacklist"

property in the `fp.conf` file. The `unconfigure` command removes all Solaris devices associated with the port WWN ap_id. The `FCP SCSI LUN` level ap_id is not valid for the `configure` and `unconfigure` commands.

The deprecated `show_FCP_dev` option has been replaced by the new `show_SCSI_LUN` option, and the deprecated `unusable_FCP_dev` option has been replaced by the new `unusable_SCSI_LUN` option.

The `cfgadm_fp` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/device/fc-fabric:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

No administrative actions on this service are required for Fabric device configuration once this service is started on boot time.

Name cfgadm_ib – InfiniBand hardware specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm -f [-y | -n] [-v] -c function ap_id...

/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -x hardware_function ap_id...

/usr/sbin/cfgadm -v [-a] [-s listing_option] [-] [ap_id | ap_type...]

/usr/sbin/cfgadm -v -h [ap_id]...
```

Description The InfiniBand hardware specific library `/usr/lib/cfgadm/ib.so.1` provides the functionality for administering its fabric through the `cfgadm`(1M) utility. `cfgadm` operates on attachment points. See `cfgadm`(1M).

An InfiniBand (IB) device is enumerated by the IB nexus driver, `ib`(7D), based on the services from the IB Device Manager (IBDM).

The IB nexus driver creates and initializes five types of child device nodes:

- IB Port devices
- IB HCA service (HCA_SVC) devices
- IB Virtual Physical Point of Attachment (VPPA) devices
- I/O Controller (IOC)
- IB Pseudo devices

See `ib`(7D) for details on enumeration of IB Port, IB VPPA, and IB HCA_SVC devices. For additional information on IBDM, see `ibdm`(7D). See `ib`(4) for details on IB Pseudo devices.

For IB administration, two types of static attachment point are created for the fabric administration as seen by the given host. There is one static attachment point `ib` and all IB devices (either an IOC, Port, VPPA, HCA_SVC, or a Pseudo device) in the fabric are represented as dynamic attachment points based off of it. There is another static attachment point for each Host Channel Adapter (HCA) in the host based on its node Globally Unique Identifier (GUID) value.

Attachment points are named through ap_ids. There are two types of ap_ids: logical and physical. The physical ap_id is based on the physical path name. For the IB fabric it is `/devices/ib:fabric`. The logical ap_id is a shorter, and has a more user friendly name.

The static ap_id for the IB fabric is `ib`. The IB devices are dynamic attachment points and have no physical ap_id. The logical ap_id of an IOC contains its GUID, `ib::`*IOC-GUID*. An example of an IOC ap_id is `ib::80020123456789a`. The logical ap_id of a Pseudo device, see ib(4) for details, is of the format `ib::`*driver_name*`,`*unit-address*. An example of a pseudo ap_id would be `ib::sdp,0` where "`sdp`" is the driver name and "`0`" is its *unit-address* property. The logical ap_id of Port, VPPA and HCA_SVC device contains its Partition Key (*P_Key*), *Port GUID* / *Node GUID* and a communication service-name. The format of ap_id is as below:

Port device
`ib::`*PORT_GUID*`,0,`*service-name*

VPPA device
`ib::`*PORT_GUID*`,`*P_Key*`,`*service-name*

HCA_SVC device
`ib::`*HCA_GUID*`,0,`*servicename*

The Partition Key (*P_Key*) is `0` for Port and HCA_SVC devices. The *P_Key* helps determine the partition to which this port belongs for a VPPA device node. A port might have more than one *P_Key*. An example of a VPPA device logical ap_id point is `ib::80245678,ffff,ipib`. The *port-GUID* is `80245678`, the *P_Key* is `0xffff`, and the service name is `ipib`. The service-name information is obtained from the file `/kernel/drv/ib.conf` which contains service-name strings. The HCA's logical ap_id contains its node GUID value, hca:HCA-GUID. An example is `hca:21346543210a987`.

A listing of the IB attachment points includes information on all IB devices (IOC, VPPA, HCA_SVC, Pseudo, and Port devices seen by the IBDM and the IB nexus driver) in the fabric even if they are not seen by the host and configured for use.

The following shows a listing of five IB devices (two IOC, one VPPA, one Port, one HCA_SVC) and one HCA:

```
example# cfgadm -al
Ap_Id                     Type       Receptacle   Occupant      Condition
hca:21346543210a987       IB-HCA     connected    configured    ok
ib                        IB-FABRIC  connected    configured    ok
ib::80020123456789a       IB-IOC     connected    configured    ok
ib::802abc9876543         IB-IOC     connected    unconfigured  unknown
ib::80245678,ffff,ipib    IB-VPPA    connected    configured    ok
ib::12245678,0,nfs        IB-PORT    connected    configured    ok
ib::21346543,0,hnfs       IB-HCA_SVC connected    configured    ok
ib::sdp,0                 IB-PSEUDO  connected    configured    ok
```

The ap_id `ib::802abc9876543` shows an IOC device that is not yet configured by the host for use or had been previously offlined by an explicit

```
cfgadm -c unconfigure
```

operation. The distinction was made by the information displayed under the `Condition` column. The IB device with a zero P_Key and HCA GUID is a HCA_SVC device. Refer to `cfgadm(1M)` for more information regarding listing attachment points.

The receptacle state for attachment points have the following meanings:

connected
For an IOC/VPPA/Port/Pseudo/HCA_SVC device, `connected` implies that it has been seen by the host. The device might not have been configured for use by Solaris.

For a HCA attachment point, `connected` implies that it has been configured and is in use.

All IB ap_ids are always shown as connected.

The occupant state for attachment points have the following meanings:

configured
The IB device, and the HCA ap_id, are configured and usable by Solaris.

unconfigured
The IB device at the ap_id was explicitly offlined using `cfgadm -c unconfigure`, was not successfully configured. This could be because it wasn not successfully configuref for use with Solaris (no driver, or a device problem), or because it was never configured for use by the IB nexus driver.

The `unconfigured` operation is not supported for the HCA attachment point. The IB static apid, `ib`, is shown unconfigured if the system has no IB hardware.

The attachment point conditions are:

failed
Not used.

failing
Not used.

ok
Normal state. Ready for use.

unknown
This state is only valid for IB device that have been probed by IBDM but not yet configured for use by Solaris. It is also shown for devices that have been explicitly offlined by a `cfgadm -c unconfigure` operation. This condition does not apply to a HCA attachment point.

unusable
Not used.

Options The following options are supported:

`-c` *function*
The IB hardware specific library supports two generic commands (*function*s). These commands are not supported on the static attachment points (that is, the HCA ap_ids and the IB static `ib` ap_id).

The following generic commands are supported:

configure
Configure the IB device to be used by Solaris.

unconfigure
Unconfigure the IB device. If successful, `cfgadm` reports the `condition` of this `ap_id` as `unknown`.

`-f`
Not supported.

`-h` *ap_id*
Obtain IB specific help for an IB attachment point.

`-l`
List the state and condition of IB attachment points. The `-l` option works as described in `cfgadm(1M)`.

When paired with the `-a` option, displays the dynamic attachment points as well (IOC, VPPA, Port, Pseudo, and HCA_SVC devices).

When paired with `-v` option, displays verbose data about the ap_ids. For an IOC, the `Info` field in the

```
cfgadm -avl
```

output displays the following information: VendorID, IOCDeviceID, DeviceVersion, SubsystemVendorID, SubsystemID, Class, Subclass, Protocol, ProtocolVersion and IDString from the IOCControllerProfile. If the ID string isn't provided then nothing is displayed in its place. These fields are defined in the InfiniBand Specification Volume 1 (http://www.infinibandta.org).

For a VPPA, Port, or HCA_SVC device the `Info` field in the `cfgadm -lav` display shows the service name information to which this device is bound. If no such information exists, nothing is displayed.

For a Pseudo device `cfgadm -alv` displays the driver name and its `unit-address` information. For a HCA the verbose listing displays the VendorID, ProductID of the HCA, number of ports it has, and the PortGUID value of its ports. See `EXAMPLES`.

`-o` *hardware_option*
This option is not currently defined.

`-s` *listing_option*
Attachment points of class ib can be listed by using the select sub-option. Refer to the `cfgadm(1M)` man page for more information.

`-x` *hardware_function*
Perform a hardware specific function. Note that the *name* can not be more than 4 characters long.

The following hardware specific functions are supported:

`add_service -ocomm=[port|vppa|hca_svc],service=`*name*
This hardware specific function is supported on the static `IB` attachment point. It can be used to add a new service to `/kernel/drv/ib.conf` file and to update the `ib(7D)` driver.

You must use the `service=`*name* option to indicate the new service to be added. You must use the option `comm=[port|vppa|hca_svc]` option to add the `name` service to either `port-svc-list` or to the `hca-svc-list` in the `/kernel/drv/ib.conf` file. See `EXAMPLES`.

`delete_service -ocomm=[port|vppa|hca_svc],service=`*name*
This hardware specific function is supported on the static `IB` attachment point only. It can be used to delete an existing service from the `/kernel/drv/ib.conf` file and also from the ib(7D) driver's data base. You must use the `service=name` option to indicate which service to delete. You must use the `comm=[port|vppa|hca_svc]` option to delete this service from the `port-svc-list`, `vppa-svc-list`, or `vppa-svc-list` of the `/kernel/drv/ib.conf` file. See `EXAMPLES`.

`list_clients`
Supported on `HCA` attachment points. Displays all the kernel IB clients using this `HCA`. It also displays the respective ap_ids of these kernel IB clients and if they have opened an alternate `HCA` device. See `EXAMPLES`.

.

If a given kernel IB client does not have a valid ap_id then a `-` is displayed in that column.

`list_services`
This hardware specific function is supported on the static `IB` attachment point only. It lists all the Port and VPPA services as read from the `/kernel/drv/ib.conf` file. See `EXAMPLES`.

`unconfig_clients`
This hardware specific function is supported on the static HCA attachment point only. It can be used to unconfigure all IB kernel clients of this given HCA. Only IB kernel clients that do not have an alternate HCA are unconfigured. See `EXAMPLES`.

`update_ioc_config`
This hardware specific function is supported on static ib attachment point and the IOC attachment points. For the `ib` APID, this function updates properties of all the IOC device nodes. For the IOC APID, this function updates the properties of specified IOC device node. This command updates the `port-list`, `port-entries`, `service-id`, and `service-name` IOC node properties .

See ib(7D).

`update_pkey_tbls`
Supported on the static `ib` attachment point. Updates the `PKEY` information inside `IBTL`. `IBTL` re-reads the `P_Key` tables for all the ports on each `HCA` present on the host.

See ibtl(7D).

Examples

EXAMPLE 1 Listing the State and Condition of IB Devices

The following command lists the state and condition of IB devices on the system. It only shows the static attachment points.

EXAMPLE 1 Listing the State and Condition of IB Devices *(Continued)*

```
example# cfgadm
hca:21346543210a987        IB-HCA      connected    configured   ok
ib                         IB-FABRIC   connected    configured   ok
```

The `-a` option lists all attachment points. The following example uses the `-a` option and lists all attachment points:

```
example# cfgadm -a
hca:21346543210a987          IB-HCA      connected    configured   ok
ib                           IB-FABRIC   connected    configured   ok
ib::80020123456789a          IB-IOC      connected    unconfigured ok
ib::80245678,ffff,ipib       IB-VPPA     connected    configured   ok
ib::21346543,0,hnfs          IB-HCA_SVC  connected    configured   ok
ib::12245678,0,nfs           IB-PORT     connected    configured   ok
ib::sdp,0                    IB-PSEUDO   connected    configured   ok
```

EXAMPLE 2 Listing the Verbose Status of a IB VPPA Device

The following command lists the verbose status of a IB VPPA device:

```
example# cfgadm -alv ib::80245678,ffff,ipib
Ap_Id                  Receptacle Occupant   Condition Information
When         Type     Busy  Phys_Id
ib::80245678,ffff,ipib    connected    configured   ok        ipib
unavailable  IB-VPPA  n     /devices/ib:fabric::80245678,ffff,ipib
```

A verbose listing of an IOC shows additional information. The following command shows a verbose listing:

```
example# cfgadm -alv ib::80020123456789a
Ap_Id      Receptacle   Occupant     Condition Information
When       Type     Busy  Phys_Id
ib::80020123456789a     connected    configured   ok         VID: 0xeaea
DEVID: 0xeaea VER: 0x5 SUBSYS_VID: 0x0 SUBSYS_ID: 0x0 CLASS: 0xffff
SUBCLASS: 0xff PROTO: 0xff PROTOVER: 0x1 ID_STRING: Sample Host Adapter
unavailable IB-IOC   n     /devices/ib:fabric::80020123456789a
```

A verbose listing of a Pseudo device shows:

```
example# cfgadm -alv ib::sdp,0
Ap_Id                 Receptacle  Occupant   Condition Information
When       Type   Busy  Phys_Id
ib::sdp,0         connected   configured   ok       Driver = "sd
p" Unit-address = "0"
unavailable  IB-PSEUDO   n  /devices/ib:fabric::sdp,0
```

A verbose listing of a HCA shows:

EXAMPLE 2 Listing the Verbose Status of a IB VPPA Device *(Continued)*

```
example# cfgadm -alv hca:21346543210a987
Ap_Id               Receptacle   Occupant     Condition Information
When       Type    Busy  Phys_Id
hca:21346543210a987  connected    configured   ok         VID: 0x15b3,
PID: 0x5a44, #ports: 0x2, port1 GUID: 0x80245678, port2 GUID: 0x80245679
unavailable  IB-HCA     n  /devices/ib:21346543210a987
```

You can obtain more user-friendly output if you specify these following `cfgadm` class and field selection options: `-s "select=class(ib),cols=ap_id:info"`

The following command displays only IB ap_ids. The output only includes the `ap_id` and `Information` fields.

```
# cfgadm -al -s "cols=ap_id:info"  ib::80245678,ffff,ipib
Ap_Id                               Information
ib::80245678,ffff,ipib              ipib
```

EXAMPLE 3 Unconfiguring an Existing IB IOC

The following command unconfigures the IB IOC attached to `ib::80020123456789a`, then displays the status of the `ap_id`:

```
# cfgadm -c unconfigure ib::80020123456789a
Unconfigure the device: /devices/ib:fabric::80020123456789a
This operation will suspend activity on the IB device
Continue (yes/no)?
```

Enter: `y`

```
IB device unconfigured successfully.
# cfgadm -al ib::80020123456789a
Ap_Id                  Type      Receptacle  Occupant     Condition
ib::80020123456789     IB-IOC    connected   unconfigured unknown
#
```

The condition unknown implies that the device node doesn't exist anymore and this IB device's existence is known only to the IB Device Manager.

EXAMPLE 4 Configuring an IB IOC

The following series of commands configures an IB device attached to `ib::80020123456789a`:

```
# cfgadm -yc configure ib::80020123456789a
# cfgadm -al ib::80020123456789a
Ap_Id                  Type        Receptacle   Occupant     Condition
ib::80020123456789a    IB-IOC      connected    configured   ok
```

EXAMPLE 5 Listing All Kernel IB Clients of a HCA

The following command lists all kernel IB clients of an HCA attached to `hca:21346543210a987`:

```
# cfgadm -x list_clients hca:21346543210a987
Attachment Point       Clients                Alternate HCA
ib::80020123456789a    ioc1                   Yes
ib::80245678,ffff,ipib ipib                   No
ib::21346543,0,hnfs    hnfs                   No
-                      ibdm                   No
-                      ibmf                   No
```

EXAMPLE 6 Adding a Port Service

The following command adds a new Port service called `srp`:

```
 # cfgadm -o comm=port,service=srp -x add_service ib
```

EXAMPLE 7 Deleting a VPPA Service

The following command deletes the `ibd` VPPA service `ibd`:

```
# cfgadm -o comm=vppa,service=ipib -x delete_service ib
```

EXAMPLE 8 Listing Port, VPPA, HCA_SVC Services

The following command lists all Port, VPPA, and HCA_SVC services:

```
# cfgadm -x list_services ib
Port communication services:
         srp

VPPA communication services:
         ipib
         nfs

HCA_SVC communication services:
         hnfs
```

EXAMPLE 9 Reprobing IOC Devices

The following command reprobes all IOC device nodes.

```
# cfgadm -x update_ioc_config ib
This operation can update properties of IOC devices.
Continue (yes/no)?

Enter: y

#
```

EXAMPLE 10 Unconfiguring All Kernel Clients of a HCA

The following command unconfigures all kernel clients of a HCA

```
# cfgadm -x unconfig_clients hca:21346543
 This operation will unconfigure clients of this HCA.
 Continue (yes/no)?

 Enter: y
```

Files

`/usr/lib/cfgadm/ib.so.1`
Hardware-specific library for generic InfiniBand device administration

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl

See Also

`cfgadm(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `ib(4)`, `attributes(5)`, `ib(7D)`, `ibdm(7D)`, `ibtl(7D)`

InfiniBand Specification Volume 1 (http://www.infinibandta.org)

Notes

Apart from the listing (`cfgadm -l` or `cfgadm -x list_clients`), only the superuser can execute any functions on an attachment point.

Name cfgadm_pci – PCI, CompactPCI, and PCI Express Hotplug hardware specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v]
     [-o hardware_options] -c function ap_id [ap_id]

/usr/sbin/cfgadm [-f] [-y | -n] [-v]
     [-o hardware_options] -x hardware_function ap_id
     [ap_id]

/usr/sbin/cfgadm [-v] [-s listing_options]
     [-o hardware_options] [-l [ap_id | ap_type]]

/usr/sbin/cfgadm [-v] [-o harware_options] -t ap_id [ap_id]

/usr/sbin/cfgadm [-v] [-o hardware_function] -h
     [ap_id| ap_type]
```

Description The PCI hardware specific library, `/usr/lib/cfgadm/pci.so.1`, provides the support for hotplugging PCI and CompactPCI adapter cards into the respective hotpluggable slots in a system that is hotplug capable, through the `cfgadm` command (see `cfgadm`(1M)). This library does not include support for PCI Express Hotplug or Standard PCI Hotplug adapter cards, which are provided by a different library (see `cfgadm_shp`(1M)). Hotplug administrative models between PCI, CompactPCI remain the same except where noted in this document.

For PCI Hot Plug, each hotplug slot on a specific PCI bus is represented by an attachment point of that specific PCI bus.

An attachment point consist of two parts: a receptacle and an occupant. The `receptacle` under PCI Hot Plug is usually referred to as the physical hotpluggable slot; and the `occupant` is usually referred to as the PCI adapter card that plugs into the slot.

Attachment points are named through `ap_id`s. There are two types of `ap_id`s: logical and physical. The physical `ap_id` is based on the physical pathname, that is, `/devices/pci@1/hpc0_slot3`, whereas the logical `ap_id` is a shorter, and more user-friendly name. For PCI hotpluggable slots, the logical `ap_id` is usually the corresponding hotplug controller driver name plus the logical slot number, that is, `pci0:hpc0slot1`; PCI nexus driver, with hotplug controller driver named `hpc` and slot number `1`. The `ap_type` for PCI Hot Plug is `pci`.

Note that the `ap_type` is not the same as the information in the `Type` field.

See the *System Administration Guide: Basic Administration* for a detailed description of the hotplug procedure.

Options The following options are supported:

`-c` *function*
The following *function*s are supported for PCI hotpluggable slots:

configure
Configure the PCI device in the slot to be used by Solaris.

connect
Connect the slot to PCI bus.

disconnect
Disconnect the slot from the PCI bus.

insert
Not supported.

remove
Not supported.

unconfigure
Logically remove the PCI device's resources from the system.

`-f`
Not supported.

`-h` *ap_id* | *ap_type*
Print out PCI Hot Plug-specific help message.

`-l` *list*
List the values of PCI Hot Plug slots.

`-o` *hardware_options*
No hardware specific options are currently defined.

`-s` *listing_options*
Same as the generic `cfgadm(1M)`.

`-t` *ap_id*
This command is only supported on platforms which support testing capability on the slot.

`-v`
Execute in verbose mode.

When the `-v` option is used with the `-l` option, the `cfgadm` command outputs information about the attachment point. For PCI Hotplug attachment points located in a PCI PCI Express hierarchy, see `cfgadm_shp(1M)` for details. For PCI Hot Plug attachment points not located in a PCI Express hiearchy, the `Information` field will be the slot's system label, if any. This string will be obtained from the `slot-name` property of the slot's bus node. The information in the `Type` field is printed with or without the `v` option. The occupant `Type` field will describe the contents of the slot. There are 2 possible values:

unknown
The slot is empty. If a card is in the slot, the card is not configured or there is no driver for the device on the card.

subclass/*board*
The card in the slot is either a single-function or multi-function device.

subclass is a string representing the subclass code of the device, for example, SCSI, `ethernet`, `pci-isa`, and so forth. If the card is a multi-functional device, `MULT` will get printed instead.

board is a string representing the board type of the device. For example, hp is the string used for a PCI Hot Plug adapter, hs is used for a Hot Swap Board, nhs for a Non—Hot Swap cPCI Board, bhs for a Basic Hot Swap cPCI Board, and fhs for a Full Hot Swap cPCI Board.

Most PCI cards with more than one device are not multi-function devices, but are implemented as a PCI bridge with arbitrary devices behind them. In those cases, the subclass displayed is that of the PCI bridge. Most commonly, the bridges are `pci-pci,` a generic PCI to PCI bridge or `stpci`, a semi-transparent PCI bridge.

`-x` *hardware_function*
Perform hardware specific function. These hardware specific functions should not normally change the state of a receptacle or occupant.

The following *hardware_functions* are supported:

enable_slot | disable_slot
Change the state of the slot and preserve the state of slot across reboot. Preservation of state across reboot is only supported on select platforms.

`enable_slot` enables the addition of hardware to this slot for hotplugging and at boot time.

`disable_slot` disables the addition of hardware to this slot for hotplugging and at boot time. When a slot is disabled its condition is shown as unusable.

enable_autoconfig | disable_autoconfig
Change the ability to autoconfigure the occupant of the slot. Only platforms that support auto configuration support this feature.

`enable_autoconfig` enables the ability to autoconfigure the slot.

`diable_autoconfig` disables the ability to autoconfigure the slot.

Autoconfiguration is done through the attention button on the PCI Express platforms and through the injector/ejector latch on the CompactPCI platforms. When autoconfiguration is disabled, the attention button or latch mechanism cannot be used to configure the occupant of the slot.

led=[*led_sub_arg*],mode=[*mode_sub_arg*]
Without sub-arguments, print a list of the current LED settings. With sub-arguments, set the mode of a specific LED for a slot.

Specify *led_sub_arg* as `fault`, `power`, `attn`, or `active`.

Specify *mode_sub_arg* as `on`, `off` or `blink`.

Changing the state of the LED does not change the state of the receptacle or occupant. Normally, the LEDs are controlled by the hotplug controller, no user intervention is necessary. Use this command for testing purposes.

Caution: Changing the state of the LED can misrepresent the state of occupant or receptacle.

The following command prints the values of LEDs:

```
example# cfgadm -x led pci0:hpc0_slot1
Ap_Id             Led
pci0:hpc0_slot1   power=on,fault=off,active=off,attn=off
```

The following command turns on the Fault LED:

```
example# cfgadm -x led=fault,mode=on pci0:hpc0_slot1
```

The following command turns off the Power LED:

```
example# cfgadm -x led=power,mode=off pci0:hpc0_slot0
```

The following command sets the *active* LED to blink to indicate the location of the slot:

```
example# cfgadm -x led=active,mode=on pci0:hpc0_slot3
```

Examples

EXAMPLE 1 Printing out the Value of Each Slot

The following command prints out the values of each slot:

```
example# cfgadm -l
Ap_Id            Type         Receptacle   Occupant       Condition
c0               scsi-bus     connected    configured     unknown
c1               scsi-bus     connected    unconfigured   unknown
c2               scsi-bus     connected    unconfigured   unknown
cpci_slot1       stpci/fhs    connected    configured     ok
cpci_slot2       unknown      empty        unconfigured   unknown
cpci_slot4       stpci/fhs    connected    configured     ok
cpci_slot5       stpci/fhs    connected    configured     ok
```

EXAMPLE 2 Replacing a Card

The following command lists all DR-capable attachment points:

```
example# cfgadm


Type             Receptacle   Occupant     Condition
c0               scsi-bus     connected    configured     unknown
c1               scsi-bus     connected    unconfigured   unknown
```

EXAMPLE 2 Replacing a Card *(Continued)*

```
c2              scsi-bus     connected    unconfigured   unknown
cpci_slot1      stpci/fhs    connected    configured     ok
cpci_slot2      unknown      empty        unconfigured   unknown
cpci_slot4      stpci/fhs    connected    configured     ok
cpci_slot5      stpci/fhs    connected    configured     ok
```

The following command unconfigures and electrically disconnects the card:

```
example# cfgadm -c disconnect cpci_slot4
```

The change can be verified by entering the following command:

```
example# cfgadm cpci_slot4
```

```
Ap_Id                  Type         Receptacle   Occupant     Condition
cpci_slot4             unknown      disconnected unconfigured unknown
```

Now the card can be swapped. The following command electrically connects and configures the card:

```
example# cfgadm -c configure cpci_slot4
```

The change can be verifed by entering the following command:

```
example# cfgadm cpci_slot4
```

```
Ap_Id                  Type         Receptacle   Occupant     Condition
cpci_slot4             stpcipci/fhs connected    configured   ok
```

Files `/usr/lib/cfgadm/pci.so.1`
Hardware specific library for PCI hotplugging.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library

See Also `cfgadm(1M)`, `cfgadm_shp(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`

System Administration Guide: Basic Administration

Name cfgadm_sata – SATA hardware-specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -c function ap_id...

/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_options]
     -x hardware_function ap_id...

/usr/sbin/cfgadm [-v] [-a] [-s listing_options]
     [-o hardware_options] [-l [ap_id | ap_type]...]

/usr/sbin/cfgadm [-v] [-o harware_options] -t ap_id...

/usr/sbin/cfgadm [-v] [-o hardware_options] -h [ap_id]...
```

Description The SATA hardware specific library, `/usr/lib/cfgadm/sata.so.1`, provides the functionality for SATA hot plugging through the `cfgadm` command. `cfgadm` operates on attachment points, which are locations in the system where hardware resources can be dynamically reconfigured. See `cfgadm(1M)` for information regarding attachment points.

Each SATA controller's and port multiplier's device port is represented by an attachment point in the device tree. SATA devices, connected and configured in the system are shown as the attachment point name extension. The terms “attachment point” and “SATA port” are used interchangeably in the following description.

Attachment points are named through `ap_id`s. All the SATA attachment points `ap_id` consist of a string in the following form:

```
sataX/P[.M][::dsk/cXtYd0]
```

where

X	is the SATA controller number
P	is the SATA controller's device port number (0 to 31)
M	is the port multiplier's device port number (0 to 14) the port multiplier host port number (15). It is used only when the port multiplier is attached to the SATA controller's device port.
dev/cXtYd0	identifies the attached SATA device
Y	is a target number

In general, the device identifier is derived from the corresponding logical link for the device in */dev*. Because only one LUN (LUN 0) is supported by the SATA device, the “d” component of the device string will always have number 0 (zero).

For example, the logical `ap_id` of the device port 4 of the port multiplier connected to the device port 5 of the SATA controller 2 would be:

```
sata2/5.4
```

If the SATA disk or CD/DVD device is connected to this attachment point, and the device is configured, the *ap_id* would be:

```
sata2/5.4::dsk/c2t645d0
```

The *cXtYd0* string identifying a device has one-to-one correspondence to the device attachment point.

A simple listing of attachment points in the system will include all SATA device ports and attached devices. For example:

```
#cfgadm -l
Ap_Id                      Type        Receptacle   Occupant     Condition
  sata0/0::dev/c0t0d0      disk        connected    configured   ok
  sata0/1::dev/c0t1d0      disk        connected    configured   ok
  sata0/2::dev/c0t2d0      cd-dvd      connected    configured   ok
  sata0/3                  sata-port   empty        unconfigured ok
  sata1/0                  sata-port   disconnected unconfigured unknown
  sata1/1                  sata port   disconnected unconfigured unknown
  sata1/2                  sata port   empty        unconfigured ok
  sata1/3.15               sata-pmult  connected    configured   ok
  sata1/3.0::dev/c0t512d0 disk         connected    configured   ok
  sata1/3.1                sata-port   empty        unconfigured ok
  sata1/3.2                sata-port   empty        unconfigured ok
  sata1/3.3                sata-port   empty        unconfigured ok
  usb0/1                   unknown     empty        unconfigured ok
  usb0/2                   unknown     empty        unconfigured ok
```

See `cfgadm(1M)`for more information regarding listing of attachment points.

The receptacle state for attachment point at the SATA port have the following meanings:

empty — The SATA port is powered-on and enabled. No device presence was detected on this port.

disconnected — The SATA port is not enabled or the SATA device presence was detected but no communication with the device was established, or the port has failed.

connected — The SATA device is detected on the port the communication with the device is established.

The occupant (device attached to the SATA port) state have the following meanings:

configured — The attached SATA device is configured and ready to use by the operating system.

unconfigured — No device is attached, or the SATA device attached to the SATA port was not yet configured. To configure it, run the command "`cfgadm -c configure ap_id`".

The attachment point (SATA port) condition have the following meanings:

ok	The SATA port is powered-on and enabled, and is ready for use.
failed	The SATA port failed. It may be disabled and/or powered-off by the system. It is unusable and its condition is unknown. It may be due to the device plugged-in.
unknown	The SATA port is disabled and its condition is unknown.

A "state table" is the combination of an attachment point receptacle state, an occupant state, and an attachment point (SATA port) condition. The valid states are:

empty/unconfigured/ok	The SATA port is enabled and active. No device presence was detected.
disconnected/unconfigured/ok	The SATA port is enabled and a device presence was detected but no communications with the device was established.
disconnected/unconfigured/unknown	The SATA Port is disabled and its condition is unknown.
disconnected/unconfigured/failed	The SATA Port is disabled and unusable. The port was disabled by the system due to a system-detected failure.
connected/unconfigured/ok	The SATA Port is enabled and active. A device presence was detected and the communication with a device was established. The device is not configured to be used by the OS.
connected/configured/ok	The device is present and configured, and is ready to use by the OS.

Options `cfgadm` defines several types of operations besides listing (`-l`). These operations include testing, (`-t`), invoking configuration state changes, (`-c`), invoking hardware specific functions (`-x`), and obtaining configuration administration help messages (`-h`).

`-c` *function*

The following generic *function*s are defined for the SATA hardware specific library. For SATA port attachment point, the following configuration state change operations are supported:

connect

Enable (activate) the SATA port and establish the communication with an attached device. This operation implies powering-on the port if necessary.

disconnect

Unconfigure the attached device, if it is not already unconfigured, and disable (deactivate) the SATA port. A subsequent "`connect`" command enables SATA port

operation but does not bring a device to the "configured" state.

For a SATA device attached to the SATA port following state change operations are supported:

configure	Configure new device for use by the operating system if it is not already configured. This command also implies connect operation, if necessary.
unconfigure	Unconfigure the device connected to the SATA port if it is not already unconfigured.

The `configure` and `unconfigure` operations cannot be used for an attachment point where the port multiplier is connected. Port multipliers are configured and unconfigured automatically by the system. However, configure and unconfigure operations apply to all SATA devices connected to the port multiplier's device ports.

`-f`
Not supported.

`-h` *ap_id*
SATA specific help can be obtained by using the help option with any SATA attachment point.

`-l` [`-v`]
The `-l` option works as described in `cfgadm(1M)`. When paired with the `-v` option, the "Information" field contains the following SATA-specific information:

- Mfg: manufacturer string
- Product: product string
- No: product Serial Number

`-o` *hardware_options*
No hardware specific options are currently defined.

`-s` *listing_options*
Attachment points of class SATA can be listed by using the select suboption. See `cfgadm(1M)`.

`-t` *ap_id*
Perform self-test of the SATA port, if supported by the SATA controller. If a port self-test operation is not supported by the SATA controller, an error message is issued.

`-x` *hardware_function*
Perform hardware specific function.

Some of the following commands used on the SATA ports or the SATA controller may affect any SATA devices that have been attached, as noted. `ap_id` refers to SATA port or the entire SATA controller, as noted. If the operation implies unconfiguring a device, but it cannot be unconfigured (that is, the device contains a mounted filesystem), an error

message is issued and the operation is not performed. An error message will be also issued if the SATA controller does not support specified operation.

sata_reset_device ap_id
: Reset the SATA device attached to `ap_id` SATA port. The SATA port state does not change.

sata_reset_port ap_id
: Reset the SATA port specified by `ap_id`. If a SATA device is attached to the port, it is also reset. This operation may be also performed on the port to which a port multiplier is connected. If a port multiplier is connected to the SATA controller port, the SATA devices attached to the port multiplier may not be reset

sata_reset_all ap_id
: Reset SATA controller specified by the controller number part in `ap_id` and all attached devices and re-enumerate all connected devices, including port multipliers and devices connected to port multipliers' device ports.

  This operations implies unconfiguring all attached devices prior to the operation. Any newly enumerated devices will be left unconfigured.

sata_port_deactivate ap_id
: Force the deactivation of the port when all else fails. This is meant as an emergency step; use with caution.

sata_port_activate ap_id
: Force the activation of a port. This is meant for emergency situations on a port which was deactivated to recover from errors.

sata_port_self_test ap_id
: Perform self-test operation on the SATA controller. This operation implies unconfiguring all devices and resetting the SATA controller.

`-v`
: Execute in verbose mode.

  The following Transitions table reports the state transitions resulting from the `-c` operations and hotplugging actions:

```
current state     operation       possible new state
-------------     ---------       ------------------
empty/
unconfigured/ok   device plug-in  connected/unconfigured/ok, or
                                  disconnected/unconfigured/ok, or
                                  disconnected/unconfigured/failed

empty/
unconfigured/ok   -c unconfigure  error message, no state change

empty/
```

```
unconfigured/ok   -c configure    error message, no state change

empty/
unconfigured/ok   -c connect      error message, no state change

empty/
unconfigured/ok   -c disconnect   disconnected/unconfigured/unknown, or
                                  disconnected/unconfigured/failed

disconnected/
unconfigured/ok   device unplug   no state change

disconnected/
unconfigured/ok   -c unconfigure  error message, no state change

disconnected/
unconfigured/ok   -c configure    error message, no state change

disconnected/
unconfigured/ok   -c connect      error message, no state change

disconnected/
unconfigured/ok   -c disconnect   error message, no state change

disconnected/
unconfigured/
unknown
(no disk plugged) -c configure    error message, state change to
                                  empty/unconfigured/ok, or
                                  disconnected/unconfigured/failed

disconnected/
unconfigured/
unknown           -c configure    state change to
(disk plugged)                    connected/configured/ok or,
                                  connected/unconfigured/ok, or
                                  disconnected/unconfigured/failed and
                                  possible error message

disconnected/
unconfigured/
unknown           -c connect      empty/unconfigured/ok, or
                                  connected/unconfigured/ok, or
                                  disconnected/unconfigured/ok, or
                                  disconnected/unconfigured/unknown, or
                                  disconnected/unconfigured/failed

disconnected/
```

unconfigured/ unknown	-c disconnect	error message, no state change
disconnected/ unconfigured/ failed	any command other than -x commands	error message, no state change
connected/ unconfigured/ok	disk unplug	error message and state: empty/unconfigured/ok, or disconnected/unconfigured/failed
connected/ unconfigured/ok	-c configure	connected/unconfigured/ok, or connected/configured/ok, or disconnected/unconfigured/ok, or disconnected/unconfigured/failed
connected/ unconfigured/ok	-c unconfigure	error message, no state change
connected/ unconfigured/ok	-c connect	error message, no state change
connected/ unconfigured/ok	-c disconnect	disconnected/unconfigured/unknown, or disconnected/unconfigured/failed
connected/ configured/ok	disk unplug	error message and state: empty/unconfigured/ok, or disconnected/unconfigured/failed
connected/ configured/ok	-c configure	error message, no state change
connected/ configured/ok	-c unconfigure	error message, if device cannot be unconfigured, no state change, or connected/unconfigured/ok, or disconnected/unconfigured/ok, or disconnected/unconfigured/failed
connected/ configured/ok	-c connect	error message, no state change

```
connected/
configured/ok     -c disconnect   error message, if device cannot be
                                  unconfigured, no state change, or
                                  disconnected/unconfigured/unknown, or
                                  disconnected/unconfigured/failed
```

Examples

EXAMPLE 1 Configuring a Disk

The following command configures a disk attached to SATA controller 0, port 0:

```
example# cfgadm -c configure sata0/0
```

This command should be issued only when there is a device connected to the SATA port.

EXAMPLE 2 Unconfiguring a Disk

The following command unconfigures a disk attached to SATA controller 0, port 3:

```
example# cfgadm -c unconfigure sata0/3::dsk/c0t3d0
```

The device identifying string is shown when the attachment point receptacle state is "connected" and occupant state is "configured".

EXAMPLE 3 Encountering a Mounted File System While Unconfiguring a Disk

The following command illustrates encountering a mounted file system while unconfiguring a disk:

```
example# cfgadm -c unconfigure sata1/5::dsk/c01t35d0
```

The system responds with the following:

```
cfgadm: Component system is busy, try again: failed to offline:
/devices/pci@0,0/pci8086,244e@1e/pci1095,3124@1/sd@5,0
     Resource              Information
------------------  -------------------------
/dev/dsk/c1t5d0s0   mounted filesystem "/mnt"
```

Files

/usr/lib/cfgadm/sata.so.1 Hardware specific library for generic SATA hot plugging.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl

See Also

`cfgadm(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`

Notes

The emergency "sata_port_deactivate" operation is not supported on ports with attached disks containing critical partitions such as root (/), /usr, swap, or /var. The deactivate operation should not be attempted on such ports. Incorrect usage can result in a system hang and require a reboot.

Hotplugging operations are not supported by all SATA controllers.

If SATA connectors are the hot-pluggable type and the SATA controller supports hotplugging, a SATA device can be hotplugged at any time. The system detects the event and establishes the communication with the device. The device has to be configured by the explicit "`cfgadm -c configure ap_id`" command.

If the SATA connectors are the hot-pluggable type and the SATA controller supports hotplugging, unplugging a device without unconfiguring it may result in system hang or data loss. If a device is unconfigured but receptacle state is not in a disconnected state, unplugging a device from the SATA port will result in error message.

Warnings The connectors on some SATA devices do not conform to SATA hotplug specifications. Performing hotplug operations on such devices can cause damage to the SATA controller and/or the SATA device.

Name cfgadm_sbd – `cfgadm` commands for system board administration

Synopsis

```
cfgadm  -l [-a] [-o parsable]  ap_id...

cfgadm  -c  function [-f] [-y | -n]
     [-o unassign | nopoweroff] [-v]  ap_id...

cfgadm  -t [-v]  ap_id...

cfgadm  -x  [-f] [-v] function  ap_id...
```

Description The `cfgadm_sbd` plugin provides dynamic reconfiguration functionality for connecting, configuring, unconfiguring, and disconnecting class `sbd` system boards. It also enables you to connect or disconnect a system board from a running system without having to reboot the system.

The `cfgadm` command resides in `/usr/sbin`. See `cfgadm(1M)`. The `cfgadm_sbd` plugin resides `/usr/platform/sun4u/lib/cfgadm`.

Each board slot appears as a single attachment point in the device tree. Each component appears as a dynamic attachment point. You can view the type, state, and condition of each component, and the states and condition of each board slot by using the `-a` option.

The `cfgadm` options perform differently depending on the platform. Additionally, the form of the attachment points is different depending on the platform. See the `Platform Notes` section for more information.

Component Conditions The following are the names and descriptions of the component conditions:

failed The component failed testing.

ok The component is operational.

unknown The component has not been tested.

Component States The following is the name and description of the receptacle state for components:

connected The component is connected to the board slot.

The following are the names and descriptions of the occupant states for components:

configured The component is available for use by the Solaris operating environment.

unconfigured The component is not available for use by the Solaris operating environment.

Board Conditions The following are the names and descriptions of the board conditions.

failed The board failed testing.

ok The board is operational.

unknown The board has not been tested.

unusable The board slot is unusable.

Board States Inserting a board changes the receptacle state from empty to disconnected. Removing a board changes the receptacle state from disconnected to empty.

Caution: Removing a board that is in the connected state or that is powered on and in the disconnected state crashes the operating system and can result in permanent damage to the system.

The following are the names and descriptions of the receptacle states for boards:

connected The board is powered on and connected to the system bus. You can view the components on a board only after it is in the connected state.

disconnected The board is disconnected from the system bus. A board can be in the disconnected state without being powered off. However, a board must be powered off and in the disconnected state before you remove it from the slot.

empty A board is not present.

The occupant state of a disconnected board is always unconfigured. The following table contains the names and descriptions of the occupant states for boards:

configured At least one component on the board is configured.

unconfigured All of the components on the board are unconfigured.

Dynamic System Domains Platforms based on dynamic system domains (DSDs, referred to as domains in this document) divide the slots in the chassis into electrically isolated hardware partitions (that is, DSDs). Platforms that are not based on DSDs assign all slots to the system permanently.

A slot can be empty or populated, and it can be assigned or available to any number of domains. The number of slots available to a given domain is controlled by an available component list (ACL) that is maintained on the system controller. The ACL is not the access control list provided by the Solaris operating environment.

A slot is visible to a domain only if the slot is in the domain's ACL and if it is not assigned to another domain. An unassigned slot is visible to all domains that have the slot in their ACL. After a slot has been assigned to a domain, the slot is no longer visible to any other domain.

A slot that is visible to a domain, but not assigned, must first be assigned to the domain before any other state changing commands are applied. The assign can be done explicitly using `-x assign` or implicitly as part of a connect. A slot must be unassigned from a domain before it can be used by another domain. The unassign is always explicit, either directly using `-x unassign` or as an option to disconnect using `-o unassign`.

State Change Functions

Functions that change the state of a board slot or a component on the board can be issued concurrently against any attachment point. Only one state changing operation is permitted at a given time. A `Y` in the Busy field in the state changing information indicates an operation is in progress.

The following list contains the functions that change the state:

- configure
- unconfigure
- connect
- disconnect

Availability Change Functions

Commands that change the availability of a board can be issued concurrently against any attachment point. Only one availability change operation is permitted at a given time. These functions also change the information string in the `cfgadm -l` output. A `Y` in the Busy field indicates that an operation is in progress.

The following list contains the functions that change the availability:

- `assign`
- `unassign`

Condition Change Functions

Functions that change the condition of a board slot or a component on the board can be issued concurrently against any attachment point. Only one condition change operation is permitted at a given time. These functions also change the information string in the `cfgadm -l` output. A `Y` in the Busy field indicates an operation is in progress.

The following list contains the functions that change the condition:

- `poweron`
- `poweroff`
- `test`

Unconfigure Process

This section contains a description of the unconfigure process, and illustrates the states of source and target boards at different stages during the process of moving permanent memory.

In the following code examples, the permanent memory on board 0 must be moved to another board in the domain. Thus, board 0 is the source, and board 1 is the target.

A status change operation cannot be initiated on a board while it is marked as busy. For brevity, the CPU information has been removed from the code examples.

The process is started with the following command:

```
# cfgadm -c unconfigure -y SB0::memory &
```

First, the memory on board 1 in the same address range as the permanent memory on board 0 must be deleted. During this phase, the source board, the target board, and the memory attachment points are marked as busy. You can display the status with the following command:

```
# cfgadm -a -s cols=ap_id:type:r_state:o_state:busy SB0 SB1

Ap_Id          Type      Receptacle     Occupant       Busy
SB0            CPU       connected      configured     y
SB0::memory    memory    connected      configured     y
SB1            CPU       connected      configured     y
SB1::memory    memory    connected      configured     y
```

After the memory has been deleted on board 1, it is marked as unconfigured. The memory on board 0 remains configured, but it is still marked as busy, as in the following example.

```
Ap_Id          Type      Receptacle     Occupant       Busy
SB0            CPU       connected      configured     y
SB0::memory    memory    connected      configured     y
SB1            CPU       connected      configured     y
SB1::memory    memory    connected      unconfigured   n
```

The memory from board 0 is then copied to board 1. After it has been copied, the occupant state for the memory is switched. The memory on board 0 becomes unconfigured, and the memory on board 1 becomes configured. At this point in the process, only board 0 remains busy, as in the following example.

```
Ap_Id          Type      Receptacle     Occupant       Busy
SB0            CPU       connected      configured     y
SB0::memory    memory    connected      unconfigured   n
SB1            CPU       connected      configured     n
SB1::memory    memory    connected      configured     n
```

After the entire process has been completed, the memory on board 0 remains unconfigured, and the attachment points are not busy, as in the following example.

```
Ap_Id          Type      Receptacle     Occupant       Busy
SB0            CPU       connected      configured     n
SB0::memory    memory    connected      unconfigured   n
SB1            CPU       connected      configured     n
SB1::memory    memory    connected      configured     n
```

The permanent memory has been moved, and the memory on board 0 has been unconfigured. At this point, you can initiate a new state changing operation on either board.

Platform-Specific Options

You can specify platform-specific options that follow the options interpreted by the system board plugin. All platform-specific options must be preceded by the `platform` keyword. The following example contains the general format of a command with platform-specific options:

command `-o` *sbd_options*`,platform=`*platform_options*

Options

This man page does not include the `-v`, `-a`, `-s`, or `-h` options for the `cfgadm` command. See `cfgadm`(1M) for descriptions of those options. The following options are supported by the `cfgadm_sbd` plugin:

`-c` *function* Performs a state change function. You can use the following functions:

unconfigure Changes the occupant state to unconfigured. This function applies to system board slots and to all of the components on the system board.

The `unconfigure` function removes the CPUs from the CPU list and deletes the physical memory from the system memory pool. If any device is still in use, the `cfgadm` command fails and reports the failure to the user. You can retry the command as soon as the device is no longer busy. If a CPU is in use, you must ensure that it is off line before you proceed. See `pbind(1M)`, `psradm(1M)` and `psrinfo(1M)`.

The `unconfigure` function moves the physical memory to another system board before it deletes the memory from the board you want to unconfigure. Depending of the type of memory being moved, the command fails if it cannot find enough memory on another board or if it cannot find an appropriate physical memory range.

For permanent memory, the operating system must be suspended (that is, quiesced) while the memory is moved and the memory controllers are reprogrammed. If the operating system must be suspended, you will be prompted to proceed with the operation. You can use the `-y` or `-n` options to always answer yes or no respectively.

Moving memory can take several minutes to complete, depending on the amount of memory and the system load. You can monitor the progress of the operation by issuing a status command against the memory attachment point. You can also interrupt the memory operation by stopping the `cfgadm` command. The deleted memory is returned to the system memory pool.

disconnect

Changes the receptacle state to disconnected. This function applies only to system board slots.

If the occupant state is configured, the `disconnect` function attempts to unconfigure the occupant. It then powers off the system board. At this point, the board can be removed from the slot.

This function leaves the board in the assigned state on platforms that support dynamic system domains.

If you specify `-o nopoweroff`, the `disconnect` function leaves the board powered on. If you specify `-o unassign`, the `disconnect` function unassigns the board from the domain.

If you unassign a board from a domain, you can assign it to another domain. However, if it is assigned to another domain, it is not available to the domain from which is was unassigned.

configure

Changes the occupant state to configured. This function applies to system board slots and to any components on the system board.

If the receptacle state is disconnected, the `configure` function attempts to connect the receptacle. It then walks the tree of devices that is created by the `connect` function, and attaches the devices if necessary. Running this function configures all of the components on the board, except those that have already been configured.

For CPUs, the `configure` function adds the CPUs to the CPU list. For memory, the `configure` function ensures that the memory is initialized then adds the memory to the system memory pool. The CPUs and the memory are ready for use after the `configure` function has been completed successfully.

For I/O devices, you must use the `mount` and the `ifconfig` commands before the devices can be used. See `ifconfig(1M)` and `mount(1M)`.

connect

Changes the receptacle state to connected. This function applies only to system board slots.

If the board slot is not assigned to the domain, the `connect` function attempts to assign the slot to the domain. Next, it powers on and tests the board, then it connects the board electronically to the system bus and probes the components.

After the `connect` function is completed successfully, you can use the `-a` option to view the status of the components on the board. The `connect` function leaves all of the components in the unconfigured state.

The assignment step applies only to platforms that support dynamic system domains.

`-f` Overrides software state changing constraints.

The `-f` option never overrides fundamental safety and availability constraints of the hardware and operating system.

`-l` Lists the state and condition of attachment points specified in the format controlled by the `-s`, `-v`, and `-a` options as specified in `cfgadm(1M)`. The `cfgadm_sbd` plugin provides specific information in the info field as described below. The format of this information might be altered by the `-o parsable` option.

The parsable `info` field is composed of the following:

cpu The `cpu` type displays the following information:

`cpuid=#[,#…]`	Where # is a number, and represents the ID of the CPU. If more than one # is present, this CPU has multiple active virtual processors.
`speed=#`	Where # is a number and represents the speed of the CPU in MHz.
`ecache=#`	Where # is a number and represents the size of the ecache in MBytes. If the CPU has multiple active virtual processors, the ecache could either be shared among the virtual processors, or divided between them.

memory The `memory` type displays the following information, as appropriate:

address=#	Where # is a number, representing the base physical address.
size=#	Where # is a number, representing the size of the memory in KBytes.

		permanent=#	Where # is a number, representing the size of permanent memory in KBytes.
		unconfigurable	An operating system setting that prevents the memory from being unconfigured.
		inter-board-interleave	The board is participating in interleaving with other boards.
		source=*ap_id*	Represents the source attachment point.
		target=*ap_id*	Represents the target attachment point.
		deleted=#	Where # is a number, representing the amount of memory that has already been deleted in KBytes.
		remaining=#	Where # is a number, representing the amount of memory to be deleted in KBytes.
	io	The `io` type displays the following information:	
		device=*path*	Represents the physical path to the I/O component.
		referenced	The I/O component is referenced.
	board	The `board` type displays the following boolean names. If they are not present, then the opposite applies.	
		assigned	The board is assigned to the domain.
		powered-on	The board is powered on.
		The same items appear in the `info` field in a more readable format if the `-o parsable` option is not specified.	
`-o` parsable	Returns the information in the `info` field as a boolean *name* or a set of `name=value` pairs, separated by a space character.		
	The `-o parsable` option can be used in conjunction with the `-s` option. See the `cfgadm(1M)` man page for more information about the `-s` option.		
`-t`	Tests the board.		
	Before a board can be connected, it must pass the appropriate level of testing.		

Use of this option always attempts to test the board, even if it has already passed the appropriate level of testing. Testing is also performed when a `-c connect` state change function is issued, in which case the test step can be skipped if the board already shows an appropriate level of testing. Thus the `-t` option can be used to explicitly request that the board be tested.

`-x` *function* Performs an sbd-class function. You can use the following functions:

assign Assigns a board to a domain.

The receptacle state must be disconnected or empty. The board must also be listed in the domain available component list. See Dynamic System Domains.

unassign Unassigns a board from a domain.

The receptacle state must be disconnected or empty. The board must also be listed in the domain available component list. See Dynamic System Domains.

poweron Powers the system board on.

The receptacle state must be disconnected.

poweroff Powers the system board off.

The receptacle state must be disconnected.

Operands The following operands are supported:

Receptacle *ap_id* For the Sun Fire high-end systems such as the Sun Fire 15K , the receptacle attachment point ID takes the form `SB`*X* or `IO`*X*, where *X* equals the slot number.

The exact format depends on the platform and typically corresponds to the physical labelling on the machine. See the platform specific information in the `NOTES` section.

Component *ap_id* The component attachment point ID takes the form *component_typeX*, where *component_type* equals one of the component types described in "Component Types" and *X* equals the component number. The component number is a board-relative unit number.

The above convention does not apply to memory compontents. Any DR action on a memory attachment point affects all of the memory on the system board.

Examples The following examples show user input and system output on a Sun Fire 15K system. User input, specifically references to attachment points and system output might differ on other Sun Fire systems, such as the Sun Fire midrange systems such as the 6800. Refer to the Platform Notes for specific information about using the `cfgadm_sbd` plugin on non-Sun Fire high-end models.

EXAMPLE 1 Listing All of the System Board

```
# cfgadm -a -s "select=class(sbd)"

Ap_Id          Type      Receptacle     Occupant       Condition
SB0            CPU       connected      configured     ok
SB0::cpu0      cpu       connected      configured     ok
SB0::memory    memory    connected      configured     ok
IO1            HPCI      connected      configured     ok
IO1::pci0      io        connected      configured     ok
IO1::pci1      io        connected      configured     ok
SB2            CPU       disconnected   unconfigured   failed
SB3            CPU       disconnected   unconfigured   unusable
SB4            unknown   empty          unconfigured   unknown
```

This example demonstrates the mapping of the following conditions:

- The board in Slot 2 failed testing.
- Slot 3 is unusable; thus, you cannot hot plug a board into that slot.

EXAMPLE 2 Listing All of the CPUs on the System Board

```
# cfgadm -a -s "select=class(sbd):type(cpu)"

Ap_Id          Type      Receptacle     Occupant       Condition
SB0::cpu0      cpu       connected      configured     ok
SB0::cpu1      cpu       connected      configured     ok
SB0::cpu2      cpu       connected      configured     ok
SB0::cpu3      cpu       connected      configured     ok
```

EXAMPLE 3 Displaying the CPU Information Field

```
# cfgadm -l -s noheadings,cols=info SB0::cpu0

cpuid 16, speed 400 MHz, ecache 8 Mbytes
```

EXAMPLE 4 Displaying the CPU Information Field in Parsable Format

```
# cfgadm -l -s noheadings,cols=info -o parsable SB0::cpu0

cpuid=16 speed=400 ecache=8
```

EXAMPLE 5 Displaying the Devices on an I/O Board

```
# cfgadm -a -s noheadings,cols=ap_id:info -o parsable IO1

IO1       powered-on assigned
IO1::pci0 device=/devices/saf@0/pci@0,2000 referenced
IO1::pci1 device=/devices/saf@0/pci@1,2000 referenced
```

EXAMPLE 6 Monitoring an Unconfigure Operation

In the following example, the memory sizes are displayed in Kbytes.

```
# cfgadm -c unconfigure -y SB0::memory &
# cfgadm -l -s noheadings,cols=info -o parsable SB0::memory SB1::memory

address=0x0 size=2097152 permanent=752592 target=SB1::memory
     deleted=1273680 remaining=823472
address=0x1000000 size=2097152 source=SB0::memory
```

EXAMPLE 7 Assigning a Slot to a Domain

```
# cfgadm -x assign SB2
```

EXAMPLE 8 Unassigning a Slot from a Domain

```
# cfgadm -x unassign SB3
```

Attributes

See `attributes(5)` for a description of the following attribute:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkvm.u
Stability	See below.

The interface stability is evolving. The output stability is unstable.

See Also

`cfgadm(1M)`, `devfsadm(1M)`, `ifconfig(1M)`, `mount(1M)`, `pbind(1M)`, `psradm(1M)`, `psrinfo(1M)`, `config_admin(3CFGADM)`, `attributes(5)`

Notes

This section contains information on how to monitor the progress of a memory delete operation. It also contains platform specific information.

Memory Delete Monitoring

The following shell script can be used to monitor the progress of a memory delete operation.

```
# cfgadm -c unconfigure -y SB0::memory &
# watch_memdel SB0

#!/bin/sh
# This is the watch_memdel script.
```

```
if [ -z "$1" ]; then
        printf "usage:  %s board_id\n" ‘basename $0‘
        exit 1
fi

board_id=$1

cfgadm_info=’cfgadm -s noheadings,cols=info -o parsable’

eval ‘$cfgadm_info $board_id::memory‘

if [ -z "$remaining" ]; then
        echo no memory delete in progress involving $board_id
        exit 0
fi

echo deleting target $target

while true
do
        eval ‘$cfgadm_info $board_id::memory‘

        if [ -n "$remaining" -a "$remaining" -ne 0 ]
        then
                echo $deleted KBytes deleted, $remaining KBytes remaining
                remaining=
        else
                echo memory delete is done
                exit 0
        fi
        sleep 1
done
exit 0
```

Sun Enterprise 10000 Platform Notes

The following syntax is used to refer to Platform Notes attachment points on the Sun Enterprise 10000 system:

board::*component*

where *board* refers to the system board; and *component* refers to the individual component. System boards can range from `SB0` (zero) to `SB15`. A maximum of sixteen system boards are available.

The DR 3.0 model running on a Sun Enterprise 10000 domain supports a limited subset of the functionality provided by the `cfgadm_sbd` plugin. The only supported operation is to view the status of attachment points in the domain. This corresponds to the `-l` option and all of its associated options.

Attempting to perform any other operation from the domain will result in an error that states that the operation is not supported. All operations to add or remove a system board must be initiated from the System Service Processor.

Sun Fire High-End System Platform Notes

The following syntax is used to refer to attachment points on the Sun Fire high-end systems:

board::*component*

where *board* refers to the system board or I/O board; and *component* refers to the individual component.

Depending on the system's configuration, system boards can range from `SB0` (zero) through `SB17`, and I/O boards can range from `IO0` (IO zero) through `IO17`. (A maximum of eighteen system and I/O boards are available).

The `-t` and `-x` options behave differently on the Sun Fire high-end system platforms. The following list describes their behavior:

`-t`
: The system controller uses a CPU to test system boards by running `LPOST`, sequenced by the `hpost` command. To test I/O boards, the driver starts the testing in response to the `-t` option, and the test runs automatically without user intervention. The driver unconfigures a CPU and a stretch of contiguous physical memory. Then, it sends a command to the system controller to test the board. The system controller uses the CPU and memory to test the I/O board from inside of a transaction/error cage. You can only use CPUs from system boards (not MCPU boards) to test I/O boards.

`-x assign | unassign`
: In the Sun Fire high-end system administration model, the platform administrator controls the platform hardware through the use of an available component list for each domain. This information is maintained on the system controller. Only the platform administrator can modify the available component list for a domain.

  The domain administrator is only allowed to assign or unassign a board if it is in the available component list for that domain. The platform administrator does not have this restriction, and can assign or unassign a board even if it is not in the available component list for a domain.

Sun Fire 15K Component Types

The following are the names and descriptions of the component types:

`cpu`	CPU
`io`	I/O device
`memory`	Memory

Note: An operation on a memory component affects all of the memory components on the board.

Sun Fire Midrange Systems Platform Notes

References to attachment points are slightly different on Sun Fire midrange servers such as the 6800, 4810, 4800, and 3800 systems than on the Sun Fire high-end systems. The following syntax is used to refer to attachment points on Sun Fire systems other than the Sun Fire 15K:

`N#`.*board*::*component*

where **`N#`** refers to the node; *board* refers to the system board or I/O board; and *component* refers to the individual component.

Depending on the system's configuration, system boards can range from `SB0` through `SB5`, and I/O boards can range from `IB6` through `IB9`. (A maximum of six system and four I/O boards are available).

Sun Fire Midrange System Component Types

The following are the names and descriptions of the component types:

`cpu`	CPU
`pci`	I/O device
`memory`	Memory

Note: An operation on a memory component affects all of the memory components on the board.

Name cfgadm_scsi – SCSI hardware specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_option]
     -c function ap_id...

/usr/sbin/cfgadm [-f] [-y | -n] [-v] [-o hardware_option]
     -x hardware_function ap_id...

/usr/sbin/cfgadm [-v] [-a] [-s listing_option] [-o hardware_option]
     [-l [ap_id | ap_type ... ]]

/usr/sbin/cfgadm [-v] [-o hardware_option] -t ap_id...

/usr/sbin/cfgadm [-v] [-o hardware_option] -h [ap_id]...
```

Description The SCSI hardware specific library `/usr/lib/cfgadm/scsi.so.1` provides the functionality for SCSI hot-plugging through the `cfgadm`(1M) command. `cfgadm` operates on attachment points, which are locations in the system where hardware resources can be dynamically reconfigured. Refer to `cfgadm`(1M) for information regarding attachment points.

For SCSI hot-plugging, each SCSI controller is represented by an attachment point in the device tree. In addition, each SCSI device is represented by a dynamic attachment point. Attachment points are named through `ap_ids`. Two types of `ap_ids` are defined: logical and physical. The physical `ap_id` is based on the physical pathname, whereas the logical `ap_id` is a shorter more user-friendly name. For SCSI controllers, the logical `ap_id` is usually the corresponding disk controller number. For example, a typical logical `ap_id` would be `c0`.

SCSI devices are named relative to the controller `ap_id`. Thus if a disk device is attached to controller `c0`, its `ap_id` can be:

```
c0::dsk/c0t0d0
```

where `dsk/c0t0d0` identifies the specific device. In general, the device identifier is derived from the corresponding logical link for the device in `/dev`. For example, a SCSI tape drive logical `ap_id` could be `c0::rmt/0`. Here `c0` is the logical `ap_id` for the `SCSI` controller and `rmt/0` is derived from the logical link for the tape drive in `/dev/rmt`. If an identifier can not be derived from the link in `/dev`, a unique identifier will be assigned to it. For example, if the tape device has no link in `/dev`, it can be assigned an `ap_id` of the form `c0::st3` where `st3` is a unique internally generated identifier.

When a controller is capable of supporting the Solaris I/O multipathing feature (formerly known as MPxIO or the Sun StorEdge Traffic Manager [STMS]) and is enabled, the controller provides a path to a `scsi_vhci`(7D) multipath device. If a device attached to such controller is supported by `scsi_vhci`(7D) its `ap_id` can be:

```
c0::0,0
```

...where `0,0` uniquely identifies the target and logical unit information. The `Type` field for a path of such `ap_id`s indicates if it represent a path to the `scsi_vhci(7D)` multipath devices, along with the type of device that is connected to through the path.

A simple listing of attachment points in the system will include attachment points at SCSI controllers but not SCSI devices. Use the `-a` flag to the list option (`-l`) to list SCSI devices as well. For example:

```
# cfgadm -l
  Ap_Id             Type         Receptacle     Occupant     Condition
  c0                scsi-bus     connected      configured   unknown
  sysctrl0:slot0    cpu/mem      connected      configured   ok
  sysctrl0:slot1    sbus-upa     connected      configured   ok
```

To list SCSI devices in addition to SCSI controllers:

```
# cfgadm -al

Ap_Id                 Type         Receptacle   Occupant     Condition
c0                    scsi-bus     connected    configured   unknown
c0::dsk/c0t14d0       disk         connected    configured   unknown
c0::dsk/c0t11d0       disk         connected    configured   unknown
c0::dsk/c0t8d0        disk         connected    configured   unknown
c0::dsk/c0t0d0        disk         connected    configured   unknown
c0::rmt/0             tape         connected    configured   unknown
sysctrl0:slot0        cpu/mem      connected    configured   ok
sysctrl0:slot1        sbus-upa     connected    configured   ok
```

If the controller `c0` was enabled with Solaris I/O multipathing and the connected disk and tape devices are supported by Solaris I/O multipathing the output would be:

```
# cfgadm -al

Ap_Id                 Type         Receptacle   Occupant     Condition
c0                    scsi-bus     connected    configured   unknown
c0::11,0              disk-path    connected    configured   unknown
c0::14,0              disk-path    connected    configured   unknown
c0::8,0               disk-path    connected    configured   unknown
c0::0,0               disk-path    connected    configured   unknown
c0::a.0               tape-path    connected    configured   unknown
sysctrl0:slot0        cpu/mem      connected    configured   ok
sysctrl0:slot1        sbus-upa     connected    configured   ok
```

Refer to `cfgadm(1M)` for more information regarding listing attachment points. The receptacle and occupant state for attachment points at the SCSI controller have the following meanings:

`empty`
: not applicable

`disconnected`
bus quiesced (I/O activity on bus is suspended)

`connected`
bus active

`configured`
one or more devices on the bus is configured

`unconfigured`
no device on the bus is configured

The corresponding states for individual SCSI devices are:

`empty`
not applicable

`disconnected`
bus to which the device is attached is quiesced

`connected`
bus to which device is attached is active

`configured`
device or path to a multipath SCSI device is configured

`unconfigured`
device or path to a multipath SCSI device is not configured

Options `cfgadm` defines several types of operations besides listing (`-l`).These operations include testing, (`-t`), invoking configuration state changes, (`-c`), invoking hardware specific functions (`-x`), and obtaining configuration administration help messages (`-h`).

`-c` *function*
The following generic commands are defined for the SCSI hardware specific library:

For SCSI controller attachment points, the following configuration state change operations are supported:

`connect`
Unquiesce the SCSI bus.

`disconnect`
Quiesce the bus (suspend I/O activity on bus).

Incorrect use of this command can cause the system to hang. See `NOTES`.

`configure`
Configure new devices on SCSI bus.

`unconfigure`
Unconfigure all devices connected to bus.

The following generic commands are defined for SCSI devices and for paths to multipath SCSI devices:

`configure`
Configure a specific device or a specific path to a multipath SCSI device.

`unconfigure`
Unconfigure a specific device or a specific path to a multipath SCSI device.

`-f`
When used with the `disconnect` command, forces a quiesce of the SCSI bus, if supported by hardware.

Incorrect use of this command can cause the system to hang. See `NOTES`.

`-h` *ap_id*
SCSI specific help can be obtained by using the help option with any SCSI attachment point.

`-o` *hardware_option*
No hardware specific options are currently defined.

`-s` *listing_option*
Attachment points of class `scsi` can be listed by using the `select` sub-option. Refer to the `cfgadm(1M)` man page for additional information.

`-t` *ap_id*
No test commands are available at present.

`-x` *hardware_function*
Some of the following commands can only be used with SCSI controllers and some only with SCSI devices.

In the following, *controller_ap_id* refers to an `ap_id` for a SCSI controller, for example, `c0`. *device_ap_id* refers to an `ap_id` for a SCSI device, for example: `c0::dsk/c0dt3d0`.

The following hardware specific functions are defined:

`insert_device` *controller_ap_id*
Add a new device to the SCSI controller, *controller_ap_id*.

This command is intended for interactive use only.

`remove_device` *device_ap_id*
Remove device *device_ap_id*.

This command is intended for interactive use only.

`replace_device` *device_ap_id*
Remove device *device_ap_id* and replace it with another device of the same kind.

This command is intended for interactive use only.

`reset_device` *device_ap_id*
Reset *device_ap_id*.

`reset_bus` *controller_ap_id*
Reset bus *controller_ap_id* without resetting any devices attached to the bus.

`reset_all` *controller_ap_id*
Reset bus *controller_ap_id* and all devices on the bus.

`locator` [`=on|off`] *device_ap_id*
Sets or gets the hard disk locator LED, if it is provided by the platform. If the [on|off] suboption is not set, the state of the hard disk locator is printed.

`led`[`=LED,mode=on|off|blink`] *device_ap_id*
If no sub-arguments are set, this function print a list of the current LED settings. If sub-arguments are set, this function sets the mode of a specific LED for a slot.

Examples

EXAMPLE 1 Configuring a Disk

The following command configures a disk attached to controller `c0`:

```
# cfgadm -c configure c0::dsk/c0t3d0
```

EXAMPLE 2 Unconfiguring a Disk

The following command unconfigures a disk attached to controller `c0`:

```
# cfgadm -c unconfigure c0::dsk/c0t3d0
```

EXAMPLE 3 Adding a New Device

The following command adds a new device to controller `c0`:

```
# cfgadm -x insert_device c0
```

The system responds with the following:

```
Adding device to SCSI HBA: /devices/sbus@1f,0/SUNW,fas@e,8800000
This operation will suspend activity on SCSI bus c0
Continue (yes/no)?
```

Enter:

```
y
```

The system responds with the following:

```
SCSI bus quiesced successfully.
It is now safe to proceed with hotplug operation.
Enter y if operation is complete or n to abort (yes/no)?
```

Enter:

```
y
```

EXAMPLE 4 Replacing a Device

The following command replaces a device attached to controller `c0`:

```
# cfgadm -x replace_device c0::dsk/c0t3d0
```

The system responds with the following:

```
Replacing SCSI device: /devices/sbus@1f,0/SUNW,fas@e,8800000/sd@3,0
This operation will suspend activity on SCSI bus: c0
Continue (yes/no)?
```

Enter:

```
y
```

The system responds with the following:

```
SCSI bus quiesced successfully.
It is now safe to proceed with hotplug operation.
Enter y if operation is complete or n to abort (yes/no)?
```

Enter:

```
y
```

EXAMPLE 5 Encountering a Mounted File System While Unconfiguring a Disk

The following command illustrates encountering a mounted file system while unconfiguring a disk:

```
# cfgadm -c unconfigure c1::dsk/c1t0d0
```

The system responds with the following:

```
cfgadm: Component system is busy, try again: failed to offline:
/devices/pci@1f,4000/scsi@3,1/sd@1,0
     Resource             Information
------------------  -------------------------
/dev/dsk/c1t0d0s0   mounted filesystem "/mnt"
```

EXAMPLE 6 Displaying the Value of the Locator for a Disk

The following command displays the value of the locator for a disk. This example is specific to the SPARC Enterprise Server family:

```
# cfgadm -x locator c0::dsk/c0t6d0
```

The system responds with the following:

```
Disk                    Led
c0t6d0                  locator=on
```

EXAMPLE 7 Setting the Value of the Locator for a Disk

The following command sets the value of the locator for a disk. This example is specific to the SPARC Enterprise Server family:

```
# cfgadm -x locator=off c0::dsk/c0t6d0
```

The system does not print anything in response.

EXAMPLE 8 Configuring a Path to a Multipath SCSI Disk

The following command configures a path connected through controller `c0`:

```
# cfgadm -c configure c0::2,0
```

EXAMPLE 9 Unconfiguring a Path to a Multipath SCSI Disk

The following command unconfigures a path connected through controller `c0`:

```
# cfgadm -c unconfigure c0::2,0
```

Files

`/usr/lib/cfgadm/scsi.so.1`
hardware-specific library for generic SCSI hot-plugging

`/usr/platform/SPARC-Enterprise/lib/cfgadm/scsi.so.1`
platform-specific library for generic SCSI hot-plugging

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library

See Also

`cfgadm(1M)`, `luxadm(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`, `scsi_vhci(7D)`

Notes

The `disconnect` (quiesce) operation is not supported on controllers which control disks containing critical partitions such as `root` (`/`), `/usr`, `swap`, or `/var`. The `disconnect` operation should not be attempted on such controllers. Incorrect usage can result in a system hang and require a reboot.

When a controller is in the disconnected (quiesced) state, there is a potential for deadlocks occurring in the system. The `disconnect` operation should be used with caution. A controller should be kept in the disconnected state for the minimum period of time required to accomplish the DR operation. The `disconnect` command is provided only to allow the replacement of the SCSI cables while the system is running. It should not be used for any other purpose. The only fix for a deadlock (if it occurs) is to reboot the system.

Hotplugging operations are not supported by all SCSI controllers.

Warnings The connectors on some SCSI devices do not conform to SCSI hotplug specifications. Performing hotplug operations on such devices can cause damage to the hardware on the SCSI bus. Refer to your hardware manual for additional information.

Name cfgadm_shp – PCI Express and Standard PCI Hotplug hardware-specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v]
     [-o hardware_options] -c function ap_id [ap_id]

/usr/sbin/cfgadm [-f] [-y | -n] [-v]
     [-o hardware_options] -x hardware_function ap_id [ap_id]

/usr/sbin/cfgadm [-v] [-s listing_options]]
     [-o hardware_options] -x hardware_function ap_id [ap_id]

/usr/sbin/cfgadm [-v] [-o hardware_options] -tap_id [ap_id]

/usr/sbin/cfgadm [-v] [-o hardware_function]-h [ap_id | ap_type]
```

Description The PCI Express and Standard PCI Hotplug hardware-specific library, `/usr/lib/cfgadm/shp.so.1`, provides support for hotplugging PCI Express and Standard PCI Hotplug adapter cards into the respective hotpluggable slots in a system that is hotplug-capable, through the `cfgadm` command (see `cfgadm(1M)`). Support for the rest PCI Hotplug adapter cards (other than PCI Express and Standard PCI Hotplug cards) are provided by `cfgadm_pci` library (see `cfgadm_pci(1M)`). Hotplug administrative models between PCI Express Hotplug and Standard PCI Hotplug remain the same except where noted in this man page.

For PCI hotplug, each hotplug slot on a specific PCI bus is represented by an attachment point of that PCI bus.

An attachment point consist of two parts: a receptacle and an occupant. The receptacle under PCI hotplug is usually referred to as the physical hot pluggable slot; and the occupant is usually referred to as the PCI adapter card that plugs into the slot.

Attachment points are named through `ap_id`s. There are two types of `ap_id`s: logical and physical. The physical `ap_id` is based on the physical pathname, for example:

```
/devices/pci@7c,0/pci10de,5d@d:pcie2
```

Whereas the logical `ap_id` is a shorter, more user-friendly name, for example, `pcie2`. The `ap_type` for Hotplug PCI is `pci`.

Note that the `ap_type` is not the same as the information in the `Type` field.

PCI Express ap_id Naming For attachment points located in a PCI Express hierarchy (that is, the parent or an ancestor is a PCI Express device), including attachment points that are not PCI Express devices themselves, the naming scheme shown below is used.

Grammar:

`APID` : *absolute-slot-path*
Fundamental term.

absolute-slot-path : *slot-path*[:*slot-path*[:*slotpath* ...]]
...where *fru-id* indicates the chassis FRU, if any, containing the *slot-id*.

fru-id : *fru-type*[*serialid#*]
...where *fru-type* is "iob" for a PCI Express expansion chassis, followed by its serial number *serialid#*, if available

slot-id : *slot-name* | *device-type physical-slot#* | \
nexus-driver-name nexus-driver-instance.\
device-type pci-device-number
...where *slot-name* is a name assigned by the platform or hardware itself. *device-type* is either `pcie` for PCI Express devices or `pci` for PCI devices. *nexus-driver-name* is the driver name for the device component; *physical-slot#* is the hardware slot number; and *pci-device-number* is the PCI device number in standard PCI nomenclature.

First, an *absolute-slot-path* is constructed that attempts to describe the attachment point's topological location in more physically identifiable terms for the user. This *absolute-slot-path* consists of *slot-path* components each separated by a `:` (colon). The leaf or leftmost *slot-path* component describes the device of the attachment point itself, while its right-adjacent *slot-path* component up to the rightmost or topmost *slot-path* component describes the parent up to the root devices, respectively.

Each *slot-path* consists of a *slot-id* optionally preceded by a *fru-id*, which identifies an expansion chassis containing the device described by *slot-id* (detailed below). *fru-id* consists of *fru-type* followed by an optional *serialid#*. *fru-type* is "iob" for PCI Express expansion chassis types, while *serialid#* is either a 64-bit hexadecimal number indicating a raw serial number obtained from the expansion chassis hardware, or an upper-case, ASCII four-character sequence for a Sun-branded expansion chassis.

Each slot-id consists of one of three possible forms:

slot-id form (1)
slot-names

slot-id form (2)
device-type physical-slot#

slot-id form (3)
nexus-driver-name nexus-driver-instance device-type pci-device-number

The precedence of which form to select flows from the lowest form number to the highest form number, or from top to bottom as described above. If a form cannot be successfully constructed, then the next numerically higher form is attempted.

The *slot-names* in *slot-id* form (1) is taken from the `slot-names` property of the corresponding node in the evice tree and is a name assigned by hardware or the platform. This format is not predefined or established.

In *slot-id* form (2), *device-type* indicates the device type of the component's slot, and is either `pcie` for PCI Express or `pci` for PCI, while *physical-slot#*, taken from the `physical-slot#` property of its corresponding device node, indicates the hardware slot number of the component.

slot-id form (3) is used when all other forms cannot be successfully constructed, and is considered to be the default form. *nexus-driver-name* is the component's driver name; *nexus-driver-instance* is this driver's instance; *device-type* is the same as described in form (2); *pci-device-number* is the PCI device number as described and used for device configuration cycles in standard PCI nomenclature.

In summary of the *slot-path* component, expanding the optional FRU component that might precede it, *slot-path* will consist one of the following forms in order:

(1) [iob[*serialid#*].]
slot-names

(2) [iob[*serialid#*].]
device_type physical_slot#

(2) [iob[*serialid#*].]
nexus-driver-name nexus-driver-instance.

device_type pci-device-number

Lastly, the final form of the actual `ap_id` name used in `cfgadm` is decided as follows, specified in order of precedence:

`ap_id` form (1)
If the *absolute-slot-path* can fit within the fixed length limit of `cfgadm`'s `ap_id` field, then *absolute-slot-path* itself is used

`ap_id` form (2)
(*absolute-slot-path* exceeds the `ap_id` length limit) If the last *slot_path* component is contained within an expansion chassis, and it contains a *serialid#*, then the last *slot_path* component is used. The requirement for a *serialid#* in this form is to ensure a globally unique `ap_id`.

`ap_id` form (3)
(*absolute-slot-path* exceeds the `ap_id` length limit) The default form, *slot-id* form (3), of the last *slot_path* component is used.

Whichever final `ap_id` name is used, the *absolute-slot-path* is stored in the Information (`info`) field which can be displayed using the `-s` or `-v` options. This information can be used to physically locate any `ap_id`s named using `ap_id` form (2) or `ap_id` form (3). The *absolute-slot-path* is transformed slightly when stored in the information field, by the replacement of a colon (`:`) with forward slashes (`/`) to more closely denote a topological context. The *absolute-slot-path* can include *slot-path* components that are not hotpluggable above the leaf or rightmost *slot-path* component up to the onboard host slot.

See the Examples section for a list of hotpluggable examples.

Options The following options are supported:

`-c` *function*
The following functions are supported for PCI hotpluggable slots:

`configure`
Configure the PCI device in the slot to be used by Solaris.

`connect`
Connect the slot to PCI bus.

`disconnect`
Disconnect the slot from the PCI bus.

`insert`
Not supported.

`remove`
Not supported.

`unconfigure`
Logically remove the PCI device's resources from the system.

`-f`
Not supported.

`-h` *ap_id* | *ap_type*
Display PCI hotplug-specific help message.

`-l` *list*
List the values of PCI Hot Plug slots.

`-o` *hardware_options*
No hardware specific options are currently defined.

`-s` *listing_options*
Same as the generic `cfgadm(1M)`.

`-t` *ap_id*
This command is only supported on platforms that support testing capability on the slot.

`-v`
Execute in verbose mode.

When the `-v` option is used with the `-l` option, the `cfgadm` command outputs information about the attachment point. For attachment points located in a PCI Express hierarchy, the Information field will contain the attachment point's absolute slot path location, including any hardware– or platform-specific labeling information for each component in the slot path. Each component in the slot path will be separated by a `/` (forward slash). See "PCI Express `ap_id` Naming," above. For PCI Hot Plug attachment points not located in a PCI

Express hierarchy, see cfgadm_pci(1M). The information in the Type field is printed with or without the -v option. The occupant Type field will describe the contents of the slot. There are two possible values:

unknown
The slot is empty. If a card is in the slot, the card is not configured or there is no driver for the device on the card.

subclass/*board*
The card in the slot is either a single-function or multi-function device.

subclass is a string representing the subclass code of the device, for example, SCSI, ethernet, pci-isa, and so forth. If the card is a multi-functional device, MULT will get displayed instead.

board is a string representing the board type of the device. For example, hp is the string used for a PCI Hot Plug adapter.

-x *hardware_function*
Perform hardware-specific function. These hardware-specific functions should not normally change the state of a receptacle or occupant.

The following *hardware_function* is supported:

led=[*led_sub_arg*],mode=[*mode_sub_arg*]
Without subarguments, display a list of the current LED settings. With subarguments, set the mode of a specific LED for a slot.

Specify *led_sub_arg* as fault, power, attn, or active.

Specify *mode_sub_arg* as on, off, or blink.

For PCI Express, only the power and attn LEDs are valid and only the state of the attn LED can be changed.

Changing the state of the LED does not change the state of the receptacle or occupant. Normally, the LEDs are controlled by the hotplug controller, no user intervention is necessary. Use this command for testing purposes.

Caution – Changing the state of the LED can misrepresent the state of occupant or receptacle.

The following command displays the values of LEDs:

```
example# cfgadm -x led pcie2
Ap_Id              Led
pcie2     power=on,fault=off,active=off,attn=off
```

The following command sets the attn LED to blink to indicate the location of the slot:

```
example# cfgadm -x led=attn,mode=blink pcie2
```

Examples

EXAMPLE 1 Displaying the Value of Each Slot

The following command displays the values of each slot:

```
example# cfgadm -l
Ap_Id            Type         Receptacle   Occupant       Condition
c0               scsi-bus     connected    configured     unknown
c1               scsi-bus     connected    unconfigured   unknown
c2               scsi-bus     connected    unconfigured   unknown
pcie7            etherne/hp   connected    configured     ok
pcie8            unknown      empty        unconfigured   unknown
pcie9            fibre/hp     connected    configured     ok
```

EXAMPLE 2 Replacing a Card

The following command lists all DR-capable attachment points:

```
example# cfgadm
Type             Receptacle   Occupant     Condition
c0               scsi-bus     connected    configured     unknown
c1               scsi-bus     connected    unconfigured   unknown
c2               scsi-bus     connected    unconfigured   unknown
pcie7            etherne/hp   connected    configured     ok
pcie8            unknown      empty        unconfigured   unknown
pcie9            fibre/hp     connected    configured     ok
```

The following command unconfigures and electrically disconnects the card identified by `pcie7`:

```
example# cfgadm -c disconnect pcie7
```

The change can be verified by entering the following command:

```
example# cfgadm pcie7
Ap_Id     Type         Receptacle   Occupant     Condition
pcie7     unknown      disconnected unconfigured unknown
```

At this point the card can be swapped. The following command electrically connects and configures the replacement card:

```
example# cfgadm -c configure pcie7
```

The change can be verified by entering the following command:

```
example# cfgadm pcie7
Ap_Id     Type         Receptacle   Occupant     Condition
pcie7     etherne/hp   connected    configured   ok
```

EXAMPLE 3 Interpreting ApIds in a PCI Express Topology

The following command shows a listing for a topology with both PCI Express and PCI attachment points in an I/O expansion chassis connected to hotpluggable slots at the host level:

EXAMPLE 3 Interpreting ApIds in a PCI Express Topology *(Continued)*

```
example# cfgadm -s cols=ap_id:info
Ap_Id                          Information
iou#0-pci#0                    Location: iou#0-pci#0
iou#0-pci#1                    Location: iou#0-pci#1
iou#0-pci#1:iob.pci3           Location: iou#0-pci#1/iob.pci3
iou#0-pci#1:iob.pci4           Location: iou#0-pci#1/iob.pci4
iou#0-pci#2                    Location: iou#0-pci#2
iou#0-pci#2:iob58071.pcie1     Location: iou#0-pci#2/iob58071.pcie1
iou#0-pci#2:iob58071.special   Location: iou#0-pci#2/iob58071.special
iou#0-pci#3                    Location: iou#0-pci#3
iou#0-pci#3:iobBADF.pcie1      Location: iou#0-pci#3/iobBADF.pcie1
iou#0-pci#3:iobBADF.pcie2      Location: iou#0-pci#3/iobBADF.pcie2
iou#0-pci#3:iobBADF.pcie3      Location: iou#0-pci#3/iobBADF.pcie3
iou#0-pci#3:iobBADF.pci1       Location: iou#0-pci#3/iobBADF.pci1
iou#0-pci#3:iobBADF.pci2       Location: iou#0-pci#3/iobBADF.pci2
```

In this example, the `iou#0-pci#[0-3]` entries represents the topmost hotpluggable slots in the system. Because the `iou#`*n*`-pci#`*n* form does not match any of the forms stated in the grammar specification section described above, we can infer that such a name for the base component in this hotplug topology is derived from the platform through the `slot-names` property.

The slots in the preceding output are described as follows:

Slot `iou#0-pci#0`
: This slot is empty or its occupant is unconfigured.

Slot `iou#0-pci#1`
: This slot contains an expansion chassis with two hotpluggable slots, `pci3` and `pci4`. `pci3` and `pci4` represent two PCI slots contained within that expansion chassis with physical slot numbers `3` and `4`, respectively. The expansion chassis in this case does not have or export a serial-id.

Slot `iou#0-pci#2`
: This slot contains a third-party expansion chassis with a hexadecimal serial-id of `58071`. Within that expansion chassis are two hotpluggable slots, `pcie1` and `special`. `pcie1` represents a PCI Express slot with physical slot number `1`. The slot `special` has a label which is derived from the platform, hardware, or firmware.

Slot `iou#0-pci#3`
: This slot contains a Sun expansion chassis with an FRU identifier of `BADF`. This expansion chassis contains three PCI Express slots, `pcie1`, `pcie2`, and `pcie3` with physical slot numbers `1`, `2`, and `3`, respectively; and two PCI slots, `pci1` and `pci2`, with physical slot numbers 1 and 2, respectively.

The following command shows a listing for a topology with both PCI Express and PCI attachment points in an I/O expansion chassis with connected hotpluggable and non-hotpluggable host slots:

EXAMPLE 3 Interpreting ApIds in a PCI Express Topology *(Continued)*

```
example# cfgadm -s cols=ap_id:info
Ap_Id                          Information
Slot1                          Location: Slot1
Slot2:iob4ffa56.pcie1          Location: Slot2/iob4ffa56.pcie1
Slot2:iob4ffa56.pcie2          Location: Slot2/iob4ffa56.pcie2
Slot5:iob3901.pci1             Location: Slot2/iob3901.pci1
Slot5:iob3901.pci2             Location: Slot2/iob3901.pci2
```

In this example, the host system only has one hotpluggable slot, `Slot1`. We can infer that `Slot2` and `Slot5` are not hotpluggable slots because they do not appear as attachment points themselves in `cfgadm`. However, `Slot2` and `Slot5` each contains a third party expansion chassis with hotpluggable slots.

The following command shows a listing for a topology with attachment points that are lacking in certain device properties:

```
example# cfgadm -s cols=ap_id:info
Ap_Id                          Information
px_pci7.pcie0                  Location: px_pci7.pcie0
px_pci11.pcie0                 Location: px_pci11.pcie0
px_pci11.pcie0:iob.pcie1       Location: px_pci11.pcie0/iob.pcie1
px_pci11.pcie0:iob.pcie2       Location: px_pci11.pcie0/iob.pcie2
px_pci11.pcie0:iob.pcie3       Location: px_pci11.pcie0/iob.pcie3
```

In this example, the host system contains two hotpluggable slots, `px_pci7.pcie0` and `px_pci11.pcie0`. In this case, it uses `slot-id` form (3) (the default form) for the base *slot-path* component in the *absolute-slot-path*, because the framework could not obtain enough information to produce other more descriptive forms of higher precedence.

Interpreting right-to-left, attachment point `px_pci7.pcie0` represents a PCI Express slot with PCIdevice number `0` (which does not imply a physical slot number of the same number), bound to nexus driver `px_pci`, instance `7`. Likewise, attachment point `px_pci11.pcie0` represents a PCI Express slot with PCI device number `0` bound to driver instance `11` of `px_pci`.

Under `px_pci11.pcie0` is a third-party expansion chassis without a serial-id and with three hotpluggable PCI Express slots.

The following command shows a listing for a topology with attachment point paths exceeding the `ApId` field length limit:

```
example# cfgadm -s cols=ap_id:info
Ap_Id                          Information
pcie4                          Location: pcie4
pcie4:iobSUNW.pcie1            Location: pcie4/iobSUNW.pcie1
pcie4:iobSUNW.pcie2            Location: pcie4/iobSUNW.pcie2
```

EXAMPLE 3 Interpreting ApIds in a PCI Express Topology *(Continued)*

```
iob8879c3f3.pci1
                Location: pcie4/iobSUNW.pcie2/iob8879c3f3.pci1
iob8879c3f3.pci2
                Location: pcie4/iobSUNW.pcie2/iob8879c3f3.pci2
iob8879c3f3.pci3
                Location: pcie4/iobSUNW.pcie2/iob8879c3f3.pci3
```

In this example, there is only one hotpluggable slot, `pcie4` in the host. Connected under `pcie4` is a Sun expansion chassis with FRU identifier `SUNW`. Nested under PCI Express slot `pcie2` of that expansion chassis (`ApId pcie4:iobSUNW.pcie2`) lies another expansion chassis with three hotpluggable PCI slots.

Because the length of the absolute-slot-path form of:

```
pcie4/iobSUNW.pcie2/iob8879c3f3.pci1...3
```

...exceeds the `ApId` field length limit, and the leaf slot-path component is globally unique, `ap_id` form (2) is used, where the leaf *slot-path* component in the *absolute-slot-path* is used as the final `ApId`.

The following command shows a listing for a topology with attachment point paths exceeding the `ApId` field-length limit and lacking enough information to uniquely identify the leaf *slot-id* on its own (for example, missing the serial-id):

```
example# cfgadm -s cols=ap_id:info
Ap_Id                          Information
pcie4                          Location: pcie4
pcie4:iob4567812345678.pcie3   Location: pcie4/iob4567812345678.pcie3
px_pci20.pcie0
                Location: pcie4/iob4567812345678.pcie3/iob.pcie1
px_pci21.pcie0
                Location: pcie4/iob4567812345678.pcie3/iob.pcie2
```

In this example, there is only one hotpluggable slot, `pcie4` in the host. Connected under `pcie4` is a third-party expansion chassis with hexadecimal serial-id `4567812345678`. Nested under the PCI Express slot `pcie3` of that expansion chassis (`ApId pcie4:iob4567812345678.pcie3`), lies another third-party expansion chassis without a serial- id and with two hotpluggable PCI Express slots.

Because the length of the absolute-slot-path form of:

```
pcie4/iob4567812345678.pcie3/iob.pcie1...2
```

exceeds the `ApId` field length limit, and the leaf *slot-path* component is not globally unique, `ap_id` form (3) is used. `ap_id` form (2) is where *slot-id* form (3) (the default form) of the leaf *slot-path* component in the *absolute-slot-path* is used as the final `ApId`.

EXAMPLE 3 Interpreting ApIds in a PCI Express Topology *(Continued)*

The default form or `slot-id` form (3) of the leaf component `.../iob.pcie1` represents a PCI Express slot with device number `0`, bound to driver instance `20` of `px_pci`. Likewise, the default form of the leaf component `.../iob.pcie2` represents a PCI Express slot with device number `0`, bound to driver instance `21` of `px_pci`.

Files

`/usr/lib/cfgadm/shp.so.1`
Hardware-specific library for PCI Express and Standard PCI hotplugging.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/library
Interface Stability	Uncommitted

See Also

`cfgadm(1M)`, `cfgadm_pci(1M)`, `hotplugd(1M)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `attributes(5)`, `smf(5)`

Notes

The `cfgadm_shp` library is dependent on the `hotplug` service, which is managed by `smf(5)` under FMRI:

```
svc:/system/hotplug:default
```

The service must be enabled for the `cfgadm_shp` library to function properly. See `hotplugd(1M)` for details.

Name cfgadm_sysctrl – EXX00 system board administration

Synopsis

```
/usr/sbin/cfgadm -c function [-f]
     [-o disable-at-boot | enable-at-boot] [-n | -y] sysctrl0:slot# ...

/usr/sbin/cfgadm -x quiesce-test sysctrl0:slot#

/usr/sbin/cfgadm -x insert-test | remove-test sysctrl0:slot# ...

/usr/sbin/cfgadm -x set-condition-test=# sysctrl0:slot# ...

/usr/sbin/cfgadm [-l]
     -o disable-at-boot | enable-at-boot sysctrl0:slot# ...
```

Description The `sysctrl` hardware specific library `/usr/platform/sun4u/lib/cfgadm/sysctrl.so.1` provides dynamic reconfiguration functionality for configuring and disconnecting system boards on `E6X00`, `E5X00`, `E4X00`, and `E3X00` systems. You can insert both I/O and CPU boards into a slot on a running system that is configured for Solaris without rebooting. You can also disconnect and remove both types of boards from a running system without rebooting.

System slots appear as attachment points in the device tree, one attachment point for each actual slot in the system chassis. If a board is not in a slot, the receptacle state is `empty`. If a board is powered-off and ready to remove, the receptacle state is `disconnected`. If a board is powered-on and is connected to the system bus, the receptacle state is `connected`.

The occupant state is `unconfigured` when the receptacle state is `empty` or `disconnected`. The occupant state is either `unconfigured` or `configured` when the receptacle state is `connected`.

In the `configured` state the devices on a board are available for use by Solaris. In the `unconfigured` state, the devices on the board are not.

Inserting a board changes the receptacle state from `empty` to `disconnected`. Removing a board changes the receptacle state from `disconnected` to `empty`. Removing a board that is in the `connected` state crashes the operating system and can result in permanent damage to the system.

Options Refer to `cfgadm`(1M) for a more complete description options.

The following options are supported:

`-c` *function* Perform the state change function. Specify *function* as `connect`, `disconnect`, `configure` or `unconfigure`.

configure Change the occupant state to `configure`.

If the receptacle state is `disconnected`, the `configure` function first attempts to connect the receptacle. The `configure` function walks the OBP device tree created as

part of the `connect` function and creates the Solaris device tree nodes, attaching devices as required. For CPU/Memory boards, `configure` adds CPUs to the CPU list in the `powered-off` state. These are visible to the `psrinfo(1M)` and `psradm(1M)` commands. Two memory attachment points are published for CPU/memory boards. Use `mount(1M)` and`ifconfig(1M)` to use I/O devices on the new board. To use CPUs, use `psradm -n` to on-line the new processors. Use `cfgadm_ac(1M)` to test and configure the memory banks.

connect

Change the receptacle state to `connected`.

Changing the receptacle state requires that the system bus be frozen while the bus signals are connected and the board tested. The bus is frozen by running a `quiesce` operation which stops all process activity and suspends all drivers. Because the `quiesce` operation and the subsequent resume can be time consuming, and are not supported by all drivers, the `-x quiesce-test` is provided. While the system bus is frozen, the board being connected is tested by firmware. This operation takes a short time for I/O boards and a significant time for CPU/Memory boards due to CPU external cache testing. This does not provide memory testing. The user is prompted for confirmation before proceeding with the `quiesce`. Use the `-y` or `-n` option to override the prompt. The `connect` operation is refused if the board is marked as

`disabled-at-boot`, unless either the force flag, `-f`, or the enable at boot flag, `-o enable-at-boot`, is given. See `-l`.

disconnect — Change the receptacle state to `disconnected`.

If the occupant state is `configure`, the `disconnect` function first attempts to unconfigure the occupant. The `disconnect` operation does not require a `quiesce` operation and operates quickly. The board is powered-off ready for removal.

unconfigure — Change the occupant state to `unconfigureed`.

Devices on the board are made invisible to Solaris during this process. The I/O devices on an I/O board are removed from the Solaris device tree. Any device that is still in use stops the unconfigure process and be reported as in use. The unconfigure operation must be retried after the device is made non-busy. For CPU/Memory boards, the memory must have been changed to the unconfigured state prior to issuing the board unconfigure operation. The CPUs on the board are off-lined, powered off and removed from the Solaris CPU list. CPUs that have processes bound to them cannot be off-lined. See `psradm`(1M), `psrinfo(1M)`, `pbind(1M)`, and`p_online(2)` for more information on off-lining CPUs.

`-f` — Force a block on connecting a board marked as `disabled-at-boot` in the non-volatile

	disabled-board-list variable. See *Platform Notes:Sun Enterprise 6x00/5x00/4x00/3x00 Systems*
-l	List options. Supported as described in cfgadm(1M)cfgadm(1M). The *type* field can be one of cpu/mem, mem, dual-sbus, sbus-upa, dual-pci, soc+sbus, soc+upa, disk or unknown. The hardware-specific info field is set as follows: [disabled at boot] [non-detachable] [100 MHz capable] For sbus-upa and soc+upa type boards, the following additional information appears first: [single buffered ffb|double buffered ffb|no ffb installed] For disk type boards, the following additional information appears first: {target: # | no disk} {target: # | no disk}
-o disable-at-boot | enable-at-boot	Modify the state of the non—volatile disabled-board-list variable. Use this the -o option in conjunction with the -c *function* or -l option. Use -o enable-at-boot with the -c connect to override a block on connecting a disabled-at-boot board.
-x insert-test | remove-test	Perform a test. Specify remove-test to change the driver state for the specified slot from disconnected to empty without the need for physically removing the board during automated test sequences. Specify insert-test to change the driver state of a slot made to appear empty using the remove-test command to the disconnected state as if it had been inserted.
-x quiesce-test sysctrl0:slot1	Perform a test. Allows the quiesce operation required for board connect operations to be exercised. The execution of this test confirms that, with the current software and hardware configuration, it is possible to quiesce the

system. If a device or process cannot be quiesced, its name is printed in an error message. Any valid board attachment point can be used with this command, but since all systems have a slot1 the given form is recommended.

-x set-condition-test=# — Perform a test.

Allows the condition of a system board attachment point to be set for testing the policy logic for state change commands. The new setting is given as a number indicating one of the following condition values:

```
0       unknown
1       ok
2       failing
3       failed
4       unusable
```

Operands The following operand is supported:

sysctrl0:slot# — The attachment points for boards on `EXX00` systems are published by instance `0` of the `sysctrl` driver (`sysctrl0`). The names of the attachment points are numbered from `slot0` through `slot15`. Specify # as a number between `0` and `15`, indicating the slot number. This form conforms to the logical `ap_id` specification given in `cfgadm(1M)`. The corresponding physical `ap_id`s are listed in the `FILES` section.

Files /usr/platform/sun4u/lib/cfgadm/sysctrl.so.1
Hardware specific library

/devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot*
Attachment Points

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkvm.u

See Also `cfgadm(1M)`, `cfgadm_ac(1M)`, `ifconfig(1M)`, `mount(1M)`, `pbind(1M)`, `psradm(1M)`, `psrinfo(1M)`, `config_admin(3CFGADM)`, `attributes(5)`

Sun Enterprise 6x00, 5x00, 4x00 and 3x00 Systems Dynamic Reconfiguration User's Guide

Platform Notes:Sun Enterprise 6x00/5x00/4x00/3x00 Systems

Notes Refer to the *Sun Enterprise 6x00, 5x00, 4x00 and 3x00 Systems Dynamic Reconfiguration User's Guide* for additional details regarding dynamic reconfiguration of EXX00 system CPU/Memory boards.

Name cfgadm_usb – USB hardware-specific commands for cfgadm

Synopsis

```
/usr/sbin/cfgadm [-f] [-y | -n] [-v] -c function ap_id...

/usr/sbin/cfgadm -f [-y | -n] [-v] [-o hardware_options]
     -x hardware_function ap_id...

/usr/sbin/cfgadm -v [-a] [-s listing_option]
     [-l [ap_id | ap_type...]]

/usr/sbin/cfgadm -v -h [ap_id]...
```

Description The Universal Serial Bus (USB) hardware-specific library `/usr/lib/cfgadm/usb.so.1` provides the functionality for administering USB devices via the `cfgadm`(1M) command. `cfgadm` operates on attachment points. For details regarding attachment points, refer to `cfgadm`(1M).

For USB administration, the only attachment points supported are the ports of hubs attached to the USB bus.

Attachment points are named through attachment point IDs (*ap_ids*). The USB bus is hierarchical, so the *ap_ids* are as well. USB hubs have ports, numbered from `1` to *n*. All USB *ap_ids* consist of a string of the following form:

`usb`*N*`/`*A*`[.`*B*`[.`*C*`[...]]]`

where

N is the *N*th USB host controller on the system,
A is port #*A* on the root (top) hub.
B is port #*B* of the hub plugged into port #*A* of the hub above it.
C is port #*C* of the hub plugged into port #*B* of the hub above it, and so forth.

For example, the first port on the root hub of USB controller `0` (the only controller), has a logical *ap_id*:

```
usb0/1
```

Similarly, the second port on the first external hub plugged into the first port on the root hub of the first USB controller has a logical *ap_id*:

```
usb0/1.2
```

For example, if the *ap_id* is `usb0/1.4.3.4`, it represents port `4` of the hub plugged into port `3` of the hub plugged into port `4` of the hub plugged into port `1` of the root hub of the first USB host controller on the system.

```
example# cfgadm -l
Ap_Id                Type         Receptacle   Occupant     Condition
usb0/1               USB-hub      connected    configured   ok
usb0/2               unknown      empty        unconfigured ok
```

```
usb0/1.1             USB-storage  connected    configured   ok
usb0/1.2             unknown      empty        unconfigured ok
usb0/1.3             unknown      empty        unconfigured ok
usb0/1.4             USB-device   connected    configured   ok
```

USB2.0 chips have one EHCI host USB2.0 host controller and a number of companion USB 1.*x* host controllers (either OHCI or UHCI host controllers).

When a USB2.0 device has been plugged in, it shows up on the EHCI logical ports which might not have a 1 to 1 mapping to external physical port numbers on the system. When a USB1.*x* device is plugged in, the EHCI host controller reroutes the device to a companion host controller and the device shows up on the companion's logical port number.

The mapping of logical port numbers to physical port numbers can get quite complicated. For example:

```
% cfgadm
Ap_Id                Type         Receptacle   Occupant     Condition
c0                   scsi-bus     connected    configured   unknown
usb0/1               usb-mouse    connected    configured   ok
usb0/2               usb-kbd      connected    configured   ok
usb0/3               unknown      empty        unconfigured ok
usb0/4               usb-hub      connected    configured   ok
usb0/4.1             unknown      empty        unconfigured ok
usb0/4.2             unknown      empty        unconfigured ok
usb0/4.3             unknown      empty        unconfigured ok
usb0/4.4             usb-storage  connected    configured   ok
usb1/1               unknown      empty        unconfigured ok
usb1/2               unknown      empty        unconfigured ok
usb1/3               unknown      empty        unconfigured ok
usb2/1               unknown      empty        unconfigured ok
usb2/2               usb-device   connected    configured   ok
usb3/1               unknown      empty        unconfigured ok
usb3/2               unknown      empty        unconfigured ok
usb3/3               unknown      empty        unconfigured ok
usb3/4               unknown      empty        unconfigured ok
usb3/5               unknown      empty        unconfigured ok
```

In this example usb0 is the onboard USB 1.*x* host controller. usb1 and usb2 are companion OHCI USB1.*x* host controllers and usb3 is an EHCI USB2.0 host controller.

The following table shows the somewhat confusing routing for this USB2.0 chip:

```
logical port number        physical port number
-------------------        --------------------
    usb1/1            internal port 1
    usb1/2            external port 1
    usb1/3            external port 3
```

```
usb2/1            internal port 2
usb2/2            external port 2

usb3/1            internal port 1
usb3/2            internal port 2
usb3/3            external port 1
usb3/4            external port 2
usb3/5            external port 3
```

Unfortunately, the exact routing can often only be determined by experimentation.

The receptacle states for attachment points at the USB port have the following meanings:

`connected` USB port is powered on and enabled. A USB device is plugged in to the port. The device is logically connected to the USB bus.

`disconnected` USB port is powered on and enabled. A USB device is plugged into the port. The device has been logically disconnected from the USB bus (using the `cfgadm -c disconnect` command).

`empty` USB port is powered on, but no device is plugged in to it.

The occupant states for devices at USB port attachment points at the USB port have the following meanings:

`configured` The USB device at the USB port is configured and usable by Solaris.

`unconfigured` The USB device at the USB port was explicitly off-lined using `cfgadm -c unconfigure`, or was not successfully configured for use with Solaris, for example, having no driver or a device problem.

The attachment point conditions are:

`ok` Normal state - ready for use.

`failing` Not used.

`failed` Not used.

`unusable` The user has physically removed a device while an application had the device open (there might be outstanding `I/O`). Users need to reinsert the same physical device and close the application properly before removing the device again. The port cannot configure other inserted devices until this is done.

If the original device cannot be reinserted into the port, see the *System Administration Guide: Basic Administration* for instructions for clearing this attachment point condition.

`unknown` Not used.

A USB device can be hotplugged or hotunplugged at any time, and the system detects the event and takes the appropriate action.

It is not necessary to transition a receptacle to the `disconnected` state before removing its device from the USB. However, it is not recommended to hot-remove devices currently in use (such as removable disks currently opened by volume manager (see `vold(1M)`) or some other application).

Options `cfgadm` defines several types of operations. These operations include invoking configuration state changes (`-c`), invoking hardware-specific functions (`-x`), and obtaining configuration administration help messages (`-h`).

If any of these operations fail, the device and attachment point might not be in the expected state. Use the `cfgadm -l` command to display the device's current status.

All other options have the same meaning as defined in `cfgadm(1M)`.

The following options are supported:

`-c` *function* The following generic commands are defined for the USB hardware specific library. The following configuration state change operations are supported:

`configure` If there is a USB device plugged into the port, this command attempts to configure it and set everything up so that it is usable by Solaris. This command does an implied `connect` (reverse of `disconnect`) if necessary. This command accomplishes nothing, and returns an error message, if the device at that port is already configured. After successful execution of this command, the device is ready for use under Solaris.

`disconnect` Performs an `unconfigure` on the *ap_id* (if it is not already `unconfigured`), and then transitions the receptacle to the `disconnected` state, even though a device is still be plugged into the port. Issuing a `cfgadm -c configure`, or physically hotplugging the device, brings the device back to the `connected` receptacle state, and to the `configured` occupant state, assuming a driver can be found and there are no problems enumerating and configuring the device.

`unconfigure` Makes the device plugged into the port unusable by Solaris (offline it). If successful, `cfgadm` reports

this *ap_id*'s occupant state as *unconfigured*. Issuing a `configure` to the *ap_id* (if successful) brings its occupant back to the `configured` (online) condition, as it physically hotplugging the device on the port.

`-f` Not supported.

`-h` *ap_id* USB specific help can be obtained by using the help option with any USB attachment point.

`-l[v]` The `-l` option works as described in `cfgadm(1M)`. When paired with the `-v` option, the `Information` field contains the following USB-specific information:

- `Mfg`: manufacturer string (`iManufacturer`)
- `Product`: product string (`iProduct`)
- `NConfigs`: total number of configurations the device supports (`bNumConfigurations`).
- `Config`: current configuration setting in decimal (configuration index, not configuration value).
- The configuration string descriptor for the current configuration (`iConfiguration`)

See the Universal Serial Bus specification for a description of these fields.

`-o` *hardware_options* Hardware options are only supported for the hardware-specific command, `-x usb_config`. See the description of that command below for an explanation of the options available.

`-s` *listing_options* Attachment points of class USB can be listed by using the `select` sub-option. See `cfgadm(1M)`.

`-x` *hardware_function* The following hardware-specific functions are defined:

`usb_config -o config=`*n* This command requires the mandatory `config` value to be specified using the `-o` option.

Sets the USB configuration of a multi-configuration USB device at *ap_id* to configuration index *n*. The device is set to this configuration henceforth and this setting persists across reboots, hot-removes, and unconfigure/configure of the device.

	Valid values of *n* range from `0` to (`Nconfigs -1`). The device is reset by a `disconnect` followed by a `configure`. The `configure` causes the device to be configured to the new configuration setting. If any of these steps fail, the configuration file and the device are restored to their previous state and an error message is issued.
`usb_reset`	Performs a software reset (re-enumeration) of the device. This is the equivalent of removing the device and inserting it back again. The port on the hub is power cycled if the hub supports power cycling of individual ports. If the connected device is a hub, this function has the effect of resetting that hub and any devices down the tree of which it is the root. If any of these steps fail, the device is restored to its previous state and an error message is issued.

State table: attachment points state versus commands:

```
Valid states:
    empty/unconfigured          → no device connected

    disconnected/unconfigured   → logically disconnected,
                                  unavailable,
                                  devinfo node removed,
                                  device physically connected

    connected/unconfigured      → logically connected,
                                  unavailable,
                                  devinfo node present

    connected/configured        → connected, available
```

The table below clarifies the state transitions resulting from actions or commands:

```
current state      operation           new state
-------------      ---------           ---------
empty/
unconfigured:
               device plugged in:     connected/configured or
                                      connected/unconfigured
                                      (if enumeration failed)
               device removed:        n/a
               cfgadm -c unconfigure: empty/unconfigured
               cfgadm -c configure:   empty/unconfigured
               cfgadm -c disconnect:  empty/unconfigured
                                      (no-op and error)

disconnected/
unconfigured:
               device plugged in:     n/a
               device removed:        empty/unconfigured
               cfgadm -c unconfigure: disconnected/unconfigured
               cfgadm -c configure:   connected/configured, or
                                      connected/unconfigured
                                      (if reenumeration failed)
             cfgadm -c disconnect:    disconnected/unconfigured

connected/unconfigured:
             device plugged in:       n/a
             device removed:          empty/unconfigured
             cfgadm -c unconfigure:   connected/unconfigured
             cfgadm -c configure:     connected/configured, or
                                      connected/unconfigured
                                      (if reenumeration failed)
             cfgadm -c disconnect:    disconnected/unconfigured

connected/configured:
             device plugged in:       n/a
             device removed:          empty/unconfigured or
                                      connected/configured,
                                      but with ap condition
                                      'unusable' if device
                                      was open when removed
             cfgadm -c unconfigure:   connected/unconfigured
             cfgadm -c configure:     connected/configured
             cfgadm -c disconnect:    disconnected/unconfigured
```

Examples

EXAMPLE 1 Listing the Status of All USB Devices

The following command lists the status of all USB devices on the system:

EXAMPLE 1 Listing the Status of All USB Devices *(Continued)*

```
# cfgadm
Ap_Id           Type         Receptacle   Occupant     Condition
usb0/1          USB-hub      connected    configured   ok
usb0/2          unknown      empty        unconfigured ok
usb0/1.1        USB-storage  connected    configured   ok
usb0/1.2        unknown      empty        unconfigured ok
usb0/1.3        unknown      empty        unconfigured ok
usb0/1.4        USB-device connected    configured   ok
```

Notice that `cfgadm` treats the `USB-device` device at `ap_id` `usb0/1.4` as a single unit, since it cannot currently control individual interfaces.

EXAMPLE 2 Listing the Status of a Port with No Device Plugged In

The following command lists the status of a port with no device plugged in:

```
example# cfgadm -l usb0/1.3
Ap_Id           Type         Receptacle   Occupant     Condition
usb0/1.3        unknown      empty        unconfigured ok
```

EXAMPLE 3 Listing the Status of the Same Port with a Device Plugged In

The following command lists the status of the same port after physically plugging in a device that configures without problems:

```
example# cfgadm -l usb0/1.3
Ap_Id           Type         Receptacle   Occupant     Condition
usb0/1.3        USB-hub      connected    configured   ok
```

EXAMPLE 4 Unconfiguring an Existing USB Device

The following command unconfigures the USB device attached to `usb0/1.3`, then displays the status of the `ap_id`:

```
example# cfgadm -c unconfigure usb0/1.3
Unconfigure the device: /devices/pci@0,0/pci8086,7112@7,2/hub@2:2.3
This operation suspends activity on the USB device
Continue (yes/no)?

Enter:

y

example# cfgadm -l usb0/1.3
Ap_Id           Type         Receptacle   Occupant     Condition
usb0/1.3        unknown      connected    unconfigured ok
```

EXAMPLE 5 Unconfiguring and Logically Disconnecting an Existing USB Device

The following command unconfigures and logically disconnects a USB device attached to `usb0/1.3`:

```
example# cfgadm -c disconnect usb0/1.3
Disconnect the device: /devices/pci@0,0/pci8086,7112@7,2/hub@2:2.3
This operation suspends activity on the USB device
Continue (yes/no)?

Enter:

y

example# cfgadm -l usb0/1.3
Ap_Id          Type         Receptacle     Occupant        Condition
usb0/1.3       unknown      disconnected   unconfigured    ok
```

A `disconnect` implies that `cfgadm` does an `unconfigure` first. The receptacle status now shows `disconnected`, even though the device is still physically connected. In this case, a physical hotplug or using the `cfgadm -c configure` on the `ap_id` brings it back on-line.

EXAMPLE 6 Configuring a Previously Unconfigured USB Device

The following command configures a USB device that was previously attached to `usb0/1.3`:

```
example # cfgadm -yc configure usb0/1.3
example# cfgadm -l usb0/1.3
Ap_Id           Type         Receptacle   Occupant     Condition
usb0/1.3        unknown      connected    configured   ok
```

EXAMPLE 7 Resetting a USB Device

The following command resets a USB device:

```
example# cfgadm -x usb_reset usb0/1.3
Reset the device: /devices/pci@0,0/pci8086,7112@7,2/hub@2:2.3
This operation suspends activity on the USB device
Continue (yes/no)?

Enter:

y
```

EXAMPLE 8 Displaying Detailed Information About a USB Device

The following command displays detailed information about a USB device. This device shows the following USB-specific information in the '`Information`' field:

- Manufacturer string: Iomega

EXAMPLE 8 Displaying Detailed Information About a USB Device *(Continued)*

- Product string: `USB Zip 250`
- Number of configurations supported: 1
- Configuration currently active: 0
- Configuration string descriptor for configuration 0: Default

```
example# cfgadm -lv  usb0/1.5
Ap_Id                 Receptacle   Occupant     Condition  Information
When         Type         Busy         Phys_Id
usb0/1.5     connected    configured   ok         Mfg:"Io
mega"  Product:"USB Zip 250"  NConfigs:1  Config:0 : Default

example# cfgadm -l -s "cols=ap_id:info" usb0/1.5
Ap_Id                        Information
usb0/1.5                     Mfg:"Iomega"  Product:"USB Zip 250"
NConfigs:1  Config:0 : Default
```

EXAMPLE 9 Displaying Detailed Information About All USB Devices

The following command displays detailed information about all USB devices on the system:

```
example# cfgadm -l -s "select=class(usb),cols=ap_id:info"
Ap_Id                        Information
usb0/1                       Mfg:<undefined>  Product:<undefined>
NConfigs:1  Config:0 <no cfg str descr>
usb0/2
usb0/1.1                     Mfg:<undefined>  Product:<undefined>
NConfigs:1  Config:0 <no cfg str descr>
usb0/1.2
usb0/1.3
usb0/1.4                     Mfg:"Wizard"  Product:"Modem/ISDN"
NConfigs:3  Config:1 : V.90 Analog Modem
usb0/1.5                     Mfg:"Iomega"  Product:"USB Zip 250"
NConfigs:1  Config:0 : Default
usb0/1.6                     Mfg:"SOLID YEAR"  Product:"SOLID YEAR
USB"NConfigs:1  Config:0 <no cfg str descr>
usb0/1.7
```

Lines containing only an `ap_id` are empty ports. These can be filtered out. This example only lists USB `ap_id`s with connected devices, and information about those devices.

```
example# cfgadm -l -s "select=class(usb),cols=ap_id:info" | grep Mfg
usb0/1                       Mfg:<undefined>  Product:<undefined>
NConfigs:1  Config:0 <no cfg str descr>
usb0/1.1                     Mfg:<undefined>  Product:<undefined>
NConfigs:1  Config:0 <no cfg str descr>
usb0/1.4                     Mfg:"Wizard"  Product:"Modem/ISDN"
NConfigs:3  Config:1 : V.90 Analog Modem
```

EXAMPLE 9 Displaying Detailed Information About All USB Devices *(Continued)*

```
usb0/1.5                      Mfg:"Iomega"  Product:"USB Zip 250"
NConfigs:1  Config:0 : Default
usb0/1.6                      Mfg:"SOLID YEAR"  Product:"SOLID YEAR USB"
Config:0 <no cfg str descr>
```

EXAMPLE 10 Listing Information About a Multi-configuration USB Device

The following example lists information about a multi-configuration USB device.

Notice the `NConfigs` field: the configurations available for this device are `0`, `1`, and `2` (`0` to (*N*`Configs-1`)).

```
example# cfgadm -l -s "cols=ap_id:info" usb0/1.4
Ap_Id                         Information
usb0/1.4                      Mfg:"Wizard"  Product:"Modem/ISDN"
NConfigs:3  Config:1 V.90 Analog Modem"
```

EXAMPLE 11 Setting the Current Configuration of a Multi-configuration USB Device

The following example sets the current configuration of a multi-configuration USB device:

```
example# cfgadm -o config=2 -x usb_config usb0/1.4
Setting the device: /devices/pci@1f,2000/usb@1/device@3
to USB configuration 2
This operation suspends activity on the USB device
Continue (yes/no)?

Enter:

y

USB configuration changed successfully.
```

The device path should be checked to ensure that the right instance of a device is being referred to, in the case where multiple devices of the exact same type are on the same bus. This information is available in the '`Information`' field.

Files

`/usr/lib/cfgadm/usb.so.1` Hardware specific library for generic USB device administration

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl

See Also cfgadm(1M), vold(1M), config_admin(3CFGADM), attributes(5), scsa2usb(7D), usba(7D)

Universal Serial Bus 1.1 Specification (www.usb.org)

System Administration Guide: Basic Administration

Notes cfgadm(1M) can not unconfigure, disconnect, reset, or change the configuration of any USB device currently opened by vold(1M) or any other application. These operations also fail on a hub if a device in its hierarchy is opened by an application. See scsa2usb(7D) for unconfiguring a USB mass-storage device that is being used by vold(1M).

Only super-users can execute any functions on an attachment point. However, one need not be a super-user to list the attachment points.

Name cfsadmin – administer disk space used for caching file systems with the Cache File-System (CacheFS)

Synopsis

```
cfsadmin -c [-o cacheFS-parameters] cache_directory

cfsadmin -d {cache_ID | all} cache_directory

cfsadmin -l cache_directory

cfsadmin -s {mntpt1 ....} | all

cfsadmin -u [-o cacheFS-parameters] cache_directory
```

Description The `cfsadmin` command provides the following functions:

- cache creation
- deletion of cached file systems
- listing of cache contents and statistics
- resource parameter adjustment when the file system is unmounted.

You must always supply an option for `cfsadmin`. For each form of the command except `-s`, you must specify a cache directory, that is, the directory under which the cache is actually stored. A path name in the front file system identifies the cache directory. For the `-s` form of the command, you must specify a mount point.

You can specify a cache ID when you mount a file system with CacheFS, or you can let the system generate one for you. The `-l` option includes the cache ID in its listing of information. You must know the cache ID to delete a cached file system.

Options The following options are supported:

`-c` [`-o` *cacheFS-parameters*] *cache_directory*	Create a cache under the directory specified by *cache_directory*. This directory must not exist prior to cache creation.
`-d` { *cache_ID* | all } *cache_directory*	Remove the file system whose cache ID you specify and release its resources, or remove all file systems in the cache by specifying `all`. After deleting a file system from the cache, you must run the `fsck_cachefs(1M)` command to correct the resource counts for the cache. As indicated by the syntax above, you must supply either a *cache_ID* or `all`, in addition to *cache_directory*.
`-l` *cache_directory*	List file systems stored in the specified cache, as well as statistics about them. Each cached

file system is listed by cache ID. The statistics document resource utilization and cache resource parameters.

`-s` { *mntpt1* ... } | all

Request a consistency check on the specified file system (or all `cachefs` mounted file systems). The `-s` option only works if the cache file system was mounted with `demandconst` enabled (see `mount_cachefs(1M)`). Each file in the specified cache file system is checked for consistency with its corresponding file in the back file system. Note that the consistency check is performed file by file as files are accessed. If no files are accessed, no checks are performed. Use of this option does not result in a sudden "storm" of consistency checks.

As indicated by the syntax above, you must supply one or more mount points, or `all`.

`-u` [`-o` *cacheFS-parameters*] *cache_directory*

Update resource parameters of the specified cache directory. Parameter values can only be increased. To decrease the values, you must remove the cache and recreate it. All file systems in the cache directory must be unmounted when you use this option. Changes take effect the next time you mount any file system in the specified cache directory. The `-u` option with no `-o` option sets all parameters to their default values.

CacheFS Resource Parameters

You can specify the following CacheFS resource parameters as arguments to the `-o` option. Separate multiple parameters with commas.

`maxblocks=`*n*

Maximum amount of storage space that CacheFS can use, expressed as a percentage of the total number of blocks in the front file system. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the space the `maxblocks` parameter allows is available. The default is `90`.

`minblocks=`*n*

Minimum amount of storage space, expressed as a percentage of the total number of blocks in the front file system, that CacheFS is always allowed to use without limitation by its internal control mechanisms. If

CacheFS does not have exclusive use of the front file system, there is no guarantee that all the space the `minblocks` parameter attempts to reserve is available. The default is `0`.

`threshblocks=`*n* — A percentage of the total blocks in the front file system beyond which CacheFS cannot claim resources once its block usage has reached the level specified by `minblocks`. The default is `85`.

`maxfiles=`*n* — Maximum number of files that CacheFS can use, expressed as a percentage of the total number of inodes in the front file system. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the inodes the `maxfiles` parameter allows is available. The default is `90`.

`minfiles=`*n* — Minimum number of files, expressed as a percentage of the total number of inodes in the front file system, that CacheFS is always allowed to use without limitation by its internal control mechanisms. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the inodes the `minfiles` parameter attempts to reserve is available. The default is `0`.

`threshfiles=`*n* — A percentage of the total inodes in the front file system beyond which CacheFS cannot claim inodes once its usage has reached the level specified by `minfiles`. The default is `85`.

`maxfilesize=`*n* — Largest file size, expressed in megabytes, that CacheFS is allowed to cache. The default is `3`. You cannot decrease the block or inode allotment for a cache. To decrease the size of a cache, you must remove it and create it again with different parameters.

Currently `maxfilesize` is ignored by `cachefs`, therefore, setting it has no effect.

Operands

cache_directory — The directory under which the cache is actually stored.

mntpt1 — The directory where the CacheFS is mounted.

Usage

See `largefile(5)` for the description of the behavior of `cfsadmin` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples

EXAMPLE 1 Creating a Cache Directory

The following example creates a cache directory named `/cache`:

```
example# cfsadmin -c /cache
```

EXAMPLE 2 Creating a Cache

The following example creates a cache named `/cache1` that can claim a maximum of 60 percent of the blocks in the front file system, can use 40 percent of the front file system blocks without interference by CacheFS internal control mechanisms, and has a threshold value of 50 percent. The threshold value indicates that after CacheFS reaches its guaranteed minimum, it cannot claim more space if 50 percent of the blocks in the front file system are already used.

```
example# cfsadmin -c -o maxblocks=60,minblocks=40,threshblocks=50 /cache1
```

EXAMPLE 3 Changing the `maxfilesize` Parameter

The following example changes the `maxfilesize` parameter for the cache directory `/cache2` to 2 megabytes:

```
example# cfsadmin -u -o maxfilesize=2 /cache2
```

EXAMPLE 4 Listing the Contents of a Cache Directory

The following example lists the contents of a cache directory named `/cache3` and provides statistics about resource utilization:

```
example# cfsadmin -l /cache3
```

EXAMPLE 5 Removing a Cached File System

The following example removes the cached file system with cache ID `23` from the cache directory `/cache3` and frees its resources (the cache ID is part of the information returned by `cfsadmin -l`):

```
example# cfsadmin -d 23 /cache3
```

EXAMPLE 6 Removing All Cached File Systems

The following example removes all cached file systems from the cache directory `/cache3`:

```
example# cfsadmin -d all /cache3
```

EXAMPLE 7 Checking for Consistency in File Systems

The following example checks for consistency all file systems mounted with `demandconst` enabled. No errors are reported if no `demandconst` file systems were found.

```
example# cfsadmin -s all
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also cachefslog(1M), cachefsstat(1M), cachefswssize(1M), fsck_cachefs(1M), mount_cachefs(1M), attributes(5), largefile(5)

Name chat – automated conversational exchange tool

Synopsis `chat [options] script`

Description The `chat` program implements a conversational text-based exchange between the computer and any serial device, including (but not limited to) a modem, an ISDN TA, and the remote peer itself, establishing a connection between the Point-To-Point Protocol daemon (`pppd`) and the remote `pppd` process.

Options The `chat` command supports the following options:

`-f` <*chat file*> Read the `chat` script from the `chat` file. This option is mutually exclusive with the `chat` script parameters. You must have `read` access to use the file. Multiple lines are permitted in the file. Use the space or horizontal tab characters to separate the strings.

`-t` <*timeout*> Set the timeout for the expected string to be received. If the string is not received within the time limit, the reply string is not sent. If specified, a 'subexpect' (alternate reply) string can be sent. Otherwise, if no alternate reply strings remain, the `chat` script fails.. A failed script will cause the `chat` program to terminate with a non-zero error code.

`-r` <*report file*> Set the file for output of the report strings. If you use the keyword `REPORT`, the resulting strings are written to this file. If the `-r` option is not used and you use the `REPORT` keyword, the `stderr` file is used for the report strings.

`-e` Start with the echo option turned on. You turn echo on or off at specific points in the `chat` script using the `ECHO` keyword. When echoing is enabled, all output from the modem is echoed to `stderr`.

`-E` Enables environment variable substitution within `chat` scripts using the standard *$xxx* syntax.

`-v` Request that the `chat` script execute in a verbose mode. The `chat` program logs the execution state of the `chat` script as well as all text received from the modem and output strings sent to the modem. The default is to log through `syslog(3C)` with facility `local2`; the logging method is alterable using the `-S` and `-s` options.

`-V` Request that the `chat` script be executed in a `stderr` verbose mode. The `chat` program logs all text received from the modem and output strings sent to the modem to `stderr`. `stderr` is usually the local console at the station running the `chat` or `pppd` program.

`-s` Use `stderr`. Log messages from `-v` and error messages are sent to `stderr`.

`-S`	Do not use syslog. By default, error messages are set to syslog. This option prevents log messages from `-v` and error messages from being sent to syslog.
`-T` <*phone number*>	Pass in an arbitrary string (usually a telephone number) that will be substituted for the `\T` substitution metacharacter in a send string.
`-U` <*phone number 2*>	Pass in a second string (usually a telephone number) that will be substituted for the `\U` substitution metacharacter in a send string. This is useful when dialing an ISDN terminal adapter that requires two numbers.
`script`	If the script is not specified in a file with the `-f` option, the script is included as parameters to the `chat` program.

Extended Description

Chat Script

The `chat` script defines communications. A script consists of one or more "expect-send" pairs of strings separated by spaces, with an optional "subexpect-subsend" string pair, separated by a dash (as in the following example:)

ogin:-BREAK-ogin: ppp ssword: hello2u2

The example indicates that the `chat` program should expect the string "ogin:". If it fails to receive a login prompt within the time interval allotted, it sends a break sequence to the remote and then expects the string "ogin:". If the first "ogin:" is received, the break sequence is not generated.

Upon receiving the login prompt, the `chat` program sends the string ”ppp” and then expects the prompt "ssword:". When the password prompt is received, it sends the password hello2u2.

A carriage return is normally sent following the reply string. It is not expected in the "expect" string unless it is specifically requested by using the `\r` character sequence.

The expect sequence should contain only what is needed to identify the received data. Because it's stored on a disk file, it should not contain variable information. Generally it is not acceptable to look for time strings, network identification strings, or other variable pieces of data as an expect string.

To correct for characters that are corrupted during the initial sequence, look for the string "ogin:" rather than "login:". The leading "l" character may be received in error, creating problems in finding the string. For this reason, scripts look for "ogin:" rather than "login:" and "ssword:" rather than "password:".

An example of a simple script follows:

```
ogin: ppp ssword: hello2u2
```

The example can be intrepreted as: expect ogin:, send ppp, expect ...ssword:, send hello2u2.

When login to a remote peer is necessary, simple scripts are rare. At minimum, you should include sub-expect sequences in case the original string is not received. For example, consider the following script:

```
ogin:--ogin: ppp ssword: hello2u2
```

This script is more effective than the simple one used earlier. The string looks for the same login prompt; however, if one is not received, a single return sequence is sent and then the script looks for login: again. If line noise obscures the first login prompt, send the empty line to generate a login prompt again.

Comments

Comments can be embedded in the `chat` script. Comment lines are ignored by the chat program. A comment starts with the hash ("#") character in column one. If a # character is expected as the first character of the expect sequence, quote the expect string. If you want to wait for a prompt that starts with a # character, write something like this:

```
# Now wait for the prompt and send logout string
'# ' logout
```

Sending Data From A File

If the string to send begins with an at sign ("@"), the remainder of the string is interpreted as the name of the file that contains the string. If the last character of the data read is a newline, it is removed. The file can be a named pipe (or fifo) instead of a regular file. This enables `chat` to communicate with another program, for example, a program to prompt the user and receive a password typed in.

Abort

Many modems report the status of a call as a string. These status strings are often "CONNECTED" or "NO CARRIER" or "BUSY." If the modem fails to connect to the remote, you can terminate the script. Abort strings may be specified in the script using the ABORT sequence. For example:

```
ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ATDT5551212 CONNECT
```

This sequence expects nothing and sends the string ATZ. The expected response is the string OK. When OK is received, the string ATDT5551212 dials the telephone. The expected string is CONNECT. If CONNECT is received, the remainder of the script is executed. When the modem finds a busy telephone, it sends the string BUSY, causing the string to match the abort character sequence. The script fails because it found a match to the abort string. If the NO CARRIER string is received, it aborts for the same reason.

Clr_Abort

The CLR_ABORT sequence clears previously set ABORT strings. ABORT strings are kept in an array of a pre-determined size; CLR_ABORT reclaims the space for cleared entries, enabling new strings to use that space.

Say The SAY string enables the script to send strings to a user at a terminal via standard error. If `chat` is being run by `pppd` and `pppd` is running as a daemon (detached from its controlling terminal), standard error is normally redirected to the `/etc/ppp/connect-errors` file.

SAY strings must be enclosed in single or double quotes. If carriage return and line feed are required for the output, you must explicitly add them to your string.

The SAY string can provide progress messages to users even with "ECHO OFF." For example, add a line similar to the following to the script:

```
ABORT BUSY
ECHO OFF
SAY "Dialing your ISP...\n"
'' ATDT5551212
TIMEOUT 120
SAY "Waiting up to 2 minutes for connection ..."
CONNECT ''
SAY "Connected, now logging in ...\n"
ogin: account
ssword: pass
$ \c
SAY "Logged in OK ... \n"
```

This sequence hides script detail while presenting the SAY string to the user. In this case, you will see:

```
Dialing your ISP...
Waiting up to 2 minutes for connection...Connected, now logging in...
Logged in OK ...
```

Report REPORT is similar to the ABORT string. With REPORT, however, strings and all characters to the next control character (such as a carriage return), are written to the report file.

REPORT strings can be used to isolate a modem's transmission rate from its CONNECT string and return the value to the `chat` user. Analysis of the REPORT string logic occurs in conjunction with other string processing, such as looking for the expect string. It's possible to use the same string for a REPORT and ABORT sequence, but probably not useful.

Report strings may be specified in the script using the REPORT sequence. For example:

```
REPORT CONNECT
ABORT BUSY
ATDT5551212 CONNECT
ogin: account
```

The above sequence expects nothing, then sends the string ATDT5551212 to dial the telephone. The expected string is CONNECT. If CONNECT is received, the remainder of the

script is executed. In addition, the program writes the string CONNECT to the report file (specified by `-r`) in addition to any characters that follow.

Clr_Report CLR_REPORT clears previously set REPORT strings. REPORT strings are kept in an array of a pre-determined size; CLR_REPORT reclaims the space for cleared entries so that new strings can use that space.

Echo ECHO determines if modem output is echoed to `stderr`. This option may be set with the `-e` option, but can also be controlled by the ECHO keyword. The "expect-send" pair ECHO ON enables echoing, and ECHO OFF disables it. With ECHO, you can select which parts of the conversation should be visible. In the following script:

```
ABORT   'BUSY'
ABORT   'NO CARRIER'
""    AT&F
OK\r\n  ATD1234567
\r\n    \c
ECHO    ON
CONNECT \c
ogin:   account
```

All output resulting from modem configuration and dialing is not visible, but output is echoed beginning with the CONNECT (or BUSY) message.

Hangup The HANGUP option determines if a modem hangup is considered as an error. HANGUP is useful for dialing systems that hang up and call your system back. HANGUP can be ON or OFF. When HANGUP is set to OFF and the modem hangs up (for example, following the first stage of logging in to a callback system), `chat` continues running the script (for example, waiting for the incoming call and second stage login prompt). When the incoming call is connected, use the HANGUP ON string to reinstall normal hang up signal behavior. An example of a simple script follows:

```
ABORT   'BUSY'
 ""  AT&F
 OK\r\n  ATD1234567
 \r\n    \c
 CONNECT \c
 'Callback login:' call_back_ID
 HANGUP OFF
 ABORT "Bad Login"
 'Callback Password:' Call_back_password
 TIMEOUT 120
 CONNECT \c
 HANGUP ON
 ABORT "NO CARRIER"
 ogin:--BREAK--ogin: real_account
```

Timeout The initial timeout value is 45 seconds. Use the `-t` parameter to change the intial timeout value.

To change the timeout value for the next expect string, the following example can be used:

```
''"AT&F
 OK ATDT5551212
 CONNECT \c
 TIMEOUT 10
 ogin:--ogin: username
 TIMEOUT 5
 assword: hello2u2
```

The example changes the timeout to ten seconds when it expects the login: prompt. The timeout is changed to five seconds when it looks for the password prompt.

Once changed, the timeout value remains in effect until it is changed again.

EOT The EOT special reply string instructs the `chat` program to send an EOT character to the remote. This is equivalent to using ^D\c as the reply string. The EOT string normally indicates the end-of-file character sequence. A return character is not sent following the EOT. The EOT sequence can embedded into the send string using the sequence ^D.

BREAK The BREAK special reply string sends a break condition. The break is a special transmitter signal. Many UNIX systems handle break by cycling through available bit rates, and sending break is often needed when the remote system does not support autobaud. BREAK is equivalent to using \K\c as the reply string. You embed the break sequence into the send string using the \K sequence.

Escape Sequences Expect and reply strings can contain escape sequences. Reply strings accept all escape sequences, while expect strings accept most sequences. A list of escape sequences is presented below. Sequences that are not accepted by expect strings are indicated.

`''` Expects or sends a null string. If you send a null string, `chat` sends the return character. If you expect a null string, `chat` proceeds to the reply string without waiting. This sequence can be a pair of apostrophes or quote mark characters.

`\b` Represents a backspace character.

`\c` Suppresses the newline at the end of the reply string. This is the only method to send a string without a trailing return character. This sequence must be at the end of the send string. For example, the sequence hello\c will simply send the characters h, e, l, l, o. (Not valid in expect.)

`\d` Delay for one second. The program uses `sleep(1)` which delays to a maximum of one second. (Not valid in expect.)

`\K` Insert a BREAK. (Not valid in expect.)

`\n`	Send a newline or linefeed character.
`\N`	Send a null character. The same sequence may be represented by \0. (Not valid in expect.)
`\p`	Pause for 1/10th of a second. (Not valid in expect.)
`\q`	Suppress writing the string to syslog. The string `??????` is written to the log in its place. (Not valid in expect.)
`\r`	Send or expect a carriage return.
`\s`	Represents a space character in the string. Can be used when it is not desirable to quote the strings which contains spaces. The sequence 'HI TIM' and HI\sTIM are the same.
`\t`	Send or expect a tab character.
`\T`	Send the phone number string as specified with the `-T` option. (Not valid in expect.)
`\U`	Send the phone number 2 string as specified with the `-U` option. (Not valid in expect.)
`\\`	Send or expect a backslash character.
`\ddd`	Collapse the octal digits (ddd) into a single ASCII character and send that character. (\000 is not valid in an expect string.)
`^C`	Substitute the sequence with the control character represented by C. For example, the character DC1 (17) is shown as ^Q. (Some characters are not valid in expect.)

Environment Variables Environment variables are available within `chat` scripts if the `-E` option is specified on the command line. The metacharacter `$` introduces the name of the environment variable to substitute. If the substition fails because the requested environment variable is not set, nothing is replaced for the variable.

Exit Status The `chat` program terminates with the following completion codes:

`0`	Normal program termination. Indicates that the script was executed without error to normal conclusion.
`1`	One or more of the parameters are invalid or an expect string was too large for the internal buffers. Indicates that the program was not properly executed.
`2`	An error occurred during the execution of the program. This may be due to a read or write operation failing or `chat` receiving a signal such as SIGINT.
`3`	A timeout event occurred when there was an expect string without having a "-subsend" string. This indicates that you may not have programmed the script correctly for the condition or that an unexpected event occurred and the expected string could not be found.

4 The first string marked as an ABORT condition occurred.

5 The second string marked as an ABORT condition occurred.

6 The third string marked as an ABORT condition occurred.

7 The fourth string marked as an ABORT condition occurred.

... The other termination codes are also strings marked as an ABORT condition.

To determine which event terminated the script, use the termination code. It is possible to decide if the string "BUSY" was received from the modem versus "NO DIALTONE." While the first event may be retried, the second probably will not succeed during a retry.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpppdu
Interface Stability	Evolving

See Also `sleep(1)`, `uucp(1C)`, `pppd(1M)`, `uucico(1M)`, `syslog(3C)`, `attributes(5)`

Additional information on `chat` scripts are available with UUCP documentation. The `chat` script format was taken from scripts used by the `uucico` program.

Name check-hostname – check if sendmail can determine the system's fully-qualified host name

Synopsis

```
/usr/sbin/check-hostname
```

Description The `check-hostname` script is a migration aid for `sendmail(1M)`. This script tries to determine the local host's fully-qualified host name (FQHN) in a manner similar to `sendmail(1M)`. If `check-hostname` is able to determine the FQHN of the local host, it reports success. Otherwise, `check-hostname` reports how to reconfigure the system so that the FQHN can be properly determined.

Files

`/etc/hosts`	Host name database
`/etc/nsswitch.conf`	Name service switch configuration file
`/etc/resolv.conf`	Configuration file for name server routines

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsndmu
Interface Stability	Evolving

See Also `domainname(1M)`, `sendmail(1M)`, `hosts(4)`, `attributes(5)`

Name check-permissions – check permissions on mail rerouting files

Synopsis

```
/usr/sbin/check-permissions [login]
```

Description The `check-permissions` script is intended as a migration aid for `sendmail(1M)`. It checks the `/etc/mail/sendmail.cf` file for all configured alias files, and checks the alias files for `:include:` files. It also checks for certain `.forward` files. For each file that `check-permissions` checks, it verifies that none of the parent directories are group- or world-writable. If any directories are overly permissive, it is reported. Otherwise it reports that no unsafe directories were found.

As to which `.forward` files are checked, it depends on the arguments included on the command line. If no argument is given, the current user's home directory is checked for the presence of a `.forward` file. If any arguments are given, they are assumed to be valid logins, and the home directory of each one is checked.

If the special argument `ALL` is given, the `passwd` entry in the `/etc/nsswitch.conf` file is checked, and all password entries that can be obtained through the switch file are checked. In large domains, this can be time-consuming.

Operands The following operands are supported:

login Where *login* is a valid user name, checks the home directory for *login*.

`ALL` Checks the home directory of *all* users.

Files

`/etc/mail/sendmail.cf` Defines environment for `sendmail`

`/etc/mail/aliases` Ascii mail aliases file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsndmu
Interface Stability	Evolving

See Also `getent(1M)`, `sendmail(1M)`, `aliases(4)`, `attributes(5)`

Name chk_encodings – check the label encodings file syntax

Synopsis

```
/usr/sbin/chk_encodings [-a] [-c maxclass] [pathname]
```

Description `chk_encodings` checks the syntax of the label-encodings file that is specified by *pathname*. With the `-a` option, `chk_encodings` also prints a semantic analysis of the label-encodings file that is specified by *pathname*. If *pathname* is not specified, `chk_encodings` checks and analyzes the `/etc/security/tsol/label_encodings` file.

If label-encodings file analysis was requested, whatever analysis can be provided is written to the standard output file even if errors were found.

Options

`-a` Provide a semantic analysis of the label encodings file.

`-c` *maxclass* Accept a maximum classification value of *maxclass* (default 255) in the label encodings file `CLASSIFICATIONS` section.

Exit Status When successful, `chk_encodings` returns an exit status of `0` (true) and writes to the standard output file a confirmation that no errors were found in *pathname*. Otherwise, `chk_encodings` returns an exit status of nonzero (false) and writes an error diagnostic to the standard output file.

Files

`/etc/security/tsol/label_encodings`
The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	See below.
Standard	DDS-2600-6216-93, *Compartmented Mode Workstation Labeling: Encodings Format*, September 1993

The command output is Not-an-Interface. The command invocation is Committed for systems that implement the DIA MAC policy.

See Also `label_encodings(4)`, `attributes(5)`, `labels(5)`

"How to Analyze and Verify the label_encodings File" in *Oracle Solaris Trusted Extensions Label Administration*

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

This file is part of the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. This file might not be applicable to other MAC policies that might be developed for future releases of Solaris Trusted Extensions software.

Name chroot – change root directory for a command

Synopsis

```
/usr/sbin/chroot newroot command
```

Description The `chroot` utility causes *command* to be executed relative to *newroot*. The meaning of any initial slashes (`/`) in the path names is changed to *newroot* for *command* and any of its child processes. Upon execution, the initial working directory is *newroot*.

Notice that redirecting the output of *command* to a file,

```
chroot newroot command >x
```

will create the file `x` relative to the original root of *command*, not the new one.

The new root path name is always relative to the current root. Even if a `chroot` is currently in effect, the *newroot* argument is relative to the current root of the running process.

This command can be run only by the super-user.

Return Values The exit status of `chroot` is the return value of *command*.

Examples **EXAMPLE 1** Using the `chroot` Utility

The `chroot` utility provides an easy way to extract `tar` files (see `tar(1)`) written with absolute filenames to a different location. It is necessary to copy the shared libraries used by `tar` (see `ldd(1)`) to the *newroot* filesystem.

```
example# mkdir /tmp/lib; cd /lib
example# cp ld.so.1 libc.so.1 libcmd.so.1 libdl.so.1 \
         libsec.so.1 /tmp/lib
example# cp /usr/bin/tar /tmp
example# dd if=/dev/rmt/0 | chroot /tmp tar xvf -
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `cd(1)`, `tar(1)`, `chroot(2)`, `ttyname(3C)`, `attributes(5)`

Notes Exercise extreme caution when referencing device files in the new root file system.

References by routines such as `ttyname(3C)` to stdin, stdout, and stderr will find that the device associated with the file descriptor is unknown after `chroot` is run.

Name cimworkshop – start the Sun WBEM CIM WorkShop application

Synopsis

```
/usr/sadm/bin/cimworkshop
```

Description The `cimworkshop` command starts Sun WBEM CIM WorkShop, a graphical user interface that enables you to create, modify, and view the classes and instances that describe the managed resources on your system.

Managed resources are described using a standard information model called Common Information Model (CIM). A CIM class is a computer representation, or model, of a type of managed resource, such as a printer, disk drive, or CPU. A CIM instance is a particular managed resource that belongs to a particular class. Instances contain actual data. Objects can be shared by any WBEM-enabled system, device, or application. CIM objects are grouped into meaningful collections called schema. One or more schemas can be stored in directory-like structures called namespaces.

The CIM WorkShop application displays a Login dialog box. Context help is displayed on the left side of the CIM WorkShop dialog boxes. When you click on a field, the help content changes to describe the selected field.

By default, CIM WorkShop uses the RMI protocol to connect to the CIM Object Manager on the local host, in the default namespace, `root\cimv2`. You can select HTTP if you want to communicate to a CIM Object Manager using the standard XML/HTTP protocol from the Desktop Management Task Force. When a connection is established, all classes contained in the default namespace are displayed in the left side of the CIM WorkShop window.

The name of the current namespace is listed in the tool bar. All programming operations are performed within a namespace. Four namespaces are created in a `root` namespace during installation:

cimv2	Contains the default CIM classes that represent managed resources on your system.
security	Contains the security classes used by the CIM Object Manager to represent access rights for users and namespaces.
system	Contains properties for configuring the CIM Object Manager.
snmp	Contains pre-defined SNMP-related classes and all SNMP MOF files that are compiled.

The `cimworkshop` application allows you to perform the following tasks:

Create, view, and change namespaces.	Use the CIM WorkShop application to view all namespaces. A namespace is a directory-like structure that can store CIM classes and instances.

Create, delete, and view CIM classes.	You cannot modify the unique attributes of the classes that make up the CIM and Solaris Schema. You can create a new instance or subclass of the class and modify the desired attributes in that instance or subclass.
Create, modify, delete, and view CIM instances.	You can add instances to a class and modify its inherited properties or create new properties. You can also change the property values of a CIM instance.
Invoke methods.	You can set input values for a parameter of a method and invoke the method.

When CIM WorkShop connects to the CIM Object Manager in a particular namespace, all subsequent operations occur within that namespace. When you connect to a namespace, you can access the classes and instances in that namespace (if they exist) and in any namespaces contained in that namespace.

When you use CIM WorkShop to view CIM data, the WBEM system validates your login information on the current host. By default, a validated WBEM user is granted read access to the CIM Schema. The CIM Schema describes managed objects on your system in a standard format that all WBEM-enabled systems and applications can interpret.

Read Only	Allows read-only access to CIM Schema objects. Users with this privilege can retrieve instances and classes, but cannot create, delete, or modify CIM objects.
Read/Write	Allows full read, write, and delete access to all CIM classes and instances.
Write	Allows write and delete, but not read access to all CIM classes and instances.
None	Allows no access to CIM classes and instances.

Usage The `cimworkshop` command is not a tool for a distributed environment. Rather, this command is used for local administration on the machine on which the CIM Object Manager is running.

Exit Status The `cimworkshop` utility terminates with exit status `0`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbdev

See Also mofcomp(1M), wbemlogviewer(1M), init.wbem(1M), attributes(5)

Name clear_locks – clear locks held on behalf of an NFS client

Synopsis

```
/usr/sbin/clear_locks [-s] hostname
```

Description The `clear_locks` command removes all file, record, and share locks created by the *hostname* and held on the current host, regardless of which process created or owns the locks.

This command can be run only by the super-user.

This command should only be used to repair the rare case of a client crashing and failing to clear held locks. Clearing locks held by an active client may cause applications to fail in an unexpected manner.

Options

`-s` Remove all locks created by the current machine and held by the server *hostname*.

Operands The following operands are supported:

hostname name of host server

Exit Status

`0` Successful operation.

`1` If not root.

`2` Usage error.

`3` If unable to contact server (RPC).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `fcntl(2)`, `attributes(5)`

Name clinfo – display cluster information

Synopsis

```
clinfo [-nh]
```

Description The `clinfo` command displays cluster configuration information about the node from which the command is executed.

Without arguments, `clinfo` returns an exit status of `0` if the node is configured and booted as part of a cluster. Otherwise, `clinfo` returns an exit status of `1`.

Options The following options are supported:

`-h` Displays the highest node number allowed to be configured. This is different from the maximum number of nodes supported in a given cluster. The current highest configured node number can change immediately after the command returns since new nodes can be dynamically added to a running cluster.

For example, `clinfo -h` might return `64`, meaning that the highest number you can use to identify a node is `64`. See the *Sun Cluster 3.0 System Administration Guide* for a description of utilities you can use to determine the number of nodes in a cluster.

`-n` Prints the number of the node from which `clinfo` is executed.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

This is usually because the node is not configured or booted as part of a cluster.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `attributes(5)`

Name clri, dcopy – clear inode

Synopsis

```
clri [-F FSType] [-V] special i-number

dcopy [-F FSType] [-V] special i-number
```

Description `clri` writes zeros on the inodes with the decimal *i-number* on the file system stored on *special*. After `clri`, any blocks in the affected file show up as missing in an `fsck(1M)` of *special*.

Read and write permission is required on the specified file system device. The inode becomes allocatable.

The primary purpose of this routine is to remove a file that for some reason appears in no directory. If it is used to zap an inode that does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the inode is reallocated to some new file, the old entry will still point to that file. At that point, removing the old entry will destroy the new file. The new entry will again point to an unallocated inode, so the whole cycle is likely to be repeated again and again.

`dcopy` is a symbolic link to `clri`.

Options

`-F` *FSType* Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching *special* with an entry in the table, or by consulting `/etc/default/fs`.

`-V` Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option should be used to verify and validate the command line.

Usage See `largefile(5)` for the description of the behavior of `clri` and `dcopy` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/default/fs` Default local file system type

`/etc/vfstab` List of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `fsck(1M)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`

Notes This command might not be supported for all *FSTypes*.

Name consadm – select or display devices used as auxiliary console devices

Synopsis

```
/usr/sbin/consadm

/usr/sbin/consadm [-a device. . .] [-p]

/usr/sbin/consadm [-d device. . .] [-p]

/usr/sbin/consadm [-p]
```

Description `consadm` selects the hardware *device* or devices to be used as auxiliary console devices, or displays the current device. Only superusers are allowed to make or display auxiliary console device selections.

Auxiliary console devices receive copies of `console` messages, and can be used as the console during `single user` mode. In particular, they receive kernel messages and messages directed to `/dev/sysmsg`. On Solaris x86 based systems they can also be used for interaction with the bootstrap.

By default, selecting a display device to be used as an auxiliary console device selects that device for the duration the system remains up. If the administrator needs the selection to persist across reboots the `-p` option can be specified.

`consadm` runs a daemon in the background, monitoring auxiliary console devices. Any devices that are disconnected (hang up, lose carrier) are removed from the auxiliary console device list, though not from the persistent list. While auxiliary console devices may have been removed from the device list receiving copies of `console` messages, those messages will always continue to be displayed by the default console device.

The daemon will not run if it finds there are not any auxiliary devices configured to monitor. Likewise, after the last auxiliary console is removed, the daemon will shut itself down. Therefore the daemon persists for only as long as auxiliary console devices remain active.

See `eeprom(1M)` for instructions on assigning an auxiliary console device as the system console.

Options The following options are supported:

`-a` *device* Adds *device* to the list of auxiliary console devices. Specify *device* as the path name to the device or devices to be added to the auxiliary console device list.

`-d` *device* Removes *device* from the list of auxiliary console devices. Specify *device* as the path name to the device or devices to be removed from the auxiliary console device list.

`-p` Prints the list of auxiliary consoles that will be auxiliary across reboots.

When invoked with the `-a` or `-d` options , tells the application to make the change persist across reboot.

Examples

EXAMPLE 1 Adding to the list of devices that will receive console messages

The following command adds `/dev/term/a` to the list of devices that will receive console messages.

```
example# consadm -a /dev/term/a
```

EXAMPLE 2 Removing from the list of devices that will receive console messages

The following command removes `/dev/term/a` from the list of devices that will receive console messages. This includes removal from the persistent list.

```
example# consadm -d -p /dev/term/a
```

EXAMPLE 3 Printing the list of devices selected as auxiliary console devices

The following command prints the name or names of the device or devices currently selected as auxiliary console devices.

```
example# consadm
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `consadm`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Exit Status

The following exit values are returned:

`0` Successful completion.

>`0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Stability Level	Evolving

See Also

`svcs(1)`, `eeprom(1M)`, `svcadm(1M)`, `syslogd(1M)`, `kadb(1M)`, `environ(5)`, `attributes(5)`, `smf(5)`, `sysmsg(7D)`, `console(7D)`

Notes

Auxiliary console devices are not usable for `kadb` or firmware I/O, do not receive panic messages, and do not receive output directed to `/dev/console`.

The `consadm` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/consadm
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name conv_lp – convert LP configuration

Synopsis

```
conv_lp [-d dir] [-f file]
```

Description `conv_lp` reads LP printer configuration information from a directory and converts it to an output file for use with print client software.

Options The following options are supported:

`-d` *dir* The root (‘ / ') directory from which LP configuration information is read. The default is root (‘ / ').

`-f` *file* The output file to which `conv_lp` writes the converted LP configuration information. The default is `/etc/printers.conf`.

Examples

EXAMPLE 1 Converting LP Configuration Information from the Default Directory and File

The following example converts LP configuration information from directory root (`/`) to file `/etc/printers.conf`.

```
% conv_lp
```

EXAMPLE 2 Converting LP Configuration Information From a Specified Directory and File

The following example converts LP configuration information from directory `/export/root/client` to file `/export/root/client/etc/printers.conf`.

```
% conv_lp -d /export/root/client -f\
   /export/root/client/etc/printers.conf
```

Exit Status The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Files `/etc/printers.conf` System printer configuration database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

See Also `lpset(1M)`, `printers.conf(4)`, `attributes(5)`

Name conv_lpd – convert LPD configuration

Synopsis

```
conv_lpd [-c printers | -c printcap] [-n] file
```

Description `conv_lpd` converts LPD printer configuration information from *file* to a `printers.conf` or a `printcap` file (see `printers.conf(4)`). *file* specifies the name of the input file, and can be either in `printers.conf` or `printcap` format. If *file* is in `printers.conf` format, it converts it to a `printcap` file. If *file* is in `printcap` format, it converts it to a `printers.conf` file.

Options The following options are supported:

`-c printers | -c printcap` Specifies the type of output file produced by the conversion. `-c printers` converts to a `printers.conf` file. `-c printcap` converts to a `printcap` file. `-c printers` is the default.

`-n` Preserves the namelist during the conversion.

Operands The following operands are supported:

file The file to be converted.

Examples

EXAMPLE 1 Converting a `printcap` file to a `printers.conf` file.

The following example converts a `printcap` file to a `printers.conf` file.

```
example% conv_lpd /etc/printcap
```

EXAMPLE 2 Converting a `printcap` file to a `printers.conf` file and preserving the namelist.

The following example converts a `printcap` file to a `printers.conf` file and preserves the namelist.

```
example% conv_lpd -c printers -n /etc/printcap
```

EXAMPLE 3 Converting a `printers.conf` file to a `printcap` file and preserving the namelist.

The following example converts a `printers.conf` file to a `printcap` file and preserves the namelist.

```
example% conv_lpd -c printcap -n /etc/printers.conf
```

Exit Status The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Files

`/etc/printers.conf` System printer configuration database.

`/etc/printcap` SunOS 4.*x* printer capability database.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

See Also lpset(1M), printers.conf(4), attributes(5)

Name coreadm – core file administration

Synopsis

```
coreadm [-g pattern] [-G content] [-i pattern] [-I content]
     [-d option]... [-e option]...

coreadm [-p pattern] [-P content] [pid]...

coreadm -u
```

Description `coreadm` specifies the name and location of core files produced by abnormally-terminating processes. See `core(4)`.

Only users who have the `sys_admin` privilege can execute the first form of the `SYNOPSIS`. This form configures system-wide core file options, including a global core file name pattern and a core file name pattern for the `init`(1M) process. All settings are saved in `coreadm`'s configuration file `/etc/coreadm.conf` to set at boot. See `init(1M)`.

Nonprivileged users can execute the second form of the `SYNOPSIS`. This form specifies the file name pattern and core file content that the operating system uses to generate a per-process core file.

Only users who have the `sys_admin` privilege can execute the third form of the `SYNOPSIS`. This form updates all system-wide core file options, based on the contents of `/etc/coreadm.conf`. Normally, this option is used on reboot when starting `svc:/system/coreadm:default`.

A core file name pattern is a normal file system path name with embedded variables, specified with a leading `%` character. The variables are expanded from values that are effective when a core file is generated by the operating system. The possible embedded variables are as follows:

%d Executable file directory name, up to a maximum of `MAXPATHLEN` characters

%f Executable file name, up to a maximum of `MAXCOMLEN` characters

%g Effective group-ID

%m Machine name (`uname -m`)

%n System node name (`uname -n`)

%p Process-ID

%t Decimal value of `time(2)`

%u Effective user-ID

%z Name of the zone in which process executed (`zonename`)

%% Literal `%`

For example, the core file name pattern `/var/core/core.%f.%p` would result, for command `foo` with process-ID `1234`, in the core file name `/var/core/core.foo.1234`.

A core file content description is specified using a series of tokens to identify parts of a process's binary image:

anon	Anonymous private mappings, including thread stacks that are not main thread stacks
ctf	CTF type information sections for loaded object files
data	Writable private file mappings
dism	DISM mappings
heap	Process heap
ism	ISM mappings
rodata	Read-only private file mappings
shanon	Anonymous shared mappings
shfile	Shared mappings that are backed by files
shm	System V shared memory
stack	Process stack
symtab	Symbol table sections for loaded object files
text	Readable and executable private file mappings

In addition, you can use the token `all` to indicate that core files should include all of these parts of the process's binary image. You can use the token `none` to indicate that no mappings are to be included. The `default` token indicates inclusion of the system default content (`stack+heap+shm+ism+dism+text+data+rodata+anon+shanon+ctf`). The `/proc` file system data structures are always present in core files regardless of the mapping content.

You can use `+` and `-` to concatenate tokens. For example, the core file content `default-ism` would produce a core file with the default set of mappings without any intimate shared memory mappings.

The `coreadm` command with no arguments reports the current system configuration, for example:

```
$ coreadm
    global core file pattern: /var/core/core.%f.%p
    global core file content: all
      init core file pattern: core
      init core file content: default
           global core dumps: enabled
      per-process core dumps: enabled
     global setid core dumps: enabled
```

```
per-process setid core dumps: disabled
    global core dump logging: disabled
```

The `coreadm` command with only a list of process-IDs reports each process's per-process core file name pattern, for example:

```
$ coreadm 278 5678
  278:   core.%f.%p default
  5678:  /home/george/cores/%f.%p.%t all-ism
```

Only the owner of a process or a user with the `proc_owner` privilege can interrogate a process in this manner.

When a process is dumping core, up to three core files can be produced: one in the per-process location, one in the system-wide global location, and, if the process was running in a local (non-global) zone, one in the global location for the zone in which that process was running. Each core file is generated according to the effective options for the corresponding location.

When generated, a global core file is created in mode `600` and owned by the superuser. Nonprivileged users cannot examine such files.

Ordinary per-process core files are created in mode `600` under the credentials of the process. The owner of the process can examine such files.

A process that is or ever has been `setuid` or `setgid` since its last `exec(2)` presents security issues that relate to dumping core. Similarly, a process that initially had superuser privileges and lost those privileges through `setuid(2)` also presents security issues that are related to dumping core. A process of either type can contain sensitive information in its address space to which the current nonprivileged owner of the process should not have access. If `setid` core files are enabled, they are created mode `600` and owned by the superuser.

Options The following options are supported:

`-d` *option*... Disable the specified core file option. See the `-e` *option* for descriptions of possible options.

Multiple `-e` and `-d` options can be specified on the command line. Only users with the `sys_admin` privilege can use this option.

`-e` *option*... Enable the specified core file option. Specify *option* as one of the following:

global	Allow core dumps that use global core pattern.
global-setid	Allow set-id core dumps that use global core pattern.
log	Generate a `syslog(3C)` message when generation of a global core file is attempted.
process	Allow core dumps that use per-process core pattern.

proc-setid — Allow set-id core dumps that use per-process core pattern.

Multiple `-e` and `-d` options can be specified on the command line. Only users with the `sys_admin` privilege can use this option.

`-g` *pattern* — Set the global core file name pattern to *pattern*. The pattern must start with a `/` and can contain any of the special `%` variables that are described in the `DESCRIPTION`.

Only users with the `sys_admin` privilege can use this option.

`-G` *content* — Set the global core file content to content. You must specify content by using the tokens that are described in the `DESCRIPTION`.

Only users with the `sys_admin` privilege can use this option.

`-i` *pattern* — Set the default per-process core file name to *pattern*. This changes the per-process pattern for any process whose per-process pattern is still set to the default. Processes that have had their per-process pattern set or are descended from a process that had its per-process pattern set (using the `-p` option) are unaffected. This default persists across reboot.

Only users with the `sys_admin` or `proc_owner` privilege can use this option.

`-I` *content* — Set the default per-process core file content to *content*. This changes the per-process content for any process whose per-process content is still set to the default. Processes that have had their per-process content set or are descended from a process that had its per-process content set (using the `-P` option) are unaffected. This default persists across reboot.

Only users with the `sys_admin` or `proc_owner` privileges can use this option.

`-p` *pattern* — Set the per-process core file name pattern to *pattern* for each of the specified process-IDs. The pattern can contain any of the special `%` variables described in the `DESCRIPTION` and need not begin with `/`. If the pattern does not begin with `/`, it is evaluated relative to the directory that is current when the process generates a core file.

A nonprivileged user can apply the `-p` option only to processes that are owned by that user. A user with the `proc_owner` privilege can apply the option to any process. The per-process core file name pattern is inherited by future child processes of the affected processes. See `fork(2)`.

If no process-IDs are specified, the `-p` option sets the per-process core file name pattern to *pattern* on the parent process (usually the shell that ran `coreadm`).

`-P` *content* Set the per-process core file content to *content* for each of the specified process-IDs. The content must be specified by using the tokens that are described in the `DESCRIPTION`.

A nonprivileged user can apply the `-p` option only to processes that are owned by that user. A user with the `proc_owner` privilege can apply the option to any process. The per-process core file name pattern is inherited by future child processes of the affected processes. See `fork(2)`.

If no process-IDs are specified, the `-P` option sets the per-process file content to *content* on the parent process (usually the shell that ran `coreadm`).

`-u` Update system-wide core file options from the contents of the configuration file `/etc/coreadm.conf`. If the configuration file is missing or contains invalid values, default values are substituted. Following the update, the configuration file is resynchronized with the system core file configuration.

Only users with the `sys_admin` privilege can use this option.

Operands The following operands are supported:

pid process-ID

Examples **EXAMPLE 1** Setting the Core File Name Pattern

When executed from a user's `$HOME/.profile` or `$HOME/.login`, the following command sets the core file name pattern for all processes that are run during the login session:

```
example$  coreadm -p core.%f.%p
```

Note that since the process-ID is omitted, the per-process core file name pattern will be set in the shell that is currently running and is inherited by all child processes.

EXAMPLE 2 Dumping a User's Files Into a Subdirectory

The following command dumps all of a user's core dumps into the `corefiles` subdirectory of the home directory, discriminated by the system node name. This command is useful for users who use many different machines but have a shared home directory.

```
example$  coreadm -p $HOME/corefiles/%n.%f.%p 1234
```

EXAMPLE 3 Culling the Global Core File Repository

The following commands set up the system to produce core files in the global repository only if the executables were run from `/usr/bin` or `/usr/sbin`.

```
example# mkdir -p /var/cores/usr/bin
example# mkdir -p /var/cores/usr/sbin
example# coreadm -G all -g /var/cores/%d/%f.%p.%n
```

Files `/etc/coreadm.conf`

Exit Status The following exit values are returned:

0 Successful completion.

1 A fatal error occurred while either obtaining or modifying the system core file configuration.

2 Invalid command-line options were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `gcore(1)`, `svcs(1)`, `init(1M)`, `svcadm(1M)`, `exec(2)`, `fork(2)`, `setuid(2)`, `time(2)`, `syslog(3C)`, `core(4)`, `attributes(5)`, `smf(5)`

Notes In a local (non-global) zone, the global settings apply to processes running in that zone. In addition, the global zone's apply to processes run in any zone.

The term *global settings* refers to settings which are applied to the system or zone as a whole, and does not necessarily imply that the settings are to take effect in the global zone.

The `coreadm` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/coreadm:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name cpustat – monitor system behavior using CPU performance counters

Synopsis

```
cpustat -c eventspec [-c eventspec]... [-p period] [-sntD]
     [interval [count]]

cpustat -h
```

Description The `cpustat` utility allows CPU performance counters to be used to monitor the overall behavior of the CPUs in the system.

If *interval* is specified, `cpustat` samples activity every *interval* seconds, repeating forever. If a *count* is specified, the statistics are repeated *count* times. If neither are specified, an interval of five seconds is used, and there is no limit to the number of samples that are taken.

Options The following options are supported:

`-c` *eventspec* Specifies a set of events for the CPU performance counters to monitor. The syntax of these event specifications is:

```
[picn=]eventn[,attr[n][=val]][,[picn=]eventn
     [,attr[n][=val]],...,]
```

You can use the `-h` option to obtain a list of available events and attributes. This causes generation of the usage message. You can omit an explicit counter assignment, in which case `cpustat` attempts to choose a capable counter automatically.

Attribute values can be expressed in hexadecimal, octal, or decimal notation, in a format suitable for `strtoll(3C)`. An attribute present in the event specification without an explicit value receives a default value of `1`. An attribute without a corresponding counter number is applied to all counters in the specification.

The semantics of these event specifications can be determined by reading the CPU manufacturer's documentation for the events.

Multiple `-c` options can be specified, in which case the command cycles between the different event settings on each sample.

`-D` Enables debug mode.

`-h` Prints an extensive help message on how to use the utility and how to program the processor-dependent counters.

`-p` *period* Causes `cpustat` to cycle through the list of *eventspec*s every *period* seconds. The tool sleeps after each cycle until *period* seconds have elapsed since the first *eventspec* was measured.

When this option is present, the optional *count* parameter specifies the number of total cycles to make (instead of the number of total samples to take). If *period* is less than the number of *eventspec*s times *interval*, the tool acts as it period is `0`.

`-s` Creates an idle soaker thread to spin while system-only *eventspec*s are bound. One idle soaker thread is bound to each CPU in the current processor set. System-only *eventspec*s contain both the `nouser` and the `sys` tokens and measure events that occur while the CPU is operating in privileged mode. This option prevents the kernel's idle loop from running and triggering system-mode events.

`-n` Omits all header output (useful if `cpustat` is the beginning of a pipeline).

`-t` Prints an additional column of processor cycle counts, if available on the current architecture.

Usage A closely related utility, `cputrack(1)`, can be used to monitor the behavior of individual applications with little or no interference from other activities on the system.

The `cpustat` utility must be run by the super-user, as there is an intrinsic conflict between the use of the CPU performance counters system-wide by `cpustat` and the use of the CPU performance counters to monitor an individual process (for example, by `cputrack`.)

Once any instance of this utility has started, no further per-process or per-LWP use of the counters is allowed until the last instance of the utility terminates.

The times printed by the command correspond to the wallclock time when the hardware counters were actually sampled, instead of when the program told the kernel to sample them. The time is derived from the same timebase as `gethrtime(3C)`.

The processor cycle counts enabled by the `-t` option always apply to both user and system modes, regardless of the settings applied to the performance counter registers.

On some hardware platforms running in system mode using the “sys” token, the counters are implemented using 32-bit registers. While the kernel attempts to catch all overflows to synthesize 64-bit counters, because of hardware implementation restrictions, overflows can be lost unless the sampling interval is kept short enough. The events most prone to wrap are those that count processor clock cycles. If such an event is of interest, sampling should occur frequently so that less than 4 billion clock cycles can occur between samples.

The output of cpustat is designed to be readily parseable by `nawk(1)` and `perl(1)`, thereby allowing performance tools to be composed by embedding `cpustat` in scripts. Alternatively, tools can be constructed directly using the same APIs that `cpustat` is built upon using the facilities of `libcpc(3LIB)`. See `cpc(3CPC)`.

The `cpustat` utility only monitors the CPUs that are accessible to it in the current processor set. Thus, several instances of the utility can be running on the CPUs in different processor sets. See `psrset(1M)` for more information about processor sets.

Because `cpustat` uses LWPs bound to CPUs, the utility might have to be terminated before the configuration of the relevant processor can be changed.

Examples

SPARC **EXAMPLE 1** Measuring External Cache References and Misses

The following example measures misses and references in the external cache. These occur while the processor is operating in user mode on an UltraSPARC machine.

```
example% cpustat -c EC_ref,EC_misses 1 3

    time cpu event      pic0      pic1
   1.008   0  tick     69284      1647
   1.008   1  tick     43284      1175
   2.008   0  tick    179576      1834
   2.008   1  tick    202022     12046
   3.008   0  tick     93262       384
   3.008   1  tick     63649      1118
   3.008   2 total    651077     18204
```

x86 **EXAMPLE 2** Measuring Branch Prediction Success on Pentium 4

The following example measures branch mispredictions and total branch instructions in user and system mode on a Pentium 4 machine.

```
 example% cpustat -c \
    pic12=branch_retired,emask12=0x4,pic14=branch_retired,\
    emask14=0xf,sys 1 3

    time cpu event      pic12     pic14
   1.010   1  tick       458       684
   1.010   0  tick       305       511
   2.010   0  tick       181       269
   2.010   1  tick       469       684
   3.010   0  tick       182       269
   3.010   1  tick       468       684
   3.010   2 total      2063      3101
```

EXAMPLE 3 Counting Memory Accesses on Opteron

The following example determines the number of memory accesses made through each memory controller on an Opteron, broken down by internal memory latency:

EXAMPLE 3 Counting Memory Accesses on Opteron *(Continued)*

```
cpustat -c \
   pic0=NB_mem_ctrlr_page_access,umask0=0x01, \
   pic1=NB_mem_ctrlr_page_access,umask1=0x02, \
   pic2=NB_mem_ctrlr_page_access,umask2=0x04,sys \
   1

    time cpu event      pic0      pic1      pic2
   1.003   0  tick     41976     53519      7720
   1.003   1  tick      5589     19402       731
   2.003   1  tick      6011     17005       658
   2.003   0  tick     43944     45473      7338
   3.003   1  tick      7105     20177       762
   3.003   0  tick     47045     48025      7119
   4.003   0  tick     43224     46296      6694
   4.003   1  tick      5366     19114       652
```

Warnings By running the `cpustat` command, the super-user forcibly invalidates all existing performance counter context. This can in turn cause all invocations of the `cputrack` command, and other users of performance counter context, to exit prematurely with unspecified errors.

If `cpustat` is invoked on a system that has CPU performance counters which are not supported by Solaris, the following message appears:

```
cpustat: cannot access performance counters - Operation not applicable
```

This error message implies that `cpc_open()` has failed and is documented in `cpc_open(3CPC)`. Review this documentation for more information about the problem and possible solutions.

If a short interval is requested, `cpustat` might not be able to keep up with the desired sample rate. In this case, some samples might be dropped.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcpcu
Interface Stability	Evolving

See Also `cputrack(1)`, `nawk(1)`, `perl(1)`, `iostat(1M)`, `prstat(1M)`, `psrset(1M)`, `vmstat(1M)`, `cpc(3CPC)`, `cpc_open(3CPC)`, `cpc_bind_cpu(3CPC)`, `gethrtime(3C)`, `strtoll(3C)`, `libcpc(3LIB)`, `attributes(5)`

Notes When `cpustat` is run on a Pentium 4 with HyperThreading enabled, a CPC set is bound to only one logical CPU of each physical CPU. See `cpc_bind_cpu(3CPC)`.

Name cron – clock daemon

Synopsis

```
/usr/sbin/cron
```

Description `cron` starts a process that executes commands at specified dates and times.

You can specify regularly scheduled commands to `cron` according to instructions found in `crontab` files in the directory `/var/spool/cron/crontabs`. Users can submit their own `crontab` file using the `crontab(1)` command. Commands which are to be executed only once can be submitted using the `at(1)` command.

`cron` only examines `crontab` or `at` command files during its own process initialization phase and when the `crontab` or `at` command is run. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

As `cron` never exits, it should be executed only once. This is done routinely by way of the `svc:/system/cron:default` service. The file `/etc/cron.d/FIFO` file is used as a lock file to prevent the execution of more than one instance of `cron`.

`cron` captures the output of the job's `stdout` and `stderr` streams, and, if it is not empty, mails the output to the user. If the job does not produce output, no mail is sent to the user. An exception is if the job is an `at(1)` job and the `-m` option was specified when the job was submitted.

`cron` and `at` jobs are not executed if your account is locked. Jobs and processses execute. The `shadow(4)` file defines which accounts are not locked and will have their jobs and processes executed.

Setting cron Jobs Across Timezones

The timezone of the `cron` daemon sets the system-wide timezone for `cron` entries. This, in turn, is by set by default system-wide using `/etc/default/init`.

If some form of *daylight savings* or *summer/winter time* is in effect, then jobs scheduled during the switchover period could be executed once, twice, or not at all.

Setting cron Defaults

To keep a log of all actions taken by `cron`, you must specify `CRONLOG=YES` in the `/etc/default/cron` file. If you specify `CRONLOG=NO`, no logging is done. Keeping the log is a user configurable option since `cron` usually creates huge log files.

You can specify the `PATH` for `user` cron jobs by using `PATH=` in `/etc/default/cron`. You can set the `PATH` for `root` cron jobs using `SUPATH=` in `/etc/default/cron`. Carefully consider the security implications of setting `PATH` and `SUPATH`.

Example `/etc/default/cron` file:

```
CRONLOG=YES
PATH=/usr/bin:/usr/ucb:
```

This example enables logging and sets the default `PATH` used by non-root jobs to `/usr/bin:/usr/ucb:`. Root jobs continue to use `/usr/sbin:/usr/bin`.

The cron log file is periodically rotated by logadm(1M).

Files

/etc/cron.d	Main cron directory
/etc/cron.d/FIFO	Lock file
/etc/default/cron	cron default settings file
/var/cron/log	cron history information
/var/spool/cron	Spool area
/etc/cron.d/queuedefs	Queue description file for at, batch, and cron
/etc/logadm.conf	Configuration file for logadm

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also svcs(1), at(1), crontab(1), sh(1), logadm(1M), svcadm(1M), queuedefs(4), shadow(4), attributes(5), rbac(5), smf(5), smf_security(5)

Notes The cron service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/cron:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command. Most administrative actions may be delegated to users with the solaris.smf.manage.cron authorization (see rbac(5) and smf_security(5)).

Diagnostics A history of all actions taken by cron is stored in /var/cron/log and possibly in /var/cron/olog.

Name cryptoadm – cryptographic framework administration

Synopsis

```
cryptoadm list [-mpv] [provider=provider-name]
     [mechanism=mechanism-list]

cryptoadm disable
     provider=provider-name mechanism=mechanism-list | random | all

cryptoadm enable
     provider=provider-name mechanism=mechanism-list | random | all

cryptoadm install provider=provider-name

cryptoadm install provider=provider-name
     [mechanism=mechanism-list]

cryptoadm uninstall provider=provider-name

cryptoadm unload provider=provider-name

cryptoadm refresh

cryptoadm start

cryptoadm stop

cryptoadm --help
```

Description The `cryptoadm` utility displays cryptographic provider information for a system, configures the mechanism policy for each provider, and installs or uninstalls a cryptographic provider. The cryptographic framework supports three types of providers: a user-level provider (a PKCS11 shared library), a kernel software provider (a loadable kernel software module), and a kernel hardware provider (a cryptographic hardware device).

For kernel software providers, the `cryptoadm` utility provides the `unload` subcommand. This subcommand instructs the kernel to unload a kernel software providers.

For the cryptographic framework's metaslot, the `cryptoadm` utility provides subcommands to enable and disable the metaslot's features, list metaslot's configuration, specify alternate persistent object storage, and configure the metaslot's mechanism policy.

Administrators will find it useful to use syslog facilities (see `syslogd(1M)` and `logadm(1M)`) to maintain the cryptographic subsystem. Logging can be especially useful under the following circumstances:

- If kernel-level daemon is dead, all applications fail. You can learn this from syslog and use `svcadm(1M)` to restart the `svc:/system/cryptosvc` service.
- If there are bad providers plugged into the framework, you can learn this from syslog and remove the bad providers from the framework.

With the exception of the subcommands or options listed below, the `cryptoadm` command needs to be run by a privileged user.

- subcommand `list`, any options

- subcommand `--help`

Options The `cryptoadm` utility has the various combinations of subcommands and options shown below.

`cryptoadm list`
Display the list of installed providers.

`cryptoadm list metaslot`
Display the system-wide configuration for metaslot.

`cryptoadm list -m [ provider=`*provider-name* `| metaslot ]`
Display a list of mechanisms that can be used with the installed providers or metaslot. If a provider is specified, display the name of the specified provider and the mechanism list that can be used with that provider. If the metaslot keyword is specified, display the list of mechanisms that can be used with metaslot.

`cryptoadm list -p [ provider=`*provider-name* `| metaslot ]`
Display the mechanism policy (that is, which mechanisms are available and which are not) for the installed providers. Also display the provider feature policy or metaslot. If a provider is specified, display the name of the provider with the mechanism policy enforced on it only. If the metaslot keyword is specified, display the mechanism policy enforced on the metaslot.

`cryptoadm list -v provider=`*provider-name* `| metaslot`
Display details about the specified provider if a provider is specified. If the metaslot keyword is specified, display details about the metaslot.

`-v`
For the various `list` subcommands described above (except for `list -p`), the `-v` (verbose) option provides details about providers, mechanisms and slots.

`cryptoadm disable provider=`*provider-name*
[mechanism=*mechanism-list* | *provider-feature* `...` | `all`]
Disable the mechanisms or provider features specified for the provider. See OPERANDS for a description of *mechanism*, *provider-feature*, and the `all` keyword.

`cryptoadm [ mechanism=`*mechanism-list* `] [ auto-key-migrate ]`
Disable the metaslot feature in the cryptographic framework or disable some of metaslot's features. If no operand is specified, this command disables the metaslot feature in the cryptographic framework. If a list of mechanisms is specified, disable mechanisms specified for metaslot. If all mechanisms are disabled for metaslot, the metaslot will be disabled. See OPERANDS for a description of mechanism. If the `auto-key-migrate` keyword is specified, it disables the migration of sensitive token objects to other slots even if it is necessary for performing crypto operations. See OPERANDS for a description of `auto-key-migrate`.

`cryptoadm enable provider=`*provider-name*
[mechanism=*mechanism-list* | *provider-feature* `...` | `all`]

Enable the mechanisms or provider features specified for the provider. See OPERANDS for a description of *mechanism*, *provider-feature*, and the `all` keyword.

`cryptoadm enable metaslot [ mechanism=`*mechanism-list* `] |`
`[ [ token=`*token-label*`] [ slot=`*slot-description*`] |`
`default-keystore ] | [ auto-key-migrate ]`

If no operand is specified, this command enables the metaslot feature in the cryptographic framework. If a list of mechanisms is specified, it enables only the list of specified mechanisms for metaslot. If *token-label* is specified, the specified token will be used as the persistent object store. If the *slot-description* is specified, the specified slot will be used as the persistent object store. If both the *token-label* and the *slot-description* are specified, the provider with the matching token label and slot description is used as the persistent object store. If the `default-keystore` keyword is specified, metaslot will use the default persistent object store. If the `auto-key-migrate` keyword is specified, sensitive token objects will automatically migrate to other slots as needed to complete certain crypto operations. See OPERANDS for a description of mechanism, token, slot, `default-keystore`, and `auto-key-migrate`.

`cryptoadm install provider=`*provider-name*

Install a user-level provider into the system. The *provider* operand must be an absolute pathname of the corresponding shared library. If there are both 32–bit and 64–bit versions for a library, this command should be run once only with the path name containing `$ISA`. Note that `$ISA` is not a reference to an environment variable. Note also that `$ISA` must be quoted (with single quotes [for example, `'$ISA'`]) or the `$` must be escaped to keep it from being incorrectly expanded by the shell. The user-level framework expands `$ISA` to an empty string or an architecture-specific directory, for example, `sparcv9`.

The preferred way of installing a user-level provider is to build a package for the provider. For more information, see the *Solaris Security for Developer's Guide*.

`cryptoadm install provider=`*provider-name*
mechanism=*mechanism-list*

Install a kernel software provider into the system. The provider should contain the base name only. The *mechanism-list* operand specifies the complete list of mechanisms to be supported by this provider.

The preferred way of installing a kernel software provider is to build a package for providers. For more information, see the *Solaris Security for Developer's Guide*.

`cryptoadm uninstall provider=`*provider-name*

Uninstall the specified *provider* and the associated mechanism policy from the system. This subcommand applies only to a user-level provider or a kernel software provider.

`cryptoadm unload provider=`*provider-name*

Unload the kernel software module specified by *provider*.

`cryptoadm refresh`
`cryptoadm start`

`cryptoadm stop`
Private interfaces for use by `smf(5)`, these must not be used directly.

`cryptoadm -help`
Display the command usage.

Operands

provider=*provider-name*	A user-level provider (a PKCS11 shared library), a kernel software provider (a loadable kernel software module), or a kernel hardware provider (a cryptographic hardware device). A valid value of the *provider* operand is one entry from the output of a command of the form: `cryptoadm` *list*. A *provider* operand for a user-level provider is an absolute pathname of the corresponding shared library. A *provider* operand for a kernel software provider contains a base name only. A *provider* operand for a kernel hardware provider is in a "*name*/*number*" form.
mechanism=*mechanism-list*	A comma separated list of one or more PKCS #11 mechanisms. A process for implementing a cryptographic operation as defined in PKCS #11 specification. You can substitute `all` for *mechanism-list*, to specify all mechanisms on a provider. See the discussion of the `all` keyword, below.
provider-feature	A cryptographic framework feature for the given provider. Currently only `random` is accepted as a feature. For a user-level provider, disabling the random feature makes the PKCS #11 routines `C_GenerateRandom` and `C_SeedRandom` unavailable from the provider. For a kernel provider, disabling the random feature prevents `/dev/random` from gathering random numbers from the provider.
`all`	The keyword all can be used with with the `disable` and `enable` subcommands to operate on all provider features.
`token=`*token-label*	The label of a token in one of the providers in the cryptographic framework. A valid value of the token operand is an item displayed under "Token Label" from the output of the command `cryptoadm list -v`.
`slot=`*slot-description*	The description of a slot in one of the providers in the cryptographic framework. A valid value of the slot operand is an item displayed under "Description" from the output of the command `cryptoadm list -v`.

`default-keystore`	The keyword `default-keystore` is valid only for metaslot. Specify this keyword to set the persistent object store for metaslot back to using the default store.
`auto-key-migrate`	The keyword auto-key-migrate is valid only for metaslot. Specify this keyword to configure whether metaslot is allowed to move sensitive token objects from the token object slot to other slots for performing cryptographic operations.

The keyword `all` can be used in two ways with the `disable` and `enable` subcommands:

- You can substitute `all` for `mechanism`=*mechanism-list*, as in:

  ```
  # cryptoadm enable provider=dca/0 all
  ```

  This command enables the mechanisms on the provider *and* any other provider-features, such as `random`. You can also use `all` as an argument to `mechanism`, as in:

  ```
  # cryptoadm enable provider=des mechanism=all
  ```

  ...which enables all mechanisms on the provider, but enables no other provider–features, such as `random`.

Examples

EXAMPLE 1 Display List of Providers Installed in System

The following command displays a list of all installed providers:

```
example% cryptoadm list
user-level providers:
/usr/lib/security/$ISA/pkcs11_kernel.so
/usr/lib/security/$ISA/pkcs11_softtoken.so
/opt/lib/libcryptoki.so.1
/opt/SUNWconn/lib/$ISA/libpkcs11.so.1

kernel software providers:
    des
    aes
    bfish
    sha1
    md5

kernel hardware providers:
    dca/0
```

EXAMPLE 2 Display Mechanism List for `md5` Provider

The following command is a variation of the `list` subcommand:

```
example% cryptoadm list -m provider=md5
md5: CKM_MD5,CKM_MD5_HMAC,CKM_MD5_HMAC_GENERAL
```

EXAMPLE 3 Disable Specific Mechanisms for Kernel Software Provider

The following command disables mechanisms `CKM_DES3_ECB` and `CKM_DES3_CBC` for the kernel software provider `des`:

```
example# cryptoadm disable provider=des
```

EXAMPLE 4 Display Mechanism Policy for a Provider

The following command displays the mechanism policy for the `des` provider:

```
example% cryptoadm list -p provider=des
des: All mechanisms are enabled, except CKM_DES3_ECB, CKM_DES3_CBC
```

EXAMPLE 5 Enable Specific Mechanism for a Provider

The following command enables the `CKM_DES3_ECB` mechanism for the kernel software provider `des`:

```
example# cryptoadm enable provider=des mechanism=CKM_DES3_ECB
```

EXAMPLE 6 Install User-Level Provider

The following command installs a user-level provider:

```
example# cryptoadm install provider=/opt/lib/libcryptoki.so.1
```

EXAMPLE 7 Install User-Level Provider That Contains 32– and 64–bit Versions

The following command installs a user-level provider that contains both 32–bit and 64–bit versions:

```
example# cryptoadm install \
provider=/opt/SUNWconn/lib/'$ISA'/libpkcs11.so.1
```

EXAMPLE 8 Uninstall a Provider

The following command uninstalls the `md5` provider:

```
example# cryptoadm uninstall provider=md5
```

EXAMPLE 9 Disable metaslot

The following command disables the metaslot feature in the cryptographic framework.

```
example# cryptoadm disable metaslot
```

EXAMPLE 10 Specify metaslot to Use Specified Token as Persistent Object Store

The following command specifies that metaslot use the Venus token as the persistent object store.

```
example# cryptoadm enable metaslot token="SUNW,venus"
```

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	See below

The `start`, `stop`, and `refresh` options are Private interfaces. All other options are Evolving. The utility name is Stable.

See Also `logadm(1M)`, `svcadm(1M)`, `syslogd(1M)`, `libpkcs11(3LIB)`, `exec_attr(4)`, `prof_attr(4)`, `attributes(5)`, `smf(5)`, `random(7D)`

System Administration Guide: Security Services

Solaris Security for Developer's Guide

Notes If a hardware provider's policy was made explicitly (that is, some of its mechanisms were disabled) and the hardware provider has been detached, the policy of this hardware provider is still listed.

`cryptoadm` assumes that, minimally, a 32–bit shared object is delivered for each user-level provider. If both a 32–bit and 64–bit shared object are delivered, the two versions must provide the same functionality. The same mechanism policy applies to both.

Name cvcd – virtual console daemon

Synopsis

```
/platform/platform_name/cvcd [-a auth] [-e encr]
     [-u esp_auth]
```

Description The virtual console daemon, `cvcd`, is a server process that supports the network console provided on some platforms. The `cvcd` daemon accepts network console connections from a remote host (only one host at any given time). Console input is read from this connection and forwarded to `cvc(7D)` by way of `cvcredir(7D)`.

Similarly, console output is read from `cvcredir`(7D) and forwarded across the network console connection. If `cvcd` dies, console traffic is automatically rerouted through an internal hardware interface.

The `cvcd` daemon normally starts at system boot time. Each domain supports only one `cvcd` process at a time.

Caution – On Sun Enterprise 10000 domains, `cvcd` uses a configuration file (`/etc/ssphostname`) to determine the name of the host from which network console connections are allowed. If the remote console host is renamed, you must edit the configuration file to reflect that change.

The `cvcd` daemon supports per-socket IP Security Architecture (IPsec) through the options described below. See `ipsec(7P)`.

Options The `cvcd` daemon supports the options listed below.

`-a` *auth*	Controls the IPsec Authentication Header (AH) algorithm. *auth* can be one of `none`, `md5`, or `sha1`.
`-e` *encr*	Controls the IPsec Encapsulating Security Payload (ESP) encryption algorithm. *encr* can be one of `none`, `des`, or `3des`.
`-u` *esp_auth*	Controls the IPsec Encapsulating Security Payload (ESP) authentication algorithm. *esp_auth* can be one of `none`, `md5`, or `sha1`.

Operands The following operands are supported:

platform_name	The official Sun platform name used in packaging and code. For example, for Sun Fire 15K servers, the *platform_name* would be `SUNW,Sun-Fire-15000`.

Examples **EXAMPLE 1** Setting an IPSec Option

The command below sets the value of the IPsec Authentication Header algorithm to `md5`. As a result of this command, `cvcd` will use the HMAC-MD5 authentication algorithm.

```
# svccfg -s svc:/system/cvc setprop cvc/ah_auth = "md5"
# svccfg -s svc:/system/cvc setprop cvc/esp_encr = "none"
# svccfg -s svc:/system/cvc setprop cvc/esp_auth = "none"
```

EXAMPLE 1 Setting an IPSec Option *(Continued)*

```
# svcadm refresh svc:/system/cvc
```

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	Sun Enterprise 10000 servers, Sun Fire High-End Systems
Availability	SUNWcvc.u

See Also

svcs(1), svcadm(1M), svccfg(1M), services(4), attributes(5), smf(5), cvc(7D), cvcredir(7D), ipsec(7P)

Sun Enterprise 10000 SSP Reference Manual

System Management Services (SMS) Reference Manual

Notes

The cvcd service is managed by the service management facility, smf(5), under the fault management resource identifier (FMRI):

```
svc:/system/cvc
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M) or svccfg(1M). The service's status can be queried using the svcs(1) command.

Name datadm – maintain DAT static registry file

Synopsis

```
/usr/bin/datadm [-v] [-u] [-a service_provider.conf]
     [-r service_provider.conf]
```

Description The `datadm` utility maintains the DAT static registry file, `dat.conf(4)`.

This administrative configuration program allows uDAPL service providers to add and remove themselves to the `dat.conf` file.

You can add or remove interface adapters that a service provider supports from a system after its installation. You can use `datadm` to update the `dat.conf` file to reflect the current state of the system. A new set of interface adapters for all the service providers currently installed is regenerated.

Options The following options are supported:

`-a service_provider.conf`	Enumerate each device entry in the `service_provider.conf(4)` file into a list of interface adapters, that is, interfaces to external network that are available to uDAPL consumers.
`-r service_provider.conf`	Remove the list of interface adapters that corresponds to the device entry in the `service_provider.conf(4)` file.
`-u`	Update the `dat.conf` to reflect the current state of the system with an up to date set of interface adapters for the service providers that are currently listed in the DAT static registry.
`-v`	Display the DAT static registry file, `dat.conf`.

Examples **EXAMPLE 1** Enumerating a Device Entry

The following example enumerates a device entry in the `service_provider.conf(4)` file into interface adapters in the `dat.conf(4)` file.

Assume that `SUNW` has a service provider library that supports the device `tavor`. It has a `service_provider.conf(4)` file installed in the directory `/usr/share/dat/SUNWudaplt.conf` with a single entry as follows:

```
driver_name=tavor u1.2 nonthreadsafe default\
    udapl_tavor.so.1 SUNW.1.0 ""
```

`tavor` is an Infiniband Host Channel Adapter with two ports. Both IB ports exist in a single IB partition, `0x8001`. If an IP interface is plumbed to each port, there are two IPoIB device instances, `ibd0` and `ibd1`:

```
# ls -l /dev/ibd*
/dev/ibd0 -> /devices/pci@1/pci15b3,5a44@0/ibport@1,8001,ipib:ibd0
/dev/ibd1 -> /devices/pci@1/pci15b3,5a44@0/ibport@2,8001,ipib:ibd1
```

EXAMPLE 1 Enumerating a Device Entry *(Continued)*

Running the command, `datadm -a /usr/share/dat/SUNWudaplt.conf` appends two new entries (if they do not already exists]) in the `/etc/dat/dat.conf`file:

```
ibd0 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=tavor"
ibd1 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=tavor"
```

EXAMPLE 2 Updating the `dat.conf` to Reflect the Current State of the System

A new IB partition, 0x8002 is added to the above example covering port 1 of the Host Channel Adapter. If a new IP interface is plumbed to `port 1/partition 0x8002`, there is a third IPoIB device instance: `ibd2`.

```
# ls -l /dev/ibd*
 /dev/ibd0 -> /devices/pci@1/pci15b3,5a44@0/ibport@1,8001,ipib:ibd0
 /dev/ibd1 -> /devices/pci@1/pci15b3,5a44@0/ibport@2,8001,ipib:ibd1
 /dev/ibd2 -> /devices/pci@1/pci15b3,5a44@0/ibport@1,8002,ipib:ibd2
```

Running `datadm -u` command, updates the `/etc/dat/dat.conf` file with a new entry added reflecting the current state of the system.

`datadm -v` shows that there are now three entries in the `/etc/dat/dat.conf` file:

```
ibd0 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=tavor"
ibd1 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=tavor"
ibd2 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=tavor"
```

Files `/etc/dat/dat.conf` DAT static registry file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWudaplu
Interface Stability	Evolving

See Also `pkgadd(1M)`, `pkgrm(1M)`, `libdat(3LIB)`, `dat.conf(4)`, `service_provider.conf(4)`. `attributes(5)`

Name dcs – domain configuration server

Synopsis

```
/usr/lib/dcs [-s sessions]
     [ [-a auth] [-e encr] [-u esp_auth] ] [-l]
```

Description The Domain Configuration Server (DCS) is a daemon process that runs on Sun servers that support remote Dynamic Reconfiguration (DR) clients. It is started by the Service Management Facility (see `smf(5)`) when the first DR request is received from a client connecting to the network service `sun-dr`. After the DCS accepts a DR request, it uses the `libcfgadm(3LIB)` interface to execute the DR operation. After the operation is performed, the results are returned to the client.

The DCS listens on the network service labeled `sun-dr`. Its underlying protocol is TCP. It is invoked as a server program by the SMF using the TCP transport. The fault management resource identifier (FMRI) for DCS is:

```
svc:/platform/sun4u/dcs:default
```

If you disable this service, DR operations initiated from a remote host fail. There is no negative impact on the server.

Security for the DCS connection is provided differently based upon the architecture of the system. The SMF specifies the correct options when invoking the DCS daemon, based upon the current architecture. For all architectures, security is provided on a per-connection basis.

The DCS daemon has no security options that are applicable when used on a Sun Enterprise 10000 system. So there are no options applicable to that architecture.

The security options for Sun Fire high-end systems are based on IPsec options defined as SMF properties. These options include the `-a` *auth*, `-e` *encr*, and `-u` *esp_auth* options, and can be set using the `svccfg(1M)` command. These options must match the IPsec policies defined for DCS on the system controller. Refer to the `kmd(1M)` man page in the *System Management Services (SMS) Reference Manual*. The `kmd(1M)` man page is not part of the SunOS man page collection.

Security on SPARC Enterprise Servers is not configurable. The DCS daemon uses a platform-specific library to configure its security options when running on such systems. The `-l` option is provided by the SMF when invoking the DCS daemon on SPARC Enterprise Servers. No other security options to the DCS daemon should be used on SPARC Enterprise Servers.

Options The following options are supported:

`-a` *auth* Controls the IPsec Authentication Header (AH) algorithm. *auth* can be one of `none`, `md5`, or `sha1`.

`-e` *encr* Controls the IPsec Encapsulating Security Payload (ESP) encryption algorithm. *encr* can be one of `none`, `des`, or `3des`.

`-l` Enables the use of platform-specific security options on SPARC Enterprise Servers.

`-s` *sessions* Sets the number of active sessions that the DCS allows at any one time. When the limit is reached, the DCS stops accepting connections until active sessions complete the execution of their DR operation. If this option is not specified, a default value of 128 is used.

`-u` *esp_auth* Controls the IPsec Encapsulating Security Payload (ESP) authentication algorithm. *esp_auth* can be one of `none`, `md5`, or `sha1`.

Examples

EXAMPLE 1 Setting an IPSec Option

The following command sets the Authentication Header algorithm for the DCS daemon to use the HMAC-MD5 authentication algorithm. These settings are only applicable for using the DCS daemon on a Sun Fire high-end system.

```
# svccfg -s svc:/platform/sun4u/dcs setprop dcs/ah_auth = "md5"
# svccfg -s svc:/platform/sun4u/dcs setprop dcs/esp_encr = "none"
# svccfg -s svc:/platform/sun4u/dcs setprop dcs/esp_auth = "none"
# svcadm refresh svc:/platform/sun4u/dcs
```

Errors

The DCS uses `syslog(3C)` to report status and error messages. All of the messages are logged with the `LOG_DAEMON` facility. Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities, and informational messages are logged with the `LOG_INFO` priority. The default entries in the `/etc/syslog.conf` file log all of the DCS error messages to the `/var/adm/messages` log.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdcsu, SUNWdcsr
Interface Stability	Evolving

See Also

`svcs(1)`, `cfgadm_sbd(1M)`, `svcadm(1M)`, `svccfg(1M)`, `syslog(3C)`, `config_admin(3CFGADM)`, `libcfgadm(3LIB)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`, `dr(7d)`

Notes

The `dcs` service is managed by the service management facility, `smf(5)`, under the fault management resource identifier (FMRI):

```
svc:/platform/sun4u/dcs:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name dd – convert and copy a file

Synopsis `/usr/bin/dd [`*operand=value*`]...`

Description The `dd` utility copies the specified input file to the specified output with possible conversions. The standard input and output are used by default. The input and output block sizes may be specified to take advantage of raw physical I/O. Sizes are specified in bytes; a number may end with `k`, `b`, or `w` to specify multiplication by 1024, 512, or 2, respectively. Numbers may also be separated by `x` to indicate multiplication.

The `dd` utility reads the input one block at a time, using the specified input block size. `dd` then processes the block of data actually returned, which could be smaller than the requested block size. `dd` applies any conversions that have been specified and writes the resulting data to the output in blocks of the specified output block size.

`cbs` is used only if `ascii`, `asciib`, `unblock`, `ebcdic`, `ebcdicb`, `ibm`, `ibmb`, or `block` conversion is specified. In the first two cases, `cbs` characters are copied into the conversion buffer, any specified character mapping is done, trailing blanks are trimmed, and a `NEWLINE` is added before sending the line to output. In the last three cases, characters up to `NEWLINE` are read into the conversion buffer and blanks are added to make up an output record of size `cbs`. ASCII files are presumed to contain `NEWLINE` characters. If `cbs` is unspecified or `0`, the `ascii`, `asciib`, `ebcdic`, `ebcdicb`, `ibm`, and `ibmb` options convert the character set without changing the input file's block structure. The `unblock` and `block` options become a simple file copy.

After completion, `dd` reports the number of whole and partial input and output blocks.

Operands The following operands are supported:

`if=`*file* — Specifies the input path. Standard input is the default.

`of=`*file* — Specifies the output path. Standard output is the default. If the `seek=expr` conversion is not also specified, the output file will be truncated before the copy begins, unless `conv=notrunc` is specified. If `seek=expr` is specified, but `conv=notrunc` is not, the effect of the copy will be to preserve the blocks in the output file over which `dd` seeks, but no other portion of the output file will be preserved. (If the size of the seek plus the size of the input file is less than the previous size of the output file, the output file is shortened by the copy.)

`ibs=`*n* — Specifies the input block size in *n* bytes (default is `512`).

`obs=`*n* — Specifies the output block size in *n* bytes (default is `512`).

`bs=`*n* — Sets both input and output block sizes to *n* bytes, superseding `ibs=` and `obs=`. If no conversion other than `sync`, `noerror`, and `notrunc` is specified, each input block is copied to the output as a single block without aggregating short blocks.

`cbs=`*n*

Specifies the conversion block size for `block` and `unblock` in bytes by *n* (default is `0`). If `cbs=` is omitted or given a value of `0`, using `block` or `unblock` produces unspecified results.

This option is used only if ASCII or EBCDIC conversion is specified. For the `ascii` and `asciib` operands, the input is handled as described for the `unblock` operand except that characters are converted to ASCII before the trailing `SPACE` characters are deleted. For the `ebcdic`, `ebcdicb`, `ibm`, and `ibmb` operands, the input is handled as described for the `block` operand except that the characters are converted to EBCDIC or IBM EBCDIC after the trailing `SPACE` characters are added.

`files=`*n*

Copies and concatenates *n* input files before terminating (makes sense only where input is a magnetic tape or similar device).

`skip=`*n*

Skips *n* input blocks (using the specified input block size) before starting to copy. On seekable files, the implementation reads the blocks or seeks past them. On non-seekable files, the blocks are read and the data is discarded.

`iseek=`*n*

Seeks *n* blocks from beginning of input file before copying (appropriate for disk files, where `skip` can be incredibly slow).

`oseek=`*n*

Seeks *n* blocks from beginning of output file before copying.

`seek=`*n*

Skips *n* blocks (using the specified output block size) from beginning of output file before copying. On non-seekable files, existing blocks are read and space from the current end-of-file to the specified offset, if any, is filled with null bytes. On seekable files, the implementation seeks to the specified offset or reads the blocks as described for non-seekable files.

`count=`*n*

Copies only *n* input blocks.

`conv=`*value*[, *value*. . .]

Where *value*s are comma-separated symbols from the following list:

`ascii`
Converts EBCDIC to ASCII.

`asciib`
Converts EBCDIC to ASCII using BSD-compatible character translations.

`ebcdic`
Converts ASCII to EBCDIC. If converting fixed-length ASCII records without NEWLINEs, sets up a pipeline with `dd conv=unblock` beforehand.

`ebcdicb`
Converts ASCII to EBCDIC using BSD-compatible character translations. If converting fixed-length

ASCII records without NEWLINEs, sets up a pipeline with dd conv=unblock beforehand.

`ibm` Slightly different map of ASCII to EBCDIC. If converting fixed-length ASCII records without NEWLINEs, sets up a pipeline with dd conv=unblock beforehand.

`ibmb` Slightly different map of ASCII to EBCDIC using BSD-compatible character translations. If converting fixed-length ASCII records without NEWLINEs, sets up a pipeline with dd conv=unblock beforehand.

The `ascii` (or `asciib`), `ebcdic` (or `ebcdicb`), and `ibm` (or `ibmb`) values are mutually exclusive.

`block` Treats the input as a sequence of NEWLINE-terminated or EOF-terminated variable-length records independent of the input block boundaries. Each record is converted to a record with a fixed length specified by the conversion block size. Any NEWLINE character is removed from the input line. SPACE characters are appended to lines that are shorter than their conversion block size to fill the block. Lines that are longer than the conversion block size are truncated to the largest number of characters that will fit into that size. The number of truncated lines is reported.

`unblock` Converts fixed-length records to variable length. Reads a number of bytes equal to the conversion block size (or the number of bytes remaining in the input, if less than the conversion block size), delete all trailing SPACE characters, and append a NEWLINE character.

The `block` and `unblock` values are mutually exclusive.

`lcase` Maps upper-case characters specified by the LC_CTYPE keyword `tolower` to the corresponding lower-case character. Characters for which no mapping is specified are not modified by this conversion.

`ucase` Maps lower-case characters specified by the LC_CTYPE keyword `toupper` to the corresponding upper-case character. Characters for which no mapping is specified are not modified by this conversion.

The `lcase` and `ucase` symbols are mutually exclusive.

`swab` Swaps every pair of input bytes. If the current input record is an odd number of bytes, the last byte in the input record is ignored.

`noerror` Does not stop processing on an input error. When an input error occurs, a diagnostic message is written on standard error, followed by the current input and output block counts in the same format as used at completion. If the `sync` conversion is specified, the missing input is replaced with null bytes and processed normally. Otherwise, the input block will be omitted from the output.

`notrunc` Does not truncate the output file. Preserves blocks in the output file not explicitly written by this invocation of `dd`. (See also the preceding `of=`*file* operand.)

`sync` Pads every input block to the size of the `ibs=` buffer, appending null bytes. (If either `block` or `unblock` is also specified, appends `SPACE` characters, rather than null bytes.)

If operands other than `conv=` are specified more than once, the last specified `operand=`*value* is used.

For the `bs=`, `cbs=`, `ibs=`, and `obs=` operands, the application must supply an expression specifying a size in bytes. The expression, `expr`, can be:

1. a positive decimal number
2. a positive decimal number followed by `k`, specifying multiplication by 1024
3. a positive decimal number followed by `b`, specifying multiplication by 512
4. two or more positive decimal numbers (with or without `k` or `b`) separated by `x`, specifying the product of the indicated values.

All of the operands will be processed before any input is read.

Usage See `largefile(5)` for the description of the behavior of `dd` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Copying from one tape drive to another

The following example copies from tape drive `0` to tape drive `1`, using a common historical device naming convention.

EXAMPLE 1 Copying from one tape drive to another *(Continued)*

```
example% dd if=/dev/rmt/0h  of=/dev/rmt/1h
```

EXAMPLE 2 Stripping the first 10 bytes from standard input

The following example strips the first 10 bytes from standard input:

```
example% dd ibs=10  skip=1
```

EXAMPLE 3 Reading a tape into an ASCII file

This example reads an EBCDIC tape blocked ten 80-byte EBCDIC card images per block into the ASCII file x:

```
example% dd if=/dev/tape of=x ibs=800 cbs=80 conv=ascii,lcase
```

EXAMPLE 4 Using conv=sync to write to tape

The following example uses `conv=sync` when writing to a tape:

```
example% tar cvf - . | compress | dd obs=1024k of=/dev/rmt/0 conv=sync
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `dd`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Exit Status

The following exit values are returned:

`0` The input file was copied successfully.

`>0` An error occurred.

If an input error is detected and the `noerror` conversion has not been specified, any partial output block will be written to the output file, a diagnostic message will be written, and the copy operation will be discontinued. If some other error is detected, a diagnostic message will be written and the copy operation will be discontinued.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Standard

See Also

`cp(1)`, `sed(1)`, `tr(1)`, `attributes(5)`, `environ(5)`, `largefile(5)`, `standards(5)`

Diagnostics

`f+p records in(out)` numbers of full and partial blocks read(written)

Notes Do not use `dd` to copy files between file systems having different block sizes.

Using a blocked device to copy a file will result in extra nulls being added to the file to pad the final block to the block boundary.

When `dd` reads from a pipe, using the `ibs=X` and `obs=Y` operands, the output will always be blocked in chunks of size Y. When `bs=Z` is used, the output blocks will be whatever was available to be read from the pipe at the time.

When using `dd` to copy files to a tape device, the file size must be a multiple of the device sector size (for example, 512 Kbyte). To copy files of arbitrary size to a tape device, use `tar(1)` or `cpio(1)`.

For `SIGINT`, `dd` writes status information to standard error before exiting. It takes the standard action for all other signals.

Name devattr – display device attributes

Synopsis

```
devattr [-v] device [attribute]...
```

Description `devattr` displays the values for a device's attributes. The display can be presented in two formats. Used without the `-v` option, only the attribute values are shown. Used with the `-v` option, the attributes are shown in an *attribute*=*value* format. When no attributes are given on the command line, all attributes for the specified device are displayed in alphabetical order by attribute name. If attributes are given on the command line, only those attributes are shown, displayed in command line order.

Options The following options are supported:

`-v` Specifies verbose format. Attribute values are displayed in an *attribute*=*value* format.

Operands The following operands are supported:

attribute Defines which attribute, or attributes, should be shown. Default is to show all attributes for a device. See the `putdev(1M)` manual page for a complete listing and description of available attributes.

device Defines the device whose attributes should be displayed. Can be the pathname of the device or the device alias.

Exit Status The following exit values are returned:

`0` successful completion.

`1` Command syntax was incorrect, invalid option was used, or an internal error occurred.

`2` Device table could not be opened for reading.

`3` Requested device could not be found in the device table.

`4` Requested attribute was not defined for the specified device.

Files `/etc/device.tab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `getdev(1M)`, `putdev(1M)`, `attributes(5)`

Name devfree – release devices from exclusive use

Synopsis

```
devfree key [device]...
```

Description `devfree` releases devices from exclusive use. Exclusive use is requested with the command `devreserv`.

When `devfree` is invoked with only the *key* argument, it releases all devices that have been reserved for that *key*. When called with *key* and *device* arguments, `devfree` releases the specified devices that have been reserved with that *key*.

Operands The following operands are supported:

device Defines device that this command will release from exclusive use. *device* can be the pathname of the device or the device alias.

key Designates the unique key on which the device was reserved.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Command syntax was incorrect, an invalid option was used, or an internal error occurred.

`2` Device table or device reservation table could not be opened for reading.

`3` Reservation release could not be completely fulfilled because one or more of the devices was not reserved or was not reserved on the specified key.

Files

```
/etc/device.tab
```

```
/etc/devlkfile
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `devreserv(1M)`, `attributes(5)`

Notes The commands `devreserv` and `devfree` are used to manage the availability of devices on a system. These commands do not place any constraints on the access to the device. They serve only as a centralized bookkeeping point for those who wish to use them. Processes that do not use `devreserv` may concurrently use a device with a process that has reserved that device.

Name devfsadm, devfsadmd – administration command for /dev

Synopsis

```
/usr/sbin/devfsadm [-C] [-c device_class] [-i driver_name]
     [ -n] [-r root_dir] [-s] [-t table_file] [-v]

/usr/lib/devfsadm/devfsadmd
```

Description `devfsadm`(1M) maintains the `/dev` namespace. It replaces the previous suite of `devfs` administration tools including `drvconfig`(1M), `disks`(1M), `tapes`(1M), `ports`(1M), `audlinks`(1M), and `devlinks`(1M).

The default operation is to attempt to load every driver in the system and attach to all possible device instances. Next, `devfsadm` creates logical links to device nodes in `/dev` and `/devices` and loads the device policy.

`devfsadmd`(1M) is the daemon version of `devfsadm`(1M). The daemon is started during system startup and is responsible for handling both reconfiguration boot processing and updating `/dev` and `/devices` in response to dynamic reconfiguration event notifications from the kernel.

For compatibility purposes, `drvconfig`(1M), `disks`(1M), `tapes`(1M), `ports`(1M), `audlinks`(1M), and `devlinks`(1M) are implemented as links to `devfsadm`.

In addition to managing `/dev`, `devfsadm` also maintains the `path_to_inst`(4) database.

Options The following options are supported:

`-C`	Cleanup mode. Prompt `devfsadm` to cleanup dangling `/dev` links that are not normally removed. If the `-c` option is also used, `devfsadm` only cleans up for the listed devices' classes.
`-c` *device_class*	Restrict operations to devices of class *device_class*. Solaris defines the following values for *device_class*: `disk`, `tape`, `port`, `audio`, and `pseudo`. This option might be specified more than once to specify multiple device classes.
`-i` *driver_name*	Configure only the devices for the named driver, *driver_name*.
`-n`	Do not attempt to load drivers or add new nodes to the kernel device tree.
`-s`	Suppress any changes to `/dev`. This is useful with the `-v` option for debugging.
`-t` *table_file*	Read an alternate `devlink.tab` file. `devfsadm` normally reads `/etc/devlink.tab`.
`-r` *root_dir*	Presume that the `/dev` directory trees are found under *root_dir*, not directly under `root` (`/`). No other use or assumptions are made about *root_dir*.
`-v`	Print changes to `/dev` in verbose mode.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

Files

`/devices`	device nodes directory
`/dev`	logical symbolic links to /devices
`/usr/lib/devfsadm/devfsadmd`	devfsadm daemon
`/dev/.devfsadm_dev.lock`	update lock file
`/dev/.devfsadm_daemon.lock`	daemon lock file
`/etc/security/device_policy`	device policy file
`/etc/security/extra_privs`	additional device privileges

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `svcs(1)`, `add_drv(1M)`, `modinfo(1M)`, `modload(1M)`, `modunload(1M)`, `rem_drv(1M)`, `svcadm(1M)`, `tapes(1M)`, `path_to_inst(4)`, `attributes(5)`, `privileges(5)`, `smf(5)`, `devfs(7FS)`

Notes This document does not constitute an API. The `/devices` directory might not exist or might have different contents or interpretations in a future release. The existence of this notice does not imply that any other documentation that lacks this notice constitutes an API.

`devfsadm` no longer manages the `/devices` name space. See `devfs(7FS)`.

The device configuration service is managed by the service management facility, `smf(5)`, under the service identifier, and can be used to start `devfsadm` during reconfiguration boot by:

```
svc:/system/device/local:default
```

Otherwise, `devfsadm` is started by:

```
svc:/system/sysevent:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name device_remap – administer the Solaris I/O remapping feature

Synopsis `/usr/platform/sun4v/sbin/device_remap [-v | -R` *dir*`]`

Description Certain multi-node sun4v platforms, such as T5440 and T5240 servers, have an integrated PCI topology that cause the I/O device paths to change in a CPU node failover condition. The device remapping script, `device_remap`, remaps the device paths in `/etc/path_to_inst` file and the symlinks under `/dev` to match the hardware.

Options The following options are supported:

`-v`
Displays the `/etc/path_to_inst` and `/dev` symlink changes.

`-R` *dir*
Perform remapping on the `/etc/path_to_inst` and `/etc/path_to_inst` files in the root image at *dir*.

Usage The primary function of `device_remap` is to remap the device paths in the `/etc/path_to_inst` file and the symlinks under `/dev` in a CPU node failover condition to match the hardware.

After adding CPU node(s) or removing CPU node(s), boot the system to the OBP prompt and use the following procedure:

1. Boot either the failsafe miniroot using: `boot -F failsafe` , or an install miniroot using `boot net -s` or similar command.
2. Mount the root disk as `/mnt`.
3. Change directory to the mounted root disk:

   ```
   # cd /mnt
   ```

4. Run `device_remap` script:

   ```
   # /mnt/usr/platform/sun4v/sbin/device_remap
   ```

5. Boot the system from disk.

All the error messages are self-explanatory, except for the error message “missing ioaliases node” which means the firmware on the system does not support device remapping.

Examples **EXAMPLE 1** Displaying Changes Following Failover

The following command displays the `path_to_inst` and `/dev` changes following a CPU node failover.

```
# device_remap -v
```

EXAMPLE 2 Changing Directory Prior to Any Changes

The following command changes the directory on which the boot image is mounted prior to making any changes.

```
# device_remap -R /newroot
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkvm.v
Interface Stability	Unstable

See Also

`boot(1M)`, `attributes(5)`

Name devinfo – print device specific information

Synopsis

```
/usr/sbin/devinfo -i device
/usr/sbin/devinfo -p device
```

Description The `devinfo` command is used to print device specific information about disk devices on standard out. The command can only be used by the superuser.

Options

`-i` Prints the following device information:

- Device name
- Software version (not supported and prints as `0`)
- Drive id number (not supported and prints as `0`)
- Device blocks per cylinder
- Device bytes per block
- Number of device partitions with a block size greater than zero

`-p` Prints the following device partition information:

- Device name
- Device major and minor numbers (in hexadecimal)
- Partition start block
- Number of blocks allocated to the partition
- Partition flag
- Partition tag

This command is used by various other commands to obtain device specific information for the making of file systems and determining partition information. If the device cannot be opened, an error message is reported.

Operands *device* Device name.

Exit Status

`0` Successful operation.

`2` Operation failed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `prtvtoc(1M)`, `attributes(5)`

Name devlinks – adds /dev entries for miscellaneous devices and pseudo-devices

Synopsis

```
/usr/sbin/devlinks [-d] [-r rootdir] [-t table-file]
```

Description devfsadm(1M) is now the preferred command for /dev and /devices and should be used instead of devlinks.

devlinks creates symbolic links from the /dev directory tree to the actual block- and character-special device nodes under the /devices directory tree. The links are created according to specifications found in the *table-file* (by default /etc/devlink.tab).

devlinks is called each time the system is reconfiguration-booted, and can only be run after drvconfig(1M) is run.

The *table-file* (normally /etc/devlink.tab) is an ASCII file, with one line per record. Comment lines, which must contain a hash character ('#') as their first character, are allowed. Each entry must contain at least two fields, but may contain three fields. Fields are separated by single TAB characters.

The fields are:

devfs-spec Specification of devinfo nodes that will have links created for them. This specification consists of one or more keyword-value pairs, where the keyword is separated from the value by an equal-sign ('='), and keyword-value pairs are separated from one another by semicolons.

The possible keywords are:

type The devinfo device type. Possible values are specified in ddi_create_minor_node(9F)

name The name of the node. This is the portion of the /devices tree entry name that occurs before the first '@' or ':' character.

addr[*n*] The address portion of a node name. This is the portion of a node name that occurs between the '@' and the ':' characters. It is possible that a node may have a name without an address part, which is the case for many of the pseudo-device nodes. If a number is given after the *addr* it specifies a match of a particular comma-separated subfield of the address field: *addr1* matches the first subfield, *addr2* matches the second, and so on. *addr0* is the same as *addr* and matches the whole field.

minor[*n*] The minor portion of a node name – the portion of the name after the ':'. As with *addr* above, a number after the *minor* keyword specifies a subfield to match.

Of these four specifications, only the *type* specification must always be present.

name Specification of the `/dev` links that correspond to the devinfo nodes. This field allows `devlinks` to determine matching `/dev` names for the `/devices` nodes it has found. The specification of this field uses escape-sequences to allow portions of the `/devices` name to be included in the `/dev` name, *or* to allow a counter to be used in creating node names. If a counter is used to create a name, the portion of the name before the counter must be specified absolutely, and all names in the `/dev/`-subdirectory that match (up to and including the counter) are considered to be subdevices of the same device. This means that they should all point to the same directory, `name` and `address` under the `/devices/`-tree

The possible escape-sequences are:

`\D` Substitute the `device-name` (name) portion of the corresponding devinfo node-name.

`\A`*n* Substitute the *n*th component of the `address` component of the corresponding devinfo node name. Sub-components are separated by commas, and sub-component `0` is the whole `address` component.

`\M`*n* Substitute the *n*th sub-component of the `minor` component of the corresponding devinfo node name. Sub-components are separated by commas, and sub-component `0` is the whole `minor` component.

`\N`*n* Substitute the value of a 'counter' starting at *n*. There can be only one counter for each dev-spec, and counter-values will be selected so they are as low as possible while not colliding with already-existing link names.

In a dev-spec the `counter` sequence should not be followed by a digit, either explicitly or as a result of another escape-sequence expansion. If this occurs, it would not be possible to correctly match already-existing links to their counter entries, since it would not be possible to unambiguously parse the already-existing `/dev`-name.

extra-dev-link Optional specification of an extra `/dev` link that points to the initial `/dev` link (specified in field 2). This field may contain a `counter` escape-sequence (as described for the *dev-spec* field) but may not contain any of the other escape-sequences. It provides a way to specify an alias of a particular `/dev` name.

Options The following options are supported:

`-d` Debugging mode – print out all `devinfo` nodes found, and indicate what links would be created, but do not do anything.

`-r` *rootdir* Use *rootdir* as the root of the `/dev` and `/devices` directories under which the device nodes and links are created. Changing the root directory does not change the location of the `/etc/devlink.tab` default table, nor is the root directory applied to the filename supplied to the `-t` option.

`-t` *table-file* Set the table file used by `devlinks` to specify the links that must be created. If this option is not given, `/etc/devlink.tab` is used. This option gives a way to instruct `devlinks` just to perform a particular piece of work, since just the links-types that `devlinks` is supposed to create can be specified in a command-file and fed to `devlinks`.

Errors If `devlinks` finds an error in a line of the *table-file* it prints a warning message on its standard output and goes on to the next line in the *table-file* without performing any of the actions specified by the erroneous rule.

If it cannot create a link for some filesystem-related reason it prints an error-message and continues with the current rule.

If it cannot read necessary data it prints an error message and continues with the next *table-file* line.

Examples **EXAMPLE 1** Using the `/etc/devlink.tab` Fields

The following are examples of the `/etc/devlink.tab` fields:

```
type=pseudo;name=win    win\M0
type=ddi_display    framebuffer/\M0    fb\N0
```

The first example states that all devices of type `pseudo` with a name component of `win` will be linked to `/dev/win`*x,* where *x* is the minor-component of the *devinfo-name* (this is always a single-digit number for the `win` driver).

The second example states that all devinfo nodes of type `ddi_display` will be linked to entries under the `/dev/framebuffer` directory, with names identical to the entire minor component of the `/devices` name. In addition an extra link will be created pointing from `/dev/fb`*n* to the entry under `/dev/framebuffer`. This entry will use a counter to end the name.

Files

`/dev`	entries for the miscellaneous devices for general use
`/devices`	device nodes
`/etc/devlink.tab`	the default rule-file

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also devfsadm(1M), attributes(5), devfs(7FS), ddi_create_minor_node(9F)

Bugs It is very easy to construct mutually-contradictory link specifications, or specifications that can never be matched. The program does not check for these conditions.

Name devnm – device name

Synopsis

```
/usr/sbin/devnm name [name]...
```

Description The `devnm` command identifies the special file associated with the mounted file system where the argument *name* resides. One or more *name* can be specified.

Examples **EXAMPLE 1** Using the devnm Command

Assuming that `/usr` is mounted on `/dev/dsk/c0t3d0s6`, the following command :

```
/usr/sbin/devnm /usr
```

produces:

```
/dev/dsk/c0t3d0s6 /usr
```

Files

```
/dev/dsk/*
```

```
/etc/mnttab
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `mnttab(4)`, `attributes(5)`

Name devreserv – reserve devices for exclusive use

Synopsis `devreserv [`*key* `[`*device-list*`]...]`

Description `devreserv` reserves devices for exclusive use. When the device is no longer required, use `devfree` to release it.

`devreserv` reserves at most one device per *device-list*. Each list is searched in linear order until the first available device is found. If a device cannot be reserved from each list, the entire reservation fails.

When `devreserv` is invoked without arguments, it lists the devices that are currently reserved and shows to which key it was reserved. When `devreserv` is invoked with only the *key* argument, it lists the devices that are currently reserved to that key.

Operands The following operands are supported:

device-list Defines a list of devices that `devreserv` will search to find an available device. The list must be formatted as a single argument to the shell.

key Designates a unique key on which the device will be reserved. The key must be a positive integer.

Examples **EXAMPLE 1** Reserving a Floppy Disk and a Cartridge Tape

The following example reserves a floppy disk and a cartridge tape:

```
$ key=$$
$ echo "The current Process ID is equal to: $key"
  The Current Process ID is equal to: 10658
$ devreserv $key diskette1 ctape1
```

EXAMPLE 2 Listing All Devices Currently Reserved

The following example lists all devices currently reserved:

```
$ devreserv
  disk1         2423
  diskette1     10658
  ctape1        10658
```

EXAMPLE 3 Listing All Devices Currently Reserved to a Particular Key

The following example lists all devices currently reserved to a particular key:

```
$ devreserv $key
  diskette1
  ctape1
```

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Command syntax was incorrect, an invalid was option used, or an internal error occurred.

`2` Device table or device reservation table could not be opened for reading.

`3` Device reservation request could not be fulfilled.

Files `/etc/device.tab`

`/etc/devlkfile`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `devfree(1M)`, `attributes(5)`

Notes The commands `devreserv` and `devfree` are used to manage the availability of devices on a system. Their use is on a participatory basis and they do not place any constraints on the actual access to the device. They serve as a centralized bookkeeping point for those who wish to use them. Devices which have been reserved cannot be used by processes which utilize the device reservation functions until the reservation has been canceled. However, processes that do not use device reservation may use a device that has been reserved since such a process would not have checked for its reservation status.

Name df – displays number of free disk blocks and free files

Synopsis

```
/usr/bin/df [-F FSType] [-abeghklntVvZ]
      [-o FSType-specific_options]
      [block_device | directory | file | resource ...]

/usr/xpg4/bin/df [-F FSType] [-abeghklnPtVZ]
      [-o FSType-specific_options]
      [block_device | directory | file | resource ...]
```

Description The `df` utility displays the amount of disk space occupied by mounted or unmounted file systems, the amount of used and available space, and how much of the file system's total capacity has been used. The file system is specified by device, or by referring to a file or directory on the specified file system.

Used without operands or options, `df` reports on all mounted file systems.

`df` may not be supported for all *FSTypes*.

If `df` is run on a networked mount point that the automounter has not yet mounted, the file system size will be reported as zero. As soon as the automounter mounts the file system, the sizes will be reported correctly.

Options The following options are supported for both `/usr/bin/df` and `/usr/xpg4/bin/df`:

`-a`
Reports on all file systems including ones whose entries in `/etc/mnttab` (see `mnttab(4)`) have the `ignore` option set.

`-b`
Prints the total number of kilobytes free.

`-e`
Prints only the number of files free.

`-F` *FSType*
Specifies the *FSType* on which to operate. The `-F` option is intended for use with unmounted file systems. The *FSType* should be specified here or be determinable from `/etc/vfstab` (see `vfstab(4)`) by matching the *directory*, *block_device*, or *resource* with an entry in the table, or by consulting `/etc/default/fs`. See `default_fs(4)`.

`-g`
Prints the entire `statvfs(2)` structure. This option is used only for mounted file systems. It can not be used with the `-o` option. This option overrides the `-b`, `-e`, `-k`, `-n`, `-P`, and `-t` options.

`-h`
Like `-k`, except that sizes are in a more human readable format. The output consists of one line of information for each specified file system. This information includes the file system name, the total space allocated in the file system, the amount of space allocated to existing

files, the total amount of space available for the creation of new files by unprivileged users, and the percentage of normally available space that is currently allocated to all files on the file system. All sizes are scaled to a human readable format, for example, `14K`, `234M`, `2.7G`, or `3.0T`. Scaling is done by repetitively dividing by `1024`.

This option overrides the `-b`, `-e`, `-g`, `-k`, `-n`, `-t`, and `-V` options. This option only works on mounted filesystems and can not be used together with `-o` option.

`-k`
Prints the allocation in kbytes. The output consists of one line of information for each specified file system. This information includes the file system name, the total space allocated in the file system, the amount of space allocated to existing files, the total amount of space available for the creation of new files by unprivileged users, and the percentage of normally available space that is currently allocated to all files on the file system. This option overrides the `-b`, `-e`, `-n`, and `-t` options.

`-l`
Reports on local file systems only. This option is used only for mounted file systems. It can not be used with the `-o` option.

`-n`
Prints only the *FSType* name. Invoked with no operands, this option prints a list of mounted file system types. This option is used only for mounted file systems. It can not be used with the `-o` option.

`-o` *FSType-specific_options*
Specifies *FSType-specific* options. These options are comma-separated, with no intervening spaces. See the manual page for the *FSType-specific* command for details.

`-t`
Prints full listings with totals. This option overrides the `-b`, `-e`, and `-n` options.

`-V`
Echoes the complete set of file system specific command lines, but does not execute them. The command line is generated by using the options and operands provided by the user and adding to them information derived from `/etc/mnttab`, `/etc/vfstab`, or `/etc/default/fs`. This option may be used to verify and validate the command line.

`-Z`
Displays mounts in all visible zones. By default, `df` only displays mounts located within the current zone. This option has no effect in a non-global zone.

/usr/bin/df The following option is supported for `/usr/bin/df` only:

`-v`
Like `-k`, except that sizes are displayed in multiples of the smallest block size supported by each specified file system.

The output consists of one line of information for each file system. This one line of information includes the following:

- the file system's mount point
- the file system's name
- the total number of blocks allocated to the file system
- the number of blocks allocated to existing files
- the number of blocks available for the creation of new files by unprivileged users
- the percentage of blocks in use by files

/usr/xpg4/bin/df The following option is supported for `/usr/xpg4/bin/df` only:

`-P`
Same as `-h` except in 512-byte units.

Operands The `df` utility interprets operands according to the following precedence: *block_device*, *directory*, *file*, *resource*. The following operands are supported:

block_device
Represents a block special device (for example, `/dev/dsk/c1d0s7`).

directory
Represents a valid directory name. `df` reports on the file system that contains *directory*.

file
Represents a valid file name. `df` reports on the file system that contains *file*.

resource
Represents an NFS resource name.

Usage See `largefile(5)` for the description of the behavior of `df` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Executing the `df` command

The following example shows the `df` command and its output:

```
example% /usr/bin/df

/                  (/dev/dsk/c0t0d0s0 ):  287530 blocks    92028 files
/system/contract   (ctfs              ):       0 blocks 2147483572 files
/system/object     (objfs             ):       0 blocks 2147483511 files
/usr               (/dev/dsk/c0t0d0s6 ): 1020214 blocks   268550 files
/proc              (/proc             ):       0 blocks      878 files
/dev/fd            (fd                ):       0 blocks        0 files
/etc/mnttab        (mnttab            ):       0 blocks        0 files
/var/run           (swap              ):  396016 blocks     9375 files
/tmp               (swap              ):  396016 blocks     9375 files
/opt               (/dev/dsk/c0t0d0s5 ):  381552 blocks    96649 files
```

EXAMPLE 1 Executing the `df` command *(Continued)*

```
/export/home       (/dev/dsk/c0t0d0s7 ):  434364 blocks   108220 files
```

where the columns represent the mount point, device (or "filesystem", according to `df -k`), free blocks, and free files, respectively. For contract file systems, `/system/contract` is the mount point, `ctfs` is the contract file system (used by SMF) with 0 free blocks and 2147483582(INTMAX-1) free files. For object file systems, `/system/object` is the mount point, `objfs` is the object file system (see `objfs(7FS)`) with 0 free blocks and 2147483511 free files.

EXAMPLE 2 Writing Portable Information About the `/usr` File System

The following example writes portable information about the `/usr` file system:

```
example% /usr/xpg4/bin/df -P /usr
```

EXAMPLE 3 Writing Portable Information About the `/usr/src` file System

Assuming that `/usr/src` is part of the `/usr` file system, the following example writes portable information :

```
example% /usr/xpg4/bin/df -P /usr/src
```

EXAMPLE 4 Using `df` to Display Inode Usage

The following example displays inode usage on all `ufs` file systems:

```
example%/usr/bin/df -F ufs -o i
```

Environment Variables

`SYSV3`
: This variable is used to override the default behavior of `df` and provide compatibility with INTERACTIVE UNIX System and SCO UNIX installation scripts. As the `SYSV3` variable is provided for compatibility purposes only, it should not be used in new scripts.

When set, any header which normally displays "files" will now display "nodes". See `environ(5)` for descriptions of the following environment variables that affect the execution of `df`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Files

`/dev/dsk/*`
Disk devices

`/etc/default/fs`
Default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`, where `LOCAL` is the default partition for a command if no FSType is specified.

`/etc/mnttab`
Mount table

`/etc/vfstab`
List of default parameters for each file system

Attributes

See `attributes(5)` for descriptions of the following attributes:

/usr/bin/df

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs

/usr/xpg4/bin/df

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxcu4
Interface Stability	Committed
Standard	See `standards(5)`.

See Also

`find(1)`, `df_ufs(1M)`, `mount(1M)`, `statvfs(2)`, `default_fs(4)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`, `environ(5)`, `largefile(5)`, `standards(5)`, `objfs(7FS)`

Notes

If UFS logging is enabled on a file system, the disk space used for the log is reflected in the `df` report. The log is allocated from free blocks on the file system, and it is sized approximately 1 Mbyte per 1 Gbyte of file system, up to 256 Mbytes. The log size may be larger (up to a maximum of 512 Mbytes) depending on the number of cylinder groups present in the file system.

Name dfmounts – display mounted resource information

Synopsis

```
dfmounts [-F FSType] [-h] [-o specific_options]
     [restriction]...
```

Description `dfmounts` shows the local resources shared through a distributed file system *FSType* along with a list of clients that have the resource mounted. If *restriction* is not specified, `dfmounts` shows file systems that are currently shared on any NFS server. *specific_options* as well as the availability and semantics of *restriction* are specific to particular distributed file system types.

If `dfmounts` is entered without arguments, remote resources currently mounted on the local system are displayed, regardless of file system type. However, the `dfmounts` command does not display the names of NFS Version 4 clients.

`dfmounts` Output The output of `dfmounts` consists of an optional header line (suppressed with the `-h` flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server pathname clients ...

where:

resource Specifies the resource name that must be given to the `mount(1M)` command.

server Specifies the system from which the resource was mounted.

pathname Specifies the pathname that must be given to the `share(1M)` command.

clients Is a comma-separated list of systems that have mounted the resource. Clients are listed in the form *domain.*, *domain.system*, or *system*, depending on the file system type.

A field can be null. Each null field is indicated by a hyphen (–) unless the remainder of the fields on the line are also null, in which case the hyphen can be omitted.

Fields with whitespace are enclosed in quotation marks (`" "`).

Options

`-F` *FSType* Specify filesystem type. Defaults to the first entry in `/etc/dfs/fstypes`. `Note`: currently the only valid *FSType* is `nfs`.

`-h` Suppress header line in output.

`-o` *specific_options* Specify options specific to the filesystem provided by the `-F` option. `Note`: currently no options are supported.

Files

`/etc/dfs/fstypes` file system types

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also dfshares(1M), mount(1M), share(1M), unshare(1M), attributes(5)

Name dfmounts_nfs – display mounted NFS resource information

Synopsis

```
dfmounts [-F nfs] [-h] [server]...
```

Description `dfmounts` shows the local resources shared through NFS, along with the list of clients that have mounted the resource. The `-F` flag may be omitted if NFS is the only file system type listed in the file `/etc/dfs/fstypes`.

`dfmounts` without options, displays all remote resources mounted on the local system, regardless of file system type.

The output of `dfmounts` consists of an optional header line (suppressed with the `-h` flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server pathname clients ...

where

resource Does not apply to NFS. Printed as a hyphen (-).

server Specifies the system from which the resource was mounted.

pathname Specifies the pathname that must be given to the `share(1M)` command.

clients Is a comma-separated list of systems that have mounted the resource.

Options

`-F nfs` Specifies the `nfs`-FSType.

`-h` Suppress header line in output.

server Displays information about the resources mounted from each server, where *server* can be any system on the network. If no server is specified, the *server* is assumed to be the local system.

Files `/etc/dfs/fstypes`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfscu

See Also `mount(1M)`, `share(1M)`, `unshare(1M)`, `attributes(5)`

Name dfshares – list available resources from remote or local systems

Synopsis

```
dfshares [-F FSType] [-h] [-o specific_options] [server]...
```

Description `dfshares` provides information about resources available to the host through a distributed file system of type *FSType*. *specific_options* as well as the semantics of *server* are specific to particular distributed file systems.

If `dfshares` is entered without arguments, all resources currently shared on the local system are displayed, regardless of file system type.

The output of `dfshares` consists of an optional header line (suppressed with the `-h` flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server access transport

where

resource Specifies the resource name that must be given to the `mount(1M)` command.

server Specifies the name of the system that is making the resource available.

access Specifies the access permissions granted to the client systems, either `ro` (for read-only) or `rw` (for read/write). If `dfshares` cannot determine access permissions, a hyphen (–) is displayed.

transport Specifies the transport provider over which the resource is shared.

A field may be null. Each null field is indicated by a hyphen (–) unless the remainder of the fields on the line are also null; in which case, the hyphen may be omitted.

Options

`-F` *FSType* Specify filesystem type. Defaults to the first entry in `/etc/dfs/fstypes`.

`-h` Suppress header line in output.

`-o` *specific_options* Specify options specific to the filesystem provided by the `-F` option.

Files `/etc/dfs/fstypes`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `dfmounts(1M)`, `mount(1M)`, `share(1M)`, `unshare(1M)`, `attributes(5)`

Name dfshares_nfs – list available NFS resources from remote systems

Synopsis

```
dfshares [-F nfs] [-h] [server]...
```

Description `dfshares` provides information about resources available to the host through NFS. The `-F` flag may be omitted if NFS is the first file system type listed in the file `/etc/dfs/fstypes`.

The query may be restricted to the output of resources available from one or more servers.

`dfshares` without arguments displays all resources shared on the local system, regardless of file system type.

Specifying *server* displays information about the resources shared by each server. *Server* can be any system on the network. If no server is specified, then *server* is assumed to be the local system.

The output of `dfshares` consists of an optional header line (suppressed with the `-h` flag) followed by a list of lines containing whitespace-separated fields. For each resource, the fields are:

resource server access transport

where

resource Specifies the resource name that must be given to the `mount(1M)` command.

server Specifies the system that is making the resource available.

access Specifies the access permissions granted to the client systems; however, `dfshares` cannot determine this information for an NFS resource and populates the field with a hyphen (-).

transport Specifies the transport provider over which the *resource* is shared; however, `dfshares` cannot determine this information for an NFS resource and populates the field with a hyphen (-).

A field may be null. Each null field is indicated by a hyphen (-) unless the remainder of the fields on the line are also null; in which case, the hyphen may be omitted.

Options

`-F nfs` Specify the NFS file system type

`-h` Suppress header line in output.

Files `/etc/dfs/fstypes`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfscu

See Also mount(1M), share(1M), unshare(1M), attributes(5)

Name df_ufs – report free disk space on ufs file systems

Synopsis `df -F ufs [`*generic_options*`] [-o i] [`*directory* | *special*`]`

Description `df` displays the amount of disk space occupied by `ufs` file systems, the amount of used and available space, and how much of the file system's total capacity has been used.The amount of space reported as used and available is less than the amount of space in the file system; this is because the system reserves a fraction of the space in the file system to allow its file system allocation routines to work well. The amount reserved is typically about 10%; this can be adjusted using `tunefs(1M)`. When all the space on the file system except for this reserve is in use, only the superuser can allocate new files and data blocks to existing files. When the file system is overallocated in this way, `df` might report that the file system is more than 100% utilized.If neither *directory* nor *special* is specified, `df` displays information for all mounted ufs file systems.

Options The following options are supported:

generic_options Options supported by the generic `df` command. See `df(1M)` for a description of these options.

`-o` Specify `ufs` file system specific options. The available option is:

`i` Report the number of used and free inodes. This option can not be used with *generic_options*.

Files `/etc/mnttab` list of file systems currently mounted

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu, SUNWxcu4

See Also `df(1M)`, `fsck(1M)`, `fstyp(1M)`, `tunefs(1M)`, `mnttab(4)`, `attributes(5)`, `ufs(7FS)`,

Notes `df` calculates its results differently for mounted and unmounted file systems. For unmounted systems, the numbers reflect the 10% reservation. This reservation is not reflected in `df` output for mounted file systems. For this reason, the available space reported by the generic command can differ from the available space reported by this module.

`df` might report remaining capacity even though `syslog` warns `filesystem full`. This issue can occur because `df` only uses the available fragment count to calculate available space, but the file system requires contiguous sets of fragments for most allocations.

If you suspect that you have exhausted contiguous fragments on your file system, you can use the `fstyp(1M)` utility with the `-v` option. In the `fstyp` output, look at the `nbfree` (number of blocks free) and `nffree` (number of fragments free) fields. On unmounted filesystems, you can use `fsck(1M)` and observe the last line of output, which reports, among other items, the number of fragments and the degree of fragmentation. See `fsck(1M)`.

Name dhcpagent – Dynamic Host Configuration Protocol (DHCP) client daemon

Synopsis

```
dhcpagent [-a] [ -d n] [-f] [-v]
```

Description `dhcpagent` implements the client half of the Dynamic Host Configuration Protocol (DHCP) for machines running Solaris software.

The `dhcpagent` daemon obtains configuration parameters for the client (local) machine's network interfaces from a DHCP server. These parameters may include a lease on an IP address, which gives the client machine use of the address for the period of the lease, which may be infinite. If the client wishes to use the IP address for a period longer than the lease, it must negotiate an extension using DHCP. For this reason, `dhcpagent` must run as a daemon, terminating only when the client machine powers down.

For IPv4, the `dhcpagent` daemon is controlled through `ifconfig(1M)` in much the same way that the `init(1M)` daemon is controlled by `telinit(1M)`. `dhcpagent` can be invoked as a user process, albeit one requiring root privileges, but this is not necessary, as `ifconfig(1M)` will start it automatically.

For IPv6, the `dhcpagent` daemon is invoked automatically by `in.ndpd(1M)`. It can also be controlled through `ifconfig(1M)`, if necessary.

When invoked, `dhcpagent` enters a passive state while it awaits instructions from `ifconfig(1M)` or `in.ndpd(1M)`. When it receives a command to configure an interface, it starts DHCP on the interface. Once DHCP is complete, `dhcpagent` can be queried for the values of the various network parameters. In addition, if DHCP was used to obtain a lease on an address for an interface, it configures the address for use. When a lease is obtained, it is automatically renewed as necessary. If the lease cannot be renewed, `dhcpagent` will unconfigure the address, and attempt to acquire a new address lease on the interface. `dhcpagent` monitors system suspend/resume events and will validate any non-permanent leases with the DHCP server upon resume. Similarly, `dhcpagent` monitors link up/down events and will validate any non-permanent leases with the DHCP server when the downed link is brought back up.

For IPv4, if the configured interface is found to be unplumbed, marked down, or to have a different IP address, subnet mask, or broadcast address from those obtained from DHCP, the interface is abandoned by DHCP control.

For IPv6, `dhcpagent` automatically plumbs and unplumbs logical interfaces as necessary for the IPv6 addresses supplied by the server. The IPv6 prefix length (netmask) is not set by the DHCPv6 protocol, but is instead set by `in.ndpd(1M)` using prefix information obtained by Router Advertisements. If any of the logical interfaces created by `dhcpagent` is unplumbed, marked down, or configured with a different IP address, it will be abandoned by DHCP control. If the link-local interface is unplumbed, then all addresses configured by DHCP on that physical interface will be removed.

In addition to DHCP, `dhcpagent` also supports BOOTP (IPv4 only). See *RFC 951, Bootstrap Protocol*. Configuration parameters obtained from a BOOTP server are treated identically to those received from a DHCP server, except that the IP address received from a BOOTP server always has an infinite lease.

DHCP also acts as a mechanism to configure other information needed by the client, for example, the domain name and addresses of routers. Aside from the IP address, and for IPv4 alone, the netmask, broadcast address, and default router, the agent does not directly configure the workstation, but instead acts as a database which may be interrogated by other programs, and in particular by `dhcpinfo(1)`.

On clients with a single interface, this is quite straightforward. Clients with multiple interfaces may present difficulties, as it is possible that some information arriving on different interfaces may need to be merged, or may be inconsistent. Furthermore, the configuration of the interfaces is asynchronous, so requests may arrive while some or all of the interfaces are still unconfigured. To handle these cases, one interface may be designated as primary, which makes it the authoritative source for the values of DHCP parameters in the case where no specific interface is requested. See `dhcpinfo(1)` and `ifconfig(1M)` for details.

For IPv4, the `dhcpagent` daemon can be configured to request a particular host name. See the `REQUEST_HOSTNAME` description in the `FILES` section. When first configuring a client to request a host name, you must perform the following steps as root to ensure that the full DHCP negotiation takes place:

```
# pkill dhcpagent
# rm /etc/dhcp/interface.dhc
# reboot
```

All DHCP packets sent by `dhcpagent` include a vendor class identifier (RFC 2132, option code 60; RFC 3315, option code 16). This identifier is the same as the platform name returned by the `uname -i` command, except:

- Any commas in the platform name are changed to periods.
- If the name does not start with a stock symbol and a comma, it is automatically prefixed with `SUNW`.

Messages

The `dhcpagent` daemon writes information and error messages in five categories:

critical
: Critical messages indicate severe conditions that prevent proper operation.

errors
: Error messages are important, sometimes unrecoverable events due to resource exhaustion and other unexpected failure of system calls; ignoring errors may lead to degraded functionality.

warnings
: Warnings indicate less severe problems, and in most cases, describe unusual or incorrect datagrams received from servers, or requests for service that cannot be provided.

informational
: Informational messages provide key pieces of information that can be useful to debugging a DHCP configuration at a site. Informational messages are generally controlled by the `-v` option. However, certain critical pieces of information, such as the IP address obtained, are always provided.

debug
: Debugging messages, which may be generated at two different levels of verbosity, are chiefly of benefit to persons having access to source code, but may be useful as well in debugging difficult DHCP configuration problems. Debugging messages are only generated when using the `-d` option.

When `dhcpagent` is run without the `-f` option, all messages are sent to the system logger `syslog(3C)` at the appropriate matching priority and with a facility identifier `LOG_DAEMON`. When `dhcpagent` is run with the `-f` option, all messages are directed to standard error.

DHCP Events and User-Defined Actions

If an executable (binary or script) is placed at `/etc/dhcp/eventhook`, the `dhcpagent` deamon will automatically run that program when any of the following events occur:

`BOUND` and `BOUND6`
: These events occur during interface configuration. The event program is invoked when `dhcpagent` receives the DHCPv4 ACK or DHCPv6 Reply message from the DHCP server for the lease request of an address, indicating successful initial configuration of the interface. (See also the `INFORM` and `INFORM6` events, which occur when configuration parameters are obtained without address leases.)

`EXTEND` and `EXTEND6`
: These events occur during lease extension. The event program is invoked just after `dhcpagent` receives the DHCPv4 ACK or DHCPv6 Reply from the DHCP server for the DHCPv4 REQUEST (renew) message or the DHCPv6 Renew or Rebind message.

  Note that with DHCPv6, the server might choose to remove some addresses, add new address leases, and ignore (allow to expire) still other addresses in a given Reply message. The `EXTEND6` event occurs when a Reply is received that leaves one or more address leases still valid, even if the Reply message does not extend the lease for any address. The event program is invoked just before any addresses are removed, but just after any new addresses are added. Those to be removed will be marked with the `IFF_DEPRECATED` flag.

`EXPIRE` and `EXPIRE6`
: These events occur during lease expiration. For DHCPv4, the event program is invoked just before the leased address is removed from an interface and the interface is marked as `down`. For DHCPv6, the event program is invoked just before the last remaining leased addresses are removed from the interface.

`DROP` and `DROP6`
: These events occur during the period when an interface is dropped. The event program is invoked just before the interface is removed from DHCP control. If the interface has been

abandoned due the user unplumbing the interface, then this event will occur after the user's action has taken place. The interface might not be present.

`INFORM` and `INFORM6`
: These events occur when an interface acquires new or updated configuration information from a DHCP server by means of the DHCPv4 `INFORM` or the DHCPv6 Information-Request message. These messages are sent using an `ifconfig(1M)` `dhcp inform` command or when the DHCPv6 Router Advertisement `O` (letter 0) bit is set and the `M` bit is not set. Thus, these events occur when the DHCP client does not obtain an IP address lease from the server, and instead obtains only configuration parameters.

`LOSS6`
: This event occurs during lease expiration when one or more valid leases still remain. The event program is invoked just before expired addresses are removed. Those being removed will be marked with the `IFF_DEPRECATED` flag.

  Note that this event is not associated with the receipt of the Reply message, which occurs only when one or more valid leases remain, and occurs only with DHCPv6. If all leases have expired, then the EXPIRE6 event occurs instead.

`RELEASE` and `RELEASE6`
: This event occurs during the period when a leased address is released. The event program is invoked just before `dhcpagent` relinquishes the address on an interface and sends the DHCPv4 `RELEASE` or DHCPv6 Release packet to the DHCP server.

The system does not provide a default event program. The file `/etc/dhcp/eventhook` is expected to be owned by root and have a mode of 755.

The event program will be passed two arguments, the interface name and the event name, respectively. For DHCPv6, the interface name is the name of the physical interface.

The event program can use the `dhcpinfo(1)` utility to fetch additional information about the interface. While the event program is invoked on every event defined above, it can ignore those events in which it is not interested. The event program runs with the same privileges and environment as `dhcpagent` itself, except that `stdin`, `stdout`, and `stderr` are redirected to `/dev/null`. Note that this means that the event program runs with root privileges.

If an invocation of the event program does not exit after 55 seconds, it is sent a `SIGTERM` signal. If does not exit within the next three seconds, it is terminated by a `SIGKILL` signal.

See EXAMPLES for an example event program.

Options The following options are supported:

`-a`
: Adopt a configured IPv4 interface. This option is for use with diskless DHCP clients. In the case of diskless DHCP, DHCP has already been performed on the network interface

providing the operating system image prior to running `dhcpagent`. This option instructs the agent to take over control of the interface. It is intended primarily for use in boot scripts.

The effect of this option depends on whether the interface is being adopted.

If the interface is being adopted, the following conditions apply:

`dhcpagent` uses the client id specified in `/chosen`:<*client_id*>, as published by the PROM or as specified on a `boot(1M)` command line. If this value is not present, the client id is undefined. The DHCP server then determines what to use as a client id. It is an error condition if the interface is an Infiniband interface and the PROM value is not present.

If the interface is not being adopted:

`dhcpagent` uses the value stored in `/etc/default/dhcpagent`. If this value is not present, the client id is undefined. If the interface is Infiniband and there is no value in `/etc/default/dhcpagent`, a client id is generated as described by the draft document on DHCP over Infiniband, available at:

`http://www.ietf.org`

`-d` *n*
Set debug level to *n*. Two levels of debugging are currently available, 1 and 2; the latter is more verbose.

`-f`
Run in the foreground instead of as a daemon process. When this option is used, messages are sent to standard error instead of to `syslog(3C)`.

`-v`
Provide verbose output useful for debugging site configuration problems.

Examples

EXAMPLE 1 Example Event Program

The following script is stored in the file `/etc/dhcp/eventhook`, owned by root with a mode of 755. It is invoked upon the occurrence of the events listed in the file.

```
#!/bin/sh

(
echo "Interface name: " $1
echo "Event: " $2

case $2 in
"BOUND")
     echo "Address acquired from server "\
         `/sbin/dhcpinfo -i $1 ServerID`
     ;;
"BOUND6")
```

EXAMPLE 1 Example Event Program *(Continued)*

```
     echo "Addresses acquired from server " \
         `/sbin/dhcpinfo -v6 -i $1 ServerID`
     ;;
"EXTEND")
    echo "Lease extended for " \
         `sbin/dhcpinfo -i $1 LeaseTim`" seconds"
     ;;
"EXTEND6")
    echo "New lease information obtained on $i"
     ;;
"EXPIRE" | "DROP" | "RELEASE")
     ;;

esac
) >/var/run/dhcp_eventhook_output 2>&1
```

Note the redirection of stdout and stderr to a file.

Files `/etc/dhcp/`*if*`.dhc`
`/etc/dhcp/`*if*`.dh6`
Contains the configuration for interface. The mere existence of this file does not imply that the configuration is correct, since the lease might have expired. On start-up, `dhcpagent` confirms the validity of the address using REQUEST (for DHCPv4) or Confirm (DHCPv6).

`/etc/dhcp/duid`
`/etc/dhcp/iaid`
Contains persistent storage for DUID (DHCP Unique Identifier) and IAID (Identity Association Identifier) values. The format of these files is undocumented, and applications should not read from or write to them.

`/etc/default/dhcpagent`
Contains default values for tunable parameters. All values may be qualified with the interface they apply to by prepending the interface name and a period (“.”) to the interface parameter name. The parameters include: the interface parameter name.

To configure IPv6 parameters, place the string `.v6` between the interface name (if any) and the parameter name. For example, to set the global IPv6 parameter request list, use `.v6.PARAM_REQUEST_LIST`. To set the `CLIENT_ID` (`DUID`) on `hme0`, use `hme0.v6.CLIENT_ID`.

The parameters include:

`RELEASE_ON_SIGTERM`
Indicates that a `RELEASE` rather than a `DROP` should be performed on managed interfaces when the agent terminates. Release causes the client to discard the lease, and the server to make the address available again. Drop causes the client to record the lease in `/etc/dhcp/`*interface*`.dhc` or `/etc/dhcp/`*interface*`.dh6` for later use.

`OFFER_WAIT`
Indicates how long to wait between checking for valid `OFFER`s after sending a `DISCOVER`. For DHCPv6, sets the time to wait between checking for valid Advertisements after sending a Solicit.

`CLIENT_ID`
Indicates the value that should be used to uniquely identify the client to the server. This value can take one of three basic forms:

decimal`,`*data*`...`
`0xHHHHH...`
"*string*`...`"

The first form is an RFC 3315 DUID. This is legal for both IPv4 DHCP and DHCPv6. For IPv4, an RFC 4361 Client ID is constructed from this value. In this first form, the format of *data*... depends on the decimal value. The following formats are defined for this first form:

1,*hwtype*,*time*,*lla*
Type 1, DUID-LLT. The *hwtype* value is an integer in the range 0-65535, and indicates the type of hardware. The *time* value is the number of seconds since midnight, January 1st, 2000 UTC, and can be omitted to use the current system time. The *lla* value is either a colon-separated MAC address or the name of a physical interface. If the name of an interface is used, the *hwtype* value can be omitted. For example: `1,,hme0`

2,*enterprise*,*hex*...
Type 2, DUID-EN. The *enterprise* value is an integer in the range 0-4294967295 and represents the SMI Enterprise number for an organization. The *hex* string is an even-length sequence of hexadecimal digits.

3,*hwtype*,*lla*
Type 3, DUID-LL. This is the same as DUID-LLT (type 1), except that a time stamp is not used.

*,*hex*
Any other type value (0 or 4-65535) can be used with an even-length hexadecimal string.

The second and third forms of `CLIENT_ID` are legal for IPv4 only. These both represent raw Client ID (without RFC 4361), in hex, or NVT ASCII string format. Thus, `Sun` and `0x53756E` are equivalent.

`PARAM_REQUEST_LIST`
Specifies a list of comma-separated integer values of options for which the client would like values.

`REQUEST_HOSTNAME`
Indicates the client requests the DHCP server to map the client's leased IPv4 address to the host name associated with the network interface that performs DHCP on the client. The host name must be specified in the `/etc/hostname.`*interface* file for the relevant interface on a line of the form

`inet` *hostname*

where *hostname* is the host name requested.

This option works with DHCPv4 only.

`/etc/dhcp/eventhook`
Location of a DHCP event program.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr
Interface Stability	Committed

See Also

`dhcpinfo(1)`, `ifconfig(1M)`, `init(1M)`, `in.ndpd(1M)`, `syslog(3C)`, `attributes(5)`, `dhcp(5)`

System Administration Guide: IP Services

Croft, B. and Gilmore, J.,*Bootstrap Protocol (BOOTP)*RFC 951, Network Working Group, September 1985.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 2131, Network Working Group, March 1997.

Lemon, T. and B. Sommerfeld. *RFC 4361, Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4)*. Nominum and Sun Microsystems. February 2006.

Droms, R. *RFC 3315, Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. Cisco Systems. July 2003.

Notes

The `dhcpagent` daemon can be used on IPv4 logical interfaces, just as with physical interfaces. When used on a logical interface, the daemon automatically constructs a Client ID value based on the DUID and IAID values, according to RFC 4361. The `/etc/default/dhcpclient` `CLIENT_ID` value, if any, overrides this automatic identifier.

As with physical IPv4 interfaces, the `/etc/hostname.hme0:1` and `/etc/dhcp.hme0:1` files must also be created in order for `hme0:1` to be automatically plumbed and configured at boot. In addition, unlike physical IPv4 interfaces, `dhcpagent` does not add or remove default routes associated with logical interfaces.

With DHCPv6, the link-local interface must be configured using `/etc/hostname6.hme0` in order for DHCPv6 to run on `hme0` at boot time. The logical interfaces for each address are plumbed by `dhcpagent` automatically.

Name dhcpconfig – DHCP service configuration utility

Synopsis

```
dhcpconfig -D -r resource -p path [-u uninterpreted]
     [-l lease_length] [-n ] [-d DNS_domain]
     [-a DNS_server_addresses] [-h hosts_resource]
     [-y hosts_domain]

dhcpconfig -R server_addresses

dhcpconfig -U [-f] [-x] [-h]

dhcpconfig -N network_address [-m subnet_mask] [-b ]
     [-t router_addresses] [-y NIS-domain]
     [-a NIS_server_addresses] [-g]

dhcpconfig -C -r resource -p path [-f] [-k]
     [-u uninterpreted]

dhcpconfig -X filename [-m macro_list] [-o option_list]
     [-a network_addresses] [-f] [-x] [-g]

dhcpconfig -I filename [-f] [-g]

dhcpconfig -P [parameter[=value],]...

dhcpconfig -S [-f] [-e | -d | -r | -q]
```

Description The `dhcpconfig` command is used to configure and manage the Dynamic Host Configuration Protocol (DHCP) service or BOOTP relay services. It is intended for use by experienced Solaris system administrators and is designed for ease of use in scripts. The `dhcpmgr` utility is recommended for less experienced administrators or those preferring a graphical utility to configure and manage the DHCP service or BOOTP relay service.

The `dhcpconfig` command can be run by root, or by other users assigned to the DHCP Management profile. See `rbac(5)` and `user_attr(4)`.

`dhcpconfig` requires one of the following function flags: `-D`, `-R`, `-U`, `-N`, `-C`, `-X`, `-I`, `-P` or `-S`.

The `dhcpconfig` menu driven mode is supported in Solaris 8 and previous versions of Solaris.

Where dhcpconfig Obtains Configuration Information

`dhcpconfig` scans various configuration files on your Solaris machine for information it can use to assign values to options contained in macros it adds to the `dhcptab` configuration table. The following table lists information `dhcpconfig` needs, the source used, and how the information is used:

Information	*Source*	*Where Used*
Timezone	System date, timezone settings	`Locale` macro
DNS parameters	`nsswitch.conf`, `/etc/resolv.conf`	Server macro

NIS parameters	System domainname, `nsswitch.conf`, NIS	Network macros
Subnetmask	Network interface, `netmasks` table in nameservice	Network macros

If you have not set these parameters on your server machine, you should do so before configuring the DHCP server with `dhcpconfig`. Note that if you specify options with the `dhcpconfig -D` command line, the values you supply override the values obtained from the system files.

Options The following options are supported:

`-C` Convert to using a new data store, recreating the DHCP data tables in a format appropriate to the new data store, and setting up the DHCP server to use the new data store.

The following sub-options are required:

`-p` *path_to_data* The paths for `SUNWfiles` and `SUNWbinfiles` must be absolute UNIX pathnames. The path for `SUNWnisplus` must be a fully specified NIS+ directory (including the tailing period.) See `dhcp_modules(5)`.

`-r` *data_resource* New data store resource. One of the following must be specified: `SUNWfiles`, `SUNWbinfiles`, or `SUNWnisplus`. See `dhcp_modules(5)`.

The following sub-options are optional:

`-f` Do not prompt for confirmation. If `-f` is not used, a warning and confirmation prompt are issued before the conversion starts.

`-k` Keep the old DHCP data tables after successful conversion. If any problem occurs during conversion, tables are not deleted even if `-k` sub-option is not specified.

`-u` *uninterpreted* Data which is ignored by `dhcpconfig`, but passed on to the datastore for interpretation. The private layer provides for module-specific configuration information through the use of the `RESOURCE_CONFIG` keyword. Uninterpreted data is stored within `RESOURCE_CONFIG` keyword of `dhcpsvc.conf(4)`. The `-u` sub-option is not used with the `SUNWfiles`, `SUNWbinfiles`, and

SUNWnisplus data stores. See dhcp_modules(5).

-D Configure the DHCP service.

The following sub-options are required:

-r *data_resource* One of the following must be specified: SUNWfiles, SUNWbinfiles, or SUNWnisplus. Other data stores may be available.See dhcp_modules(5).

-p *path* The paths for SUNWfiles and SUNWbinfiles must be absolute UNIX pathnames. The path for SUNWnisplus must be a fully specified NIS+ directory (including the tailing period.) . See dhcp_modules(5).

The following sub-options are optional:

-a *DNS_servers* IP addresses of DNS servers, separated with commas.

-d *DNS_domain* DNS domain name.

-h *hosts_resource* Resource in which to place hosts data. Usually, the name service in use on the server. Valid values are nisplus, files, or dns.

-l *seconds* Lease length used for addresses not having a specified lease length, in seconds.

-n Non-negotiable leases

-y *hosts_domain* DNS or NIS+ domain name to be used for hosts data. Valid only if dns or nisplus is specified for -h sub-option.

-u *uninterpreted* Data which is ignored by dhcpconfig, but passed on to the datastore for interpretation. The private layer provides for module-specific configuration information through the use of the RESOURCE_CONFIG keyword. Uninterpreted data is stored within RESOURCE_CONFIG keyword of dhcpsvc.conf(4). The -u sub-option is not used with the SUNWfiles, SUNWbinfiles, and SUNWnisplus data stores. See dhcp_modules(5).

-I *filename* Import data from *filename*, containing data previously exported from a Solaris DHCP server. Note that after importing, you may have to edit

macros to specify the correct domain names, and edit network tables to change the owning server of addresses in imported networks. Use `dhtadm` and `pntadm` to do this.

The following sub-options are supported:

`-f` Replace any conflicting data with the data being imported.

`-g` Signal the daemon to reload the `dhcptab` once the import has been completed.

`-N` *net_address* Configure an additional network for DHCP service.

The following sub-options are supported:

`-a` *NIS_server_addresses*	List of IP addresses of NIS servers.
`-b`	Network is a point-to-point (PPP) network, therefore no broadcast address should be configured. If `-b` is not used, the network is assumed to be a LAN, and the broadcast address is determined using the network address and subnet mask.
`-g`	Signal the daemon to reload the `dhcptab`.
`-m` *xxx.xxx.xxx.xxx*	Subnet mask for the network; if `-m` is not used, subnet mask is obtained from netmasks.
`-t` *router_addresses*	List of router IP addresses; if not specified, router discovery flag is set.
`-y` *NIS_domain_name*	If NIS is used on this network, specify the NIS domain name.

`-P` Configure the DHCP service parameters. Each parameter and value are specified by the following pattern:

parameter`[=`*value*`],...`

Where parameter and value are:

parameter One of the DHCP service parameters listed in `dhcpsvc.conf(4)`. If the corresponding *value* is not specified, the current parameter value is displayed. If *parameter* is not specified, all parameters and current values are displayed.

value Optional string to set the servers parameter to if the value is acceptable. If the value is missing or is empty (""), the parameter and its current value are deleted.

After a parameter has changed the DHCP server requires re-starting before you can use new parameter values.

`-R` *server_addresses* Configure the BOOTP relay service. BOOTP or DHCP requests are forwarded to the list of servers specified.

server_addresses is a comma separated list of hostnames and/or IP addresses.

`-S` Control the DHCP service.

The following sub-options are supported:

`-d` Disable and stop the DHCP service.

`-e` Enable and start the DHCP service.

`-q` Display the state of the DHCP service. The state is encoded into the exit status.

```
0    DHCP service disabled and stopped
1    DHCP service enabled and stopped
2    DHCP service disabled and running
3    DHCP service enabled and running
```

`-r` Enable and restart the DHCP service.

`-U` Unconfigure the DHCP service or BOOTP relay service.

The following sub-options are supported:

`-f` Do not prompt for confirmation. If `-f` is not used, a warning and confirmation prompt is issued.

`-h` Delete hosts entries from name service.

`-x` Delete the `dhcptab` and network tables.

`-X` *filename* Export data from the DHCP data tables, saving to *filename*, to move the data to another Solaris DHCP server.

The following sub-options are optional:

`-a` *networks_to_export* List of networks whose addresses should be exported, or the keyword ALL to specify all networks. If `-a` is not specified, no networks are exported.

`-g`	Signal the daemon to reload the `dhcptab` after the export has been completed.
`-m` *macros_to_export*	List of macros to export, or the keyword `ALL` to specify all macros. If `-m` is not specified, no macros are exported.
`-o` *options_to_export*	List of options to export, or the keyword `ALL` to specify all options. If `-o` is not specified, no options are exported.
`-x`	Delete the data from this server after it is exported. If `-x` is not specified you are in effect copying the data.

Examples

EXAMPLE 1 Configuring DHCP Service with Binary Files Data Store

The following command configures DHCP service, using the binary files data store, in the DNS domain `acme.eng`, with a lease time of 28800 seconds (8 hours),

```
example# dhcpconfig -D -r SUNWbinfiles -p /var/dhcp -l 28800\
            -d acme.eng -a 120.30.33.4 -h dns -y acme.eng
```

EXAMPLE 2 Configuring BOOTP Relay Agent

The following command configures the DHCP daemon as a BOOTP relay agent, which forwards BOOTP and DHCP requests to the servers having the IP addresses 120.30.33.7 and 120.30.42.132:

```
example# dhcpconfig -R 120.30.33.7,120.30.42.132
```

EXAMPLE 3 Unconfiguring DHCP Service

The following command unconfigures the DHCP service, with confirmation, and deletes the DHCP data tables and host table entries:

```
example# dhcpconfig -U -x -h
```

EXAMPLE 4 Configuring a Network for DHCP Service

The following command configures an additional LAN network for DHCP service, specifying that clients should use router discovery and providing the NIS domain name and NIS server address:

```
example# dhcpconfig -N 120.30.171.0 -y east.acme.eng.com\
            -a 120.30.33.4
```

EXAMPLE 5 Converting to SUNWnisplus Data Store

The following command converts a DHCP server from using a text or binary files data store to a NIS+ data store, deleting the old data store's DHCP tables:

EXAMPLE 5 Converting to SUNWnisplus Data Store *(Continued)*

```
example# dhcpconfig -C -r SUNWnisplus -p whatever.com.
```

EXAMPLE 6 Exporting a Network, Macros, and Options from a DHCP Server

The following command exports one network (`120.30.171.0`) and its addresses, the macro `120.30.171.0`, and the options `motd` and `PSptr`from a DHCP server, saves the exported data in file `/export/var/120301710_data`, and deletes the exported data from the server.

```
example# dhcpconfig -X /var/dhcp/120301710_export
     -a 120.30.171.0 -m 120.30.171.0 -o motd,PSptr
```

EXAMPLE 7 Importing Data on a DHCP Server

The following command imports DHCP data from a file, `/net/golduck/export/var/120301710_data`, containing data previously exported from a Solaris DHCP server, overwrites any conflicting data on the importing server, and signals the daemon to reload the `dhcptab` once the import is completed:

```
example# dhcpconfig -I /net/golduck/export/var/120301710_data -f -g
```

EXAMPLE 8 Setting DHCP Server Parameters

The following command sets the number of minutes that the DHCP server waits before timing out when updating DNS information on DHCP clients to five minutes.

```
example# example# dhcpconfig -P UPDATE_TIMEOUT=5
```

EXAMPLE 9 Re-starting the DHCP server

The following command stops and re-starts the DHCP server.

```
example#  example# dhcpconfig -S -r
DHCP server stopped
DHCP server started
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdhcsu
Interface Stability	Evolving

See Also

`dhcpmgr(1M)`, `dhtadm(1M)`, `in.dhcpd(1M)`, `pntadm(1M)`, `dhcp_network(4)`, `dhcptab(4)`, `dhcpsvc.conf(4)`, `nsswitch.conf(4)`, `resolv.conf(4)`, `user_attr(4)`, `attributes(5)`, `dhcp(5)`, `dhcp_modules(5)`, `rbac(5)`

System Administration Guide: IP Services

Name dhcpmgr – graphical interface for managing DHCP service

Synopsis

```
/usr/sadm/admin/bin/dhcpmgr
```

Description `dhcpmgr` is a graphical user interface which enables you to manage the Dynamic Host Configuration Protocol (DHCP) service on the local system. It performs the functions of the `dhcpconfig`, `dhtadm`, and `pntadm` command line utilities. You must be `root` to use `dhcpmgr`. The `dhcpmgr` Help, available from the `Help` menu, contains detailed information about using the tool.

Usage You can perform the following tasks using `dhcpmgr`:

Configure DHCP service	Use `dhcpmgr` to configure the DHCP daemon as a DHCP server, and select the data store to use for storing network configuration tables..
Configure BOOTP relay service	Use `dhcpmgr` to configure the DHCP daemon as a BOOTP relay.
Manage DHCP or BOOTP relay service	Use `dhcpmgr` to start, stop, enable, disable or unconfigure the DHCP service or BOOTP relay service, or change DHCP server parameters.
Manage DHCP addresses	Use `dhcpmgr` to add, modify, or delete IP addresses leased by the DHCP service.
Manage DHCP macros	Use `dhcpmgr` to add, modify or delete macros used to supply configuration parameters to DHCP clients.
Manage DHCP options	Use `dhcpmgr` to add, modify or delete options used to define parameters deliverable through DHCP.
Convert to a new DHCP data store	Use `dhcpmgr` to configure the DHCP server to use a different data store, and convert the DHCP data to the format used by the new data store.
Move DHCP data to another server	Use `dhcpmgr` to export data from one Solaris DHCP server and import data onto another Solaris DHCP server.

Exit Status The following exit values are returned:

`0`	Successful completion.
`non-zero`	An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdhcm
Interface Stability	Evolving

See Also dhcpconfig(1M), dhtadm(1M), pntadm(1M), in.dhcpd(1M), dhcpsvc.conf(4), dhcp_network(4), dhcptab(4), attributes(5), dhcp(5), dhcp_modules(5)

Solaris DHCP Service Developer's Guide

System Administration Guide: IP Services

Name dhtadm – DHCP configuration table management utility

Synopsis

```
dhtadm -C [-r resource] [-p path] [-u uninterpreted] [-g]

dhtadm -A -s symbol_name -d definition [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -A -m macro_name -d definition [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -M -s symbol_name -d definition [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -M -s symbol_name -n new_name [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -M -m macro_name -n new_name [-r resource] [-p path]
     [-u uninterpreted] [-g]

dhtadm -M -m macro_name -d definition [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -M -m macro_name -e symbol=value [-r resource]
     [-p path] [-u uninterpreted] [-g]

dhtadm -D -s symbol_name [-r resource] [-p path]
     [-u uninterpreted] [-g]

dhtadm -D -m macro_name [-r resource] [-p path]
     [-u uninterpreted] [-g]

dhtadm -P [-r resource] [-p path] [-u uninterpreted] [-g]

dhtadm -R [-r resource] [-p path] [-u uninterpreted] [-g]

dhtadm -B [-v] [batchfile] [-g]
```

Description `dhtadm` manages the Dynamic Host Configuration Protocol (DHCP) service configuration table, `dhcptab`. You can use it to add, delete, or modify DHCP configuration macros or options or view the table. For a description of the table format, see `dhcptab(4)`.)

The `dhtadm` command can be run by root, or by other users assigned to the DHCP Management profile. See `rbac(5)` and `user_attr(4)`.

After you make changes with `dhtadm`, you should issue a `SIGHUP` to the DHCP server, causing it to read the `dhcptab` and pick up the changes. Do this using the `-g` option.

Options One of the following function flags must be specified with the `dhtadm` command: `-A`, `-B`, `-C`, `-D`, `-M`, `-P` or `-R`.

The following options are supported:

`-A` Add a symbol or macro definition to the `dhcptab` table.

The following sub-options are required:

`-d` *definition* — Specify a macro or symbol definition.

definition must be enclosed in single quotation marks. For macros, use the form `-d ':`*symbol*`=`*value*`:`*symbol*`=`*value*`:'`. Enclose a *value* that contains colons in double quotation marks. For symbols, the definition is a series of fields that define a symbol's characteristics. The fields are separated by commas. Use the form `-d '`*context*`,`*code*`,`*type*`,`*granularity*`,`*maximum*`'`. See `dhcptab(4)` for information about these fields.

`-m` *macro_name* — Specify the name of the macro to be added.

The `-d` option must be used with the `-m` option. The `-s` option cannot be used with the `-m` option.

`-s` *symbol_name* — Specify the name of the symbol to be added.

The `-d` option must be used with the `-s` option. The `-m` option cannot be used with the `-s` option.

`-B` — Batch process `dhtadm` commands. `dhtadm` reads from the specified file or from standard input a series of `dhtadm` commands and execute them within the same process. Processing many `dhtadm` commands using this method is much faster than running an executable batchfile itself. Batch mode is recommended for using `dhtadm` in scripts.

The following sub-option is optional:

`-v` — Display commands to standard output as they are processed.

`-C` — Create the DHCP service configuration table, `dhcptab`.

`-D` — Delete a symbol or macro definition.

The following sub-options are required:

`-m` *macro_name* — Delete the specified macro.

`-s` *symbol_name* — Delete the specified symbol.

`-g` Signal the DHCP daemon to reload the `dhcptab` after successful completion of the operation.

`-M` Modify an existing symbol or macro definition.

The following sub-options are required:

`-d` *definition* Specify a macro or symbol definition to modify.

The definition must be enclosed in single quotation marks. For macros, use the form `-d ':`*symbol*`=`*value*`:`*symbol*`=`*value*`:'`. Enclose a *value* that contains colons in double quotation marks. For symbols, the definition is a series of fields that define a symbol's characteristics. The fields are separated by commas. Use the form `-d '`*context*`,`*code*`,`*type*`,`*granularity*`,`*maximum*`'`. See `dhcptab(4)` for information about these fields.

`-e` This sub-option uses the *symbol*`=`*value* argument. Use it to edit a *symbol*/*value* pair within a macro. To add a symbol which does not have an associate value, enter:

symbol`=_NULL_VALUE_`

To delete a symbol definition from a macro, enter:

symbol`=`

`-m` This sub-option uses the *macro_name* argument. The `-n`, `-d`, or `-e` sub-options are legal companions for this sub-option..

`-n` This sub-option uses the *new_name* argument and modifies the name of the object specified by the `-m` or `-s` sub-option. It is not limited to macros. . Use it to specify a new macro name or symbol name.

`-s` This sub-option uses the *symbol_name* argument. Use it to specify a symbol. The `-d` sub-option is a legal companion.

`-p` *path*	Override the `dhcpsvc.conf(4)` configuration value for `PATH=` with *path*. See `dhcpsvc.conf(4)` for more details regarding *path*. See `dhcp_modules(5)` for information regarding data storage modules for the DHCP service.
`-P`	Print (display) the `dhcptab` table.
`-r` *data_store_resource*	Override the `dhcpsvc.conf`(4) configuration value for `RESOURCE=` with the *data_store_resource* specified. See `dhcpsvc.conf(4)` for more details on resource type. See*Solaris DHCP Service Developer's Guide* for more information about adding support for other data stores. See `dhcp_modules(5)` for information regarding data storage modules for the DHCP service.
`-R`	Remove the `dhcptab` table.
`-u` *uninterpreted*	Data which is ignored by `dhtadm`, but passed to currently configured public module, to be interpreted by the data store. The private layer provides for module-specific configuration information through the use of the `RESOURCE_CONFIG` keyword. Uninterpreted data is stored within `RESOURCE_CONFIG` keyword of `dhcpsvc.conf(4)`. See `dhcp_modules(5)` for information regarding data storage modules for the DHCP service.

Examples

EXAMPLE 1 Creating the DHCP Service Configuration Table

The following command creates the DHCP service configuration table, `dhcptab`:

```
# dhtadm -C
```

EXAMPLE 2 Adding a Symbol Definition

The following command adds a `Vendor` option symbol definition for a new symbol called `MySym` to the `dhcptab` table in the `SUNWfiles` resource in the `/var/mydhcp` directory:

```
# dhtadm -A -s MySym
   -d 'Vendor=SUNW.PCW.LAN,20,IP,1,0'
   -r SUNWfiles -p /var/mydhcp
```

EXAMPLE 3 Adding a Macro Definition

The following command adds the `aruba` macro definition to the `dhcptab` table. Note that symbol/value pairs are bracketed with colons (`:`).

```
# dhtadm -A -m aruba \
     -d ':Timeserv=10.0.0.10 10.0.0.11:DNSserv=10.0.0.1:'
```

EXAMPLE 4 Modifying a Macro Definition

The following command modifies the `Locale` macro definition, setting the value of the `UTCOffst` symbol to 18000 seconds. Note that any macro definition which includes the definition of the `Locale` macro inherits this change.

```
# dhtadm -M -m Locale -e 'UTCOffst=18000'
```

EXAMPLE 5 Deleting a Symbol

The following command deletes the `Timeserv` symbol from the `aruba` macro. Any macro definition which includes the definition of the `aruba` macro inherits this change.

```
# dhtadm -M -m aruba -e 'Timeserv='
```

EXAMPLE 6 Adding a Symbol to a Macro

The following command adds the `Hostname` symbol to the `aruba` macro. Note that the `Hostname` symbol takes no value, and thus requires the special value `_NULL_VALUE_`. Note also that any macro definition which includes the definition of the `aruba` macro inherits this change.

```
# dhtadm -M -m aruba -e 'Hostname=_NULL_VALUE_'
```

EXAMPLE 7 Renaming a Macro

The following command renames the `Locale` macro to `MyLocale`. Note that any `Include` statements in macro definitions which include the `Locale` macro also need to be changed.

```
# dhtadm -M -m Locale -n MyLocale
```

EXAMPLE 8 Deleting a Symbol Definition

The following command deletes the `MySym` symbol definition. Note that any macro definitions which use `MySym` needs to be modified.

```
# dhtadm -D -s MySym
```

EXAMPLE 9 Removing a dhcptab

The following command removes the `dhcptab` table in the NIS+ directory specified.

```
# dhtadm -R -r SUNWnisplus -p Test.Nis.Plus.
```

EXAMPLE 10 Printing a dhcptab

The following command prints to standard output the contents of the `dhcptab` that is located in the data store and path indicated in the `dhcpsvc.conf` file:.

```
# dhtadm -P
```

EXAMPLE 11 Executing dhtadm in Batch Mode

The following command runs a series of `dhtadm` commands contained in a batch file and signals the daemon to reload the `dhcptab` once the commands have been executed: :

```
# dhtadm -B addmacros -g
```

Exit Status

`0` Successful completion.

`1` Object already exists.

`2` Object does not exist.

`3` Non-critical error.

`4` Critical error.

Files `/etc/inet/dhcpsvc.conf`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdhcsu
Interface Stability	Evolving

See Also `dhcpconfig(1M)`, `dhcpmgr(1M)`, `in.dhcpd(1M)`, `dhcpsvc.conf(4)`, `dhcp_network(4)`, `dhcptab(4)`, `hosts(4)`, `user_attr(4)`, `attributes(5)`, `dhcp(5)`, `dhcp_modules(5)rbac(5)`

Solaris DHCP Service Developer's Guide

System Administration Guide: IP Services

Alexander, S., and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, RFC 1533, Lachman Technology, Inc., Bucknell University, October 1993.

Droms, R., *Interoperation Between DHCP and BOOTP*, RFC 1534, Bucknell University, October 1993.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 1541, Bucknell University, October 1993.

Wimer, W., *Clarifications and Extensions for the Bootstrap Protocol*, RFC 1542, Carnegie Mellon University, October 1993.

Name dig – DNS lookup utility

Synopsis

```
dig [@server] [-b address] [-c class] [-f filename]
     [-k filename] [-m] [-p port#] [-q name] [-t type] [-x addr]
     [-y [hmac:]name:key] [-4] [-6] [name] [type] [class] [queryopt]...

dig [-h]

dig [global-queryopt...] [query...]
```

Description The `dig` utility (domain information groper) is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use `dig` to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output. Other lookup tools tend to have less functionality than `dig`.

Although `dig` is normally used with command-line arguments, it also has a batch mode of operation for reading lookup requests from a file. A brief summary of its command-line arguments and options is printed when the `-h` option is specified. Unlike earlier versions, the BIND 9 implementation of `dig` allows multiple lookups to be issued from the command line.

Unless it is told to query a specific name server, `dig` tries each of the servers listed in `/etc/resolv.conf`.

When no command line arguments or options are given, `dig` performs an NS query for "." (the root).

It is possible to set per-user defaults for `dig` with `${HOME}/.digrc`. This file is read and any options in it are applied before the command line arguments.

The `IN` and `CH` class names overlap with the `IN` and `CH` top level domains names. Either use the `-t` and `-c` options to specify the type and class, or use "`IN.`" and "`CH.`" when looking up these top level domains.

Simple Usage The following is a typical invocation of `dig`:

```
dig @server name type
```

where:

server
: The name or IP address of the name server to query. This can be an IPv4 address in dotted-decimal notation or an IPv6 address in colon-delimited notation. When the supplied *server* argument is a hostname, `dig` resolves that name before querying that name server. If no *server* argument is provided, `dig` consults `/etc/resolv.conf` and queries the name servers listed there. The reply from the name server that responds is displayed.

name
: The name of the resource record that is to be looked up.

type
Indicates what type of query is required (ANY, A, MX, SIG, among others.) *type* can be any valid query type. If no *type* argument is supplied, `dig` performs a lookup for an A record.

Options The following options are supported:

`-4`
Use only IPv4 transport. By default both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-6`
Use only IPv6 transport. By default both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-b` *address*
Set the source IP address of the query to *address*. This must be a valid address on one of the host's network interfaces or `0.0.0.0` or `::`. An optional port may be specified by appending: `#`<*port*>

`-c` *class*
Override the default query class (IN for internet). The *class* argument is any valid class, such as HS for Hesiod records or CH for CHAOSNET records.

`-f` *filename*
Operate in batch mode by reading a list of lookup requests to process from the file *filename*. The file contains a number of queries, one per line. Each entry in the file should be organized in the same way they would be presented as queries to `dig` using the command-line interface.

`-h`
Print a brief summary of command-line arguments and options.

`-k` *filename*
Specify a transaction signature (TSIG) key file to sign the DNS queries sent by `dig` and their responses using TSIGs.

`-m`
Enable memory usage debugging.

`-p` *port#*
Query a non-standard port number. The *port#* argument is the port number that `dig` sends its queries instead of the standard DNS port number 53. This option tests a name server that has been configured to listen for queries on a non-standard port number.

`-q` *name*
Sets the query name to *name*. This can be useful in that the query name can be easily distinguished from other arguments.

`-t` *type*
Set the query type to *type*, which can be any valid query type supported in BIND9. The default query type "A", unless the `-x` option is supplied to indicate a reverse lookup. A zone transfer can be requested by specifying a type of AXFR. When an incremental zone transfer (IXFR) is required, *type* is set to `ixfr=`*N*. The incremental zone transfer will contain the changes made to the zone since the serial number in the zone's SOA record was *N*.

`-x` *addr*
Simplify reverse lookups (mapping addresses to names). The *addr* argument is an IPv4 address in dotted-decimal notation, or a colon-delimited IPv6 address. When this option is used, there is no need to provide the *name*, *class* and *type* arguments. The `dig` utility automatically performs a lookup for a name like `11.12.13.10.in-addr.arpa` and sets the query type and class to PTR and IN, respectively. By default, IPv6 addresses are looked up using nibble format under the IP6.ARPA domain. To use the older RFC1886 method using the IP6.INT domain, specify the `-i` option. Bit string labels (RFC 2874) are now experimental and are not attempted.

`-y` [*hmac*:]*name*:*key*
Specify a transaction signature (TSIG) key on the command line. This is done to sign the DNS queries sent by `dig`, as well as their responses. You can also specify the TSIG key itself on the command line using the `-y` option. The optional *hmac* is the type of TSIG; the default is `HMAC-MD5`. The *name* argument is the name of the TSIG key and the *key* argument is the actual key. The key is a base-64 encoded string, typically generated by `dnssec-keygen(1M)`.

Caution should be taken when using the `-y` option on multi-user systems, since the key can be visible in the output from `ps(1)` or in the shell's history file. When using TSIG authentication with `dig`, the name server that is queried needs to know the key and algorithm that is being used. In BIND, this is done by providing appropriate `key` and `server` statements in `named.conf`.

Query Options The `dig` utility provides a number of query options which affect the way in which lookups are made and the results displayed. Some of these set or reset flag bits in the query header, some determine which sections of the answer get printed, and others determine the timeout and retry strategies.

Each query option is identified by a keyword preceded by a plus sign (+). Some keywords set or reset an option. These may be preceded by the string no to negate the meaning of that keyword. Other keywords assign values to options like the timeout interval. They have the form `+keyword=`*value*. The query options are:

`+[no]tcp`
Use [do not use] TCP when querying name servers. The default behaviour is to use UDP unless an AXFR or IXFR query is requested, in which case a TCP connection is used.

`+[no]vc`
Use [do not use] TCP when querying name servers. This alternate syntax to `+[no]tcp` is provided for backwards compatibility. The "vc" stands for "virtual circuit".

`+[no]ignore`
Ignore truncation in UDP responses instead of retrying with TCP. By default, TCP retries are performed.

`+domain=`*somename*
Set the search list to contain the single domain *somename*, as if specified in a `domain` directive in `/etc/resolv.conf`, and enable search list processing as if the `+search` option were given.

`+[no]search`
Use [do not use] the search list defined by the `searchlist` or `domain` directive in `resolv.conf` (if any). The search list is not used by default.

`+[no]showsearch`
Perform [do not perform] a search showing intermediate results.

`+[no]defname`
Deprecated, treated as a synonym for `+[no]search`.

`+[no]aaonly`
Sets the aa flag in the query.

`+[no]aaflag`
A synonym for `+[no]aaonly`.

`+[no]adflag`
Set [do not set] the AD (authentic data) bit in the query. This requests that the server return, regardless of whether all of the answer and authority sections have all been validated as secure according to the security policy of the server. A setting of `AD=1` indicates that all records have been validated as secure and the answer is not from an `OPT-OUT` range. `AD=0` indicates that some part of the answer is insecure or not validated.

`+[no]cdflag`
Set [do not set] the CD (checking disabled) bit in the query. This requests the server to not perform DNSSEC validation of responses.

`+[no]cl`
Display [do not display] the CLASS when printing the record.

`+[no]ttlid`
Display [do not display] the TTL when printing the record.

`+[no]recurse`
Toggle the setting of the RD (recursion desired) bit in the query. This bit is set by default, which means `dig` normally sends recursive queries. Recursion is automatically disabled when the `+nssearch` or `+trace` query options are used.

`+[no]nssearch`
When this option is set, `dig` attempts to find the authoritative name servers for the zone containing the name being looked up and display the SOA record that each name server has for the zone.

`+[no]trace`
Toggle tracing of the delegation path from the root name servers for the name being looked up. Tracing is disabled by default. When tracing is enabled, `dig` makes iterative queries to resolve the name being looked up. It will follow referrals from the root servers, showing the answer from each server that was used to resolve the lookup.

`+[no]cmd`
Toggle the printing of the initial comment in the output identifying the version of `dig` and the query options that have been applied. This comment is printed by default.

`+[no]short`
Provide a terse answer. The default is to print the answer in a verbose form.

`+[no]identify`
Show [or do not show] the IP address and port number that supplied the answer when the *+short* option is enabled. If short form answers are requested, the default is not to show the source address and port number of the server that provided the answer.

`+[no]comments`
Toggle the display of comment lines in the output. The default is to print comments.

`+[no]stats`
Toggle the printing of statistics: when the query was made, the size of the reply and so on. The default behaviour is to print the query statistics.

`+[no]qr`
Print [do not print] the query as it is sent. By default, the query is not printed.

`+[no]question`
Print [do not print] the question section of a query when an answer is returned. The default is to print the question section as a comment.

`+[no]answer`
Display [do not display] the answer section of a reply. The default is to display it.

`+[no]authority`
Display [do not display] the authority section of a reply. The default is to display it.

`+[no]additional`
Display [do not display] the additional section of a reply. The default is to display it.

`+[no]all`
Set or clear all display flags.

`+time=`*T*
Sets the timeout for a query to *T* seconds. The default time out is 5 seconds. An attempt to set *T* to less than 1 will result in a query timeout of 1 second being applied.

`+tries=`*T*
Sets the maximum number of UDP attempts to *T*. The default number is 3 (1 initial attempt followed by 2 retries). If T is less than or equal to zero, the number of retries is silently rounded up to 1.

`+retry=`*T*
Sets the number of UDP retries to *T*. The default is 2.

`+ndots=`*D*
Set the number of dots that have to appear in *name* to *D* for it to be considered absolute. The default value is that defined using the `ndots` statement in `/etc/resolv.conf`, or 1 if no `ndots` statement is present. Names with fewer dots are interpreted as relative names and will be searched for in the domains listed in the `search` or `domain` directive in `/etc/resolv.conf`.

`+bufsize=`*B*
Set the UDP message buffer size advertised using EDNS0 to *B* bytes. The maximum and minimum sizes of this buffer are 65535 and 0 respectively. Values outside this range are rounded up or down appropriately.

`+edns=#`
Specify the EDNS version with which to query. Valid values are 0 to 255. Setting the EDNS version causes a EDNS query to be sent. `+noedns` clears the remembered EDNS version.

`+[no]multiline`
Print records like the SOA records in a verbose multi-line format with human-readable comments. The default is to print each record on a single line, to facilitate machine parsing of the `dig` output.

`+[no]fail`
Do not try the next server if you receive a `SERVFAIL`. The default is to not try the next server which is the reverse of normal stub resolver behavior.

`+[no]besteffort`
Attempt to display the contents of messages which are malformed. The default is to not display malformed answers.

`+[no]dnssec`
Request DNSSEC records be sent by setting the DNSSEC OK bit (DO) in the OPT record in the additional section of the query.

`+[no]sigchase`
Chase DNSSEC signature chains. Requires `dig` be compiled with `-DDIG_SIGCHASE`.

`+trusted-key=####`
Specifies a file containing trusted keys to be used with `+sigchase`. Each `DNSKEY` record must be on its own line.

If not specified dig will look for `/etc/trusted-key.key` then `trusted-key.key` in the current directory.

Requires `dig` be compiled with `-DDIG_SIGCHASE`.

`+[no]topdown`
When chasing DNSSEC signature chains, perform a top-down validation. Requires `dig` be compiled with `-DDIG_SIGCHASE`.

`+[no]nsid`
Include an EDNS name server ID request when sending a query.

Multiple Queries The BIND 9 implementation of `dig` supports specifying multiple queries on the command line (in addition to supporting the `-f` batch file option). Each of those queries can be supplied with its own set of flags, options and query options.

In this case, each *query* argument represent an individual query in the command-line syntax described above. Each consists of any of the standard options and flags, the name to be looked up, an optional query type, and class and any query options that should be applied to that query.

A global set of query options, which should be applied to all queries, can also be supplied. These global query options must precede the first tuple of name, class, type, options, flags, and query options supplied on the command line. Any global query options (except the `+[no]cmd` option) can be overridden by a query-specific set of query options. For example:

```
dig +qr www.isc.org any -x 127.0.0.1 isc.org ns +noqr
```

...shows how `dig` could be used from the command line to make three lookups: an ANY query for `www.isc.org`, a reverse lookup of 127.0.0.1 and a query for the NS records of `isc.org`. A global query option of `+qr` is applied, so that `dig` shows the initial query it made for each lookup. The final query has a local query option of `+noqr` which means that `dig` will not print the initial query when it looks up the NS records for `isc.org`.

Files `/etc/resolv.conf`
Resolver configuration file

`${HOME}/.digrc`
User-defined configuration file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dns/bind

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Volatile

See Also dnssec-keygen(1M), host(1M), named(1M), nslookup(1M), attributes(5)

RFC 1035

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at https://www.isc.org/software/bind/documentation.

Bugs There are probably too many query options.

Notes nslookup(1M) and dig now report "Not Implemented" as NOTIMP rather than NOTIMPL. This will have impact on scripts that are looking for NOTIMPL.

Name directoryserver – front end for the Directory Server (DS)

Synopsis

```
/usr/sbin/directoryserver
     { setup [-f configuration_file] | uninstall}

/usr/sbin/directoryserver
     {start-admin | stop-admin | restart-admin | startconsole}

/usr/sbin/directoryserver [{-s | -server} server-instance ]
     {start | stop | restart}

/usr/sbin/directoryserver { -s |-server } server-instance
     { monitor | saveconfig | restoreconfig | db2index-task |
       ldif2db-task | ldif2db | ldif2ldap | vlvindex | db2ldif |
       db2ldif-task | db2bak | db2bak-task | bak2db | bak2db-task |
       suffix2instance | account-status | account-activate |
       account-inactivate }
     {...}

/usr/sbin/directoryserver nativetoascii | admin_ip | ldif |
     pwdhash | idsktune | mmldif | keyupg
     {...}

/usr/sbin/directoryserver { magt | sagt } {...}

/usr/sbin/directoryserver help [subcommand]
```

Description The `directoryserver` command is a comprehensive, front end to the utility programs provided by the Solaris Directory Server (DS).

Options for the `directoryserver` command itself must appear before the subcommand. Arguments for a subcommand must appear after the subcommand. Subcommands have specific arguments. See `SUBCOMMANDS`.

Subcommands The following subcommands are supported:

account-inactivate *args*

Inactivates and locks an entry or group of entries.

The `account-inactivate` subcommand supports the following arguments:

[`-D` *rootdn*] Directory Server *userDN* with root permissions, such as Directory Manager.

[`-h` *host*] Host name of Directory Server. The default value is the full hostname of the machine where Directory Server is installed.

`-I` *DN* Entry DN or role DN to activate.

`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
[`-p` *port*]	Directory Server port. The default value is the LDAP port of Directory Server specified at installation time.
`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.

account-activate *args*

Activates an entry or group of entries.

The `account-activate` subcommand supports the following arguments

`-D` *rootdn*	Directory Server userDN with root permissions, such as Directory Manager.
`-h` *host*	Host name of Directory Server. The default value is the full hostname of the machine where Directory Server is installed.
`-I` DN	Entry DN or role DN to activate.
`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.

	`-p` *port*	Directory Server port. The default value is the LDAP port of Directory Server specified at installation time.
	`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.
account-status *args*	Provides account status information to establish whether an entry or group of entries is inactivated or not. The `account-status` subcommand supports the following arguments:	
	`-D` *rootdn*	
	`-h` *host*	Host name of Directory Server. The default value is the full hostname of the machine where Directory Server is installed.
	`-I` *DN*	Entry DN or role DN whose status is required.
	`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
	`-p` *port*	Directory Server port. The default value is the LDAP port of Directory Server specified at installation time.
	`-w` *password*	Password associated with the rootDN. Supplying the password on the command line is visible using the

`/bin/ps` command. This is considered insecure. Use with extreme caution.

The value `-` can be used in place of the password. The program prompts the user for a password to be entered from the terminal.

admin_ip *args*

Change the IP address of the administrative server in the configuration.

The `admin_ip` subcommand supports the following arguments:

dir_mgr_DN	Directory Manager's DN.
dir_mgr_password	Directory Manager's password.
old_ip	Old IP.
new_ip	New IP.
port_#	Port number.

bak2db *backup_directory*

Restore the database from the most recent archived backup.

Specify *backup_directory* as the backup directory.

bak2db-task *args*

Restore the data to the database.

The bak2db-task subcommand supports the following arguments:

[`-a` *directory*]	Directory where the backup files are stored. By default it is under `/var/ds5/slapd-serverID/bak`
`-D` *rootDN*	User DN with root permissions, such as Directory Manager. The default is the DN of the directory manager which is read from the `nsslapd-root` attribute under `cn=config`.

	`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
	[`-t` *database_type*]	Database type. The only possible database type is `ldbm`.
	[`-v`]	Verbose mode.
	`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.
db2bak-task *args*	Back up the contents of the database. It creates an entry in the directory that launches this dynamic task. An entry is generated based upon the values provided for each option. The `db2bak-task` subcommand supports the following arguments:	
	[`-a` *directory*]	Directory where the backup files are stored. By default it is under `/var/ds5/slapd-serverID/bak`. The backup file is named according to the year-month-day-hour format (*YYYY_MM_DD_hhmmss*).
	`-D` *rootDN*	User DN with root permissions, such as Directory

		Manager. The default is the DN of the directory manager which is read from the `nsslapd-root` attribute under `cn=config`.
	`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
	`-t` *database_type*	Database type. The only possible database type is `ldbm`.
	[`-v`]	Verbose mode.
	`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.
db2bak [*backup_directory*]		Create a backup of the current database contents. The server must be stopped to run this subcommand. The default is `/var/ds5/slapd-serverID/bak`. The backup file is named according to the year-month-day-hour format (*YYYY_MM_DD_hhmmss*).
db2index-text *args*		Create and generate the new set of indexes to be maintained following the modification of indexing entries in the `cn=config` configuration file. The `db2index-text` subcommand supports the following arguments:

`-D` *rootdn*	User DN with root permissions, such as Directory Manager.
`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
`-n` *backend_instance*	Instance to be indexed.
[`-t` *attributeName*]	Name of the attribute to be indexed. If omitted, all indexes defined for that instance are generated.
[`-v`]	Verbose mode.
`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.

db2ldif-task *args*

Exports the contents of the database to LDIF. It creates an entry in the directory that launches this dynamic task. The entry is generated based upon the values you provide for each option. To run this subcommand the server must be running and either `-n` *backend_instance* or `-s` *include* suffix is required.

The `db2ldif-task` subcommand supports the following arguments:

[`-a` *outputfile*]	File name of the output LDIF file.

`-C`	Only the main db file is used.
`-D` *rootDN*	User DN with root permissions, such as Directory Manager.
`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure.Use with extreme caution.
[`-M`]	Output LDIF is stored in multiple files.
[`-m`]	Minimal base 64 encoding.
{`-n` *backend_instance*}*	Instance to be exported.
[`-N`]	Minimal base 64 encoding.
[`-o`]	Output LDIF to be stored in one file by default with each instance stored in *instance_file* name.
[`-r`]	Export replica.
[`-s`]*includesuffix*}*	Suffix(es) to be included or to specify the subtrees to be included if `-n` has been used.
[`-u`]	Request that the unique ID is not exported.
[`-U`]	Request that the output LDIF is not folded.
`-w` *password*	Password associated with the user DN. Supplying the password on the command line is visible

		using the `/bin/ps` command. This is considered insecure. Use with extreme caution. The value `-` can be used in place the password. The program prompts the user for a password to be entered from the terminal.
	{`-x` *excludesuffix*}*	Suffixes to be excluded.
	[`-1`]	Delete, for reasons of backward compatibility the first line of the LDIF file that gives the version of the LDIF standard.
db2ldif *args*	Export the contents of the database to LDIF. You must specify either the `-n` or the `-s` option or both. The `db2ldif` subcommand supports the following options:	
	[`-a` *outputfile*]	File name of the output LDIF file.
	[`-C`]	Only use the main db file.
	[`-m`]	Minimal base64 encoding.
	[`-M`]	Use of several files for storing the output LDIF with each instance stored in *instance_file name* (where file name is the file name specified for `-a` option).
	{`-n` *baclemd_instance*}*	Instance to be exported.
	[`-N`]	Specify that the entry IDs are not to be included in the LDIF output. The entry IDs are necessary

	only if the `db2ldif` output is to be used as input to `db2index-text`.
[`-r`]	Export replica.
{`-s` *includesuffix*}*	Suffixes to be included or to specify the subtrees to be included if `-n` has been used.
[{`-x` *excludesuffix*}]*	Suffixes to be excluded.
[`-u`]	Request that the unique id is not exported.
[`-U`]	Request that the output LDIF is not folded.
[`-1`]	Delete, for reasons of backward compatibility, the first line of the LDIF file which gives the version of the LDIF standard.

help [*subcommand*]

Display `directoryserver` usage message or subcommand specific usage message.

idsktune *args*

Provide an easy and reliable way of checking the patch levels and kernel parameter settings for your system. You must install the Directory Server before you can run `idsktune`. It gathers information about the operating system, kernel, and TCP stack to make tuning recommendations.

The `idsktune` subcommand supports the following arguments:

[`-c`]	Client-specific tuning: the output only includes tuning recommendations for running a directory client application.
[`-D`]	Debug mode: the output includes the commands it runs internally, preceded by `DEBUG` heading.
[`-i` *installdir*]	The install directory.

[`-q`]	Quiet mode. Output only includes tuning recommendations. OS version statements are omitted.
[`-v`]	Version. Gives the build date identifying the version of the toll.

keyupg *args*

Upgrade the key from Lite to normal (only one way).

The `keyupg` subcommand supports the following arguments:

`-k`*key*	The key to be upgraded.
`-f` *key_file_path*	The key file path.

ldif2db-task args

Import data to the directory. It create an entry in the directory that launches this dynamic task. The entry is generated based upon the values you provide for each option. The server must be running when you run this subcommand.

The `ldif2sb-task` subcommand supports the following arguments:

[`-c`]	Request that only the core db is created without attribute indexes.
`-D` rootDN	User DN with root permissions, such as Directory Manager.
[`-g` *string*]	Generation of a unique ID. Enter `none` for no unique ID to be generated and `deterministic` for the generated unique ID to be name-based. Generates a time based unique ID by default. If you use the `deterministic` generation to have a name-based unique ID, you can also

	specify the namespace you want the server to use as follows:
`-g deterministic` *namespace_id*	
	where namespace_id is a string of characters in the following format
`00-xxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxxxx`	
	Use this option if you want to import the same LDIF file into two different directory servers, and you want the contents of both directories to have the same set of unique IDs. If unique IDs already exist in the LDIF file you are importing, then the existing IDs are imported to the server regardless of the options you have specified.
[`-G` *namespace_id*]	Generate a namespace ID as a name-based unique ID. This is the same as specifying `-g deterministic`.
{`-i` *filename*}*	File name of the input LDIF files. When you import multiple files, they are imported in the order in which you specify them on the command line.
`-j` *file*	Password associated with the user DN. This option allows the password to be stored in clear text in the named file for scripting. This is considered insecure. Use with extreme caution.
`-n` backend_instance	Instance to be imported.

	[-O]	Request that only the core db is created without attribute indexes.
	{-s *includesuffix* }*	Suffixes to be included. This argument can also be used to specify the subtrees to be included with -n.
	-w *password*	Password associated with the user DN. Supplying the password on the command line is visible using the /bin/ps command. This is considered insecure. Use with extreme caution. The value - can be used in place the password. The program prompts the user for a password to be entered from the terminal.
	[{-x *excludesuffix* }*]	
	[-v]	Verbose mode.
ldif *args*	Format LDIF files, and create base 64 encoded attribute values. With Base 64 Encoding you can represent binary data, such as a JPEG image, in LDIF by using base 64 encoding. You identify base 64 encoded data by using the :: symbol. The ldifsubcommand takes any input and formats it with the correct line continuation and appropriate attribute information. The subcommand also senses whether the input requires base 64 encoding. The ldif subcommand supports the following arguments	
	[-b]	Interpret the entire input as a single binary value. If -b is not present, each line is considered to be a separate input value.
	[*attrtype*]	If -b is specified, the output is attrtype:: <base 64 encoded

`value`.

ldif2db *args*

Import the data to the directory. To run this subcommand the server must be stopped. Note that ldif2db supports LDIF version 1 specifications. You can load an attribute using the URL specifier notation, for example:
`jpegphoto:file:///tmp/myphoto.jpg`

[`-c`] Merge chunk size.

[`-g` *string*] Generation of a unique ID. Type none for no unique ID to be generated and deterministic for the generated unique ID to be name-based. By default a time based unique ID is generated.

If you use the deterministic generation to have a name-based unique ID, you can also specify the namespace you want the server to use as follows:

`-g deterministic` *namespace_id*

where *namespace_id* is a string of characters in the following format:

`00-xxxxxxxx-xxxxxxxx-xxxxxxxx-xxxxxxxx`

Use this option if you want to import the same LDIF file into two different directory servers, and you want the contents of both directories to have the same set of unique IDs. If unique IDs already exist in the LDIF file you are importing, then the existing IDs are imported to the server regardless of the options you have specified.

	[`-G` *naemspace_id*]	Generate a namespace ID as a name-based unique ID. This is the same as specifying the `-g deterministic` option.
	{`-` *filename*}*	File name of the input LDIF file(s). When you import multiple files, they are imported in the order in which you specify them on the command line.
	`-n` *backend_instance*	Instance to be imported.
	[`-O`]	Request that only the core db is created without attribute indexes.
	{`-s` *includesuffix*}*	Suffixes to be included or to specify the subtrees to be included if `-n` has been used.
	[{`-x` *excludesuffix*}*]	Suffixes to be excluded
ldif2ldap *rootDN password filename*	Perform an import operation over LDAP to the Directory Server. To run this subcommand the server must be running.	
	The `ldif2ldap` subcommand supports the following arguments:	
	rootdn	User DN with root permissions, such as Directory Manager.
	password	Password associated with the user DN.
	filename	File name of the file to be imported. When you import multiple files, they are imported in the order in which you specify them on the command line.
magt CONFIG INIT	Start SNMP master agent. The Config and INIT files are in `/usr/iplanet/ds5/plugins/snmp/magt`. For more information, see the *iPlanet Directory Server 5.1 Administrator's Guide*.	
	The `magt` subcommand supports the following options:	

CONFIG	The CONFIG file defines the community and the manager that master agent works with. Specify the manager value as a valid system name or an IP address.
INIT	The INIT file is a nonvolatile file that contains information from the MIB-II system group, including system location and contact information. If INIT doesn't already exist, starting the master agent for the first time creates it. An invalid manager name in the CONFIG file causes the master agent start-up to fail.

monitor

Retrieves performance monitoring information using the `ldapsearch` command-line utility.

mmldif *args*

Combine multiple LDIF files into a single authoritative set of entries. Typically each LDIF file is from a master server cooperating in a multi master replication agreement.[e.g. masters that refuse to sync up for whatever reason]. Optionally, it can generate LDIF change files that could be applied to original to bring it up to date with authoritative. At least two input files must be specified.

The `mmldif` subcommand supports the following arguments:

[`-c` *inputfile* ...]	Write a change file (.delta) for each input file. Specify *inputfile* as the input LDIF files.
[`-D`]	Print debugging information.
[`-o` *out.ldif*]	Write authoritative data to this file.

nativetoascii *args*

Convert one language encoding to another. For example, convert a native language to UTF-8 format.

The `nativetoascii` subcommand supports the following options:

`-d` *Encodings Directory*	Path to the

		directory which contains the `conv` directory
	[`-i` *input_filename* `-o` *output_filename*]	The input file name and output file name.
	`-l`	List supported encodings
	`-r`	Replace existing files.
	`-s` *suffix*	Suffix to be mapped to the backend.
	`-s` *SourceEncoding*	Source Encoding of input stream.
	`-t` *TargetEncoding*	Target Encoding of output stream.
	`-v`	Verbose output.
pwdhash *args*	Print the encrypted form of a password using one of the server's encryption algorithms. If a user cannot log in, you can use this script to compare the user's password to the password stored in the directory. The `pwdhash` subcommand supports the following arguments:	

	`-c` *comparepwd* | `-s` *scheme*	The available schemes are `SSHA`, `SHA`, `CRYPT` and `CLEARE`. It generates the encrypted passwords according to scheme's algorithm. The `-c` specifies the encrypted password to be compared with. The result of comparison is either `OK` or `doesn't match`.
	`-D` *instance-dir*	The instance directory.
	[`-H`]	The passwords are hex-encoded.
	password ...	The clear passwords to generate encrypted form from or to be compared with.
restart	Restarts the directory server. When the `-s` option is not specified, restarts all instances of servers. When the `-s` option is specified, restarts the server specified by `-s`. restart-admin Restarts the administration server.	
restoreconfig	Restores the most recently saved Administration Server configuration information to the NetscapeRoot partition under `/var/ds5/slapd-serverID/confbak`.	
sagt `-c` CONFIG	Start proxy SNMP agent. For more information, see the *iPlanet Directory Server 5.1 Administrator's Guide*. The `sagt` subcommand supports the following options:	

	`-c` *configfile* The CONFIG file includes the port that the SNMP daemon listens to. It also needs to include the MIB trees and traps that the proxy SNMP agent forwards. Edit the CONFIG file located in `/usr/iplanet/ds5/plugins/snmp/sagt`.
saveconfig	Saves the administration server configuration information to the `/var/ds5/slapd-serverID/confbak` directory.
setup [`-f` *configuration_file*]	Configures an instance of the directory server or administration server. Creates a basic configuration for the directory server and the administrative server that is used to manage the directory. The `setup` subcommand has two modes of operation. You can invoke it with a curses-based interaction to gather input. Alternatively, you can provide input in a configuration file using the `-f` option. The setup subcommand supports the following option: `-f` *configuration_file* Specifies the configuration file for silent installation.
start	Starts the directory server. When the `-s` option is not specified, starts servers of all instances. When the `-s` option is specified, starts the server instance specified by `-s`.
start-admin	Starts the directory server. When the `-s` option is not specified, restarts all instances of servers. When the `-s` option is specified, restarts the server specified by `-s`.
startconsole	Starts the directory console..
stop	Stops the directory server. When the `-s` option is not specified, restarts all instances of servers. When the `-s` option is specified, restarts the server specified by `-s`.
stop-admin	Stop the administration server.

suffix2instance {`-s` *suffix*}
: Map a suffix to a backend name.

  Specify `-s` *suffix* as the suffix to be mapped to the backend.

uninstall
: Uninstalls the directory server and the administration server.

  This subcommand stops servers of all instances and removes all the changes created by setup.

vlvindex *args*
: Create virtual list view (VLV) indexes, known in the Directory Server Console as Browsing Indexes. The server must be stopped beforehand.

  The `vlvindex` subcommand supports the following arguments:

  `-d` *debug_level*
  : Specify the debug level to use during index creation. Debug levels are defined in `nsslapd-errorlog-level (error Log Level)`. See the *iPlanet Directory Server 5.1 Configuration, Command, and File Reference*.

  `-n` *backend_instance*
  : Name of the database containing the entries to index.

  `-s` *suffix*
  : Name of the suffix containing the entries to index.

  `-T` *VLVTag*
  : Name of the database containing the entries to index.

Options Options for the `directoryserver` command itself must appear before the subcommand argument.

The following options are supported:

`-s` *server-instance*
`-server` *server-instance*
: The server instance name. Specify the directory server instance to process the command against. For some of the listed subcommands the server instance is optional and for other sub

commands it is a required option.

Examples

EXAMPLE 1 Starting All Instances of the Directory Servers

The following command starts all the instances of the directory servers:

```
example% directoryserver start
```

EXAMPLE 2 Starting the Instances of `myhost` of the Directory Server

The following command starts the instances `myhost` of the directory server.

```
example% directoryserver -s myhost start
```

EXAMPLE 3 Running the Monitor Tool and Outputting the Current Status

The following command runs the monitor tool and output the current status of the `ephesus` directory instance.

```
example% directoryserver -s ephesus monitor
```

EXAMPLE 4 Running the idsktune Tool and Outputting Performance Tuning Information

The following command runs the idsktune tool and outputs performance tuning information:

```
example% directoryserver idsktune
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	IPLTdsr, IPLTdsu

See Also

iPlanet Directory Server 5.1 Administrator's Guide

iPlanet Directory Server 5.1 Configuration, Command, and File Reference

Name disks – creates /dev entries for hard disks attached to the system

Synopsis `/usr/sbin/disks [-C] [-r rootdir]`

Description `devfsadm(1M)` is now the preferred command for `/dev` and should be used instead of `disks`.

`disks` creates symbolic links in the `/dev/dsk` and `/dev/rdsk` directories pointing to the actual disk device special files under the `/devices` directory tree. It performs the following steps:

1. `disks` searches the kernel device tree to see what hard disks are attached to the system. It notes the `/devices` pathnames for the slices on the drive and determines the physical component of the corresponding `/dev/dsk` or `/dev/rdsk` name.
2. The `/dev/dsk` and `/dev/rdsk` directories are checked for disk entries – that is, symbolic links with names of the form `c`*N*`[t`*N*`]d`*N*`s`*N*, or `c`*N*`[t`*N*`]dNp`*N*, where *N* represents a decimal number. `c`*N* is the logical controller number, an arbitrary number assigned by this program to designate a particular disk controller. The first controller found on the first occasion this program is run on a system, is assigned number `0`. `t`*N* is the bus-address number of a subsidiary controller attached to a peripheral bus such as SCSI or IPI (the `target` number for SCSI, and the `facility` number for IPI controllers). `d`*N* is the number of the disk attached to the controller. `s`*N* is the *slice* number on the disk. `p`*N* is the FDISK partition number used by `fdisk(1M)`. (x86 Only)
3. If only some of the disk entries are found in `/dev/dsk` for a disk that has been found under the `/devices` directory tree, disks creates the missing symbolic links. If none of the entries for a particular disk are found in `/dev/dsk`, `disks` checks to see if any entries exist for other disks attached to the same controller, and if so, creates new entries using the same controller number as used for other disks on the same controller. If no other `/dev/dsk` entries are found for slices of disks belonging to the same physical controller as the current disk, `disks` assigns the lowest-unused controller number and creates entries for the disk slices using this newly-assigned controller number.

`disks` is run automatically each time a reconfiguration-boot is performed or when `add_drv(1M)` is executed. When invoking `disks(1M)` manually, first run `drvconfig(1M)` to ensure `/devices` is consistent with the current device configuration.

Notice to Driver Writers `disks` considers all devices with a node type of `DDI_NT_BLOCK`, `DDI_NT_BLOCK_CHAN`, `DDI_NT_CD`, `DDI_NT_BLOCK_WWN` or `DDI_NT_CD_CHAN` to be disk devices. `disks(1M)` requires the minor name of disk devices obey the following format conventions.

The minor name for block interfaces consists of a single lowercase ASCII character, `a` through `u`. The minor name for character (raw) interfaces consists of a single lowercase ASCII character, `a` through `u`, followed by `,raw`.

`disks` translates `a` through `p` to `s0` through `s15`, while it translates `q` through `u` to `p0` through `p4`. SPARC drivers should only use the first 8 slices: `a` through `h`, while x86 drivers can use `a` through `u`, with `q` through `u` corresponding to `fdisk(1M)` partitions. `q` represents the entire disk, while `r`, `s`, `t`, and `u` represent up to 4 additional partitions.

To prevent `disks` from attempting to automatically generate links for a device, drivers must specify a private node type and refrain from using a node type: `DDI_NT_BLOCK`, `DDI_NT_BLOCK_CHAN`, `DDI_NT_CD`, or `DDI_NT_CD_CHAN` when calling `ddi_create_minor_node(9F)`.

Options The following options are supported:

`-C` Causes disks to remove any invalid links after adding any new entries to `/dev/dsk` and `/dev/rdsk`. Invalid links are links which refer to non-existent disk nodes that have been removed, powered off, or are otherwise inaccessible.

`-r` *rootdir* Causes `disks` to presume that the `/dev/dsk`, `/dev/rdsk` and `/devices` directory trees are found under *rootdir*, not directly under `/`.

Errors If disks finds entries of a particular logical controller linked to different physical controllers, it prints an error message and exits without making any changes to the `/dev` directory, since it cannot determine which of the two alternative logical-to-physical mappings is correct. The links should be manually corrected or removed before another reconfiguration-boot is performed.

Examples **EXAMPLE 1** Creating Block and Character Minor Devices

The following example demonstrates creating the block and character minor devices from within the `xkdisk` driver's `attach(9E)` function.

```
#include    <sys/dkio.h>
/*
 * Create the minor number by combining the instance number
 * with the slice number.
 */
#define MINOR_NUM(i, s)   ((i) << 4 | (s))

int
xkdiskattach(dev_info_t *dip, ddi_attach_cmd_t cmd)
{
    int instance, slice;
    char name[8];

        /* other stuff in attach... */

    instance = ddi_get_instance(dip);
    for (slice = 0; slice < V_NUMPAR; slice++) {
        /*
         * create block device interface
         */
        sprintf(name, "%c", slice + 'a');
        ddi_create_minor_node(dip, name, S_IFBLK,
            MINOR_NUM(instance, slice), DDI_NT_BLOCK_CHAN, 0);
```

EXAMPLE 1 Creating Block and Character Minor Devices *(Continued)*

```
        /*
         * create the raw (character) device interface
         */
        sprintf(name,"%c,raw", slice + 'a');
        ddi_create_minor_node(dip, name, S_IFCHR,
            MINOR_NUM(instance, slice), DDI_NT_BLOCK_CHAN, 0);
    }
}
```

Installing the `xkdisk` disk driver on a Sun Fire 4800, with the driver controlling a SCSI disk (target 3 attached to an `isp(7D)` SCSI HBA) and performing a reconfiguration-boot (causing disks to be run) creates the following special files in `/devices`.

```
# ls -l /devices/ssm@0,0/pci@18,700000/pci@1/SUNW,isptwo@4/
brw-r-----   1 root sys   32, 16 Aug 29 00:02 xkdisk@3,0:a
crw-r-----   1 root sys   32, 16 Aug 29 00:02 xkdisk@3,0:a,raw
brw-r-----   1 root sys   32, 17 Aug 29 00:02 xkdisk@3,0:b
crw-r-----   1 root sys   32, 17 Aug 29 00:02 xkdisk@3,0:b,raw
brw-r-----   1 root sys   32, 18 Aug 29 00:02 xkdisk@3,0:c
crw-r-----   1 root sys   32, 18 Aug 29 00:02 xkdisk@3,0:c,raw
brw-r-----   1 root sys   32, 19 Aug 29 00:02 xkdisk@3,0:d
crw-r-----   1 root sys   32, 19 Aug 29 00:02 xkdisk@3,0:d,raw
brw-r-----   1 root sys   32, 20 Aug 29 00:02 xkdisk@3,0:e
crw-r-----   1 root sys   32, 20 Aug 29 00:02 xkdisk@3,0:e,raw
brw-r-----   1 root sys   32, 21 Aug 29 00:02 xkdisk@3,0:f
crw-r-----   1 root sys   32, 21 Aug 29 00:02 xkdisk@3,0:f,raw
brw-r-----   1 root sys   32, 22 Aug 29 00:02 xkdisk@3,0:g
crw-r-----   1 root sys   32, 22 Aug 29 00:02 xkdisk@3,0:g,raw
brw-r-----   1 root sys   32, 23 Aug 29 00:02 xkdisk@3,0:h
crw-r-----   1 root sys   32, 23 Aug 29 00:02 xkdisk@3,0:h,raw
```

/dev/dsk will contain the disk entries to the block device nodes in `/devices`

```
# ls -l /dev/dsk
/dev/dsk/c0t3d0s0 -> ../../devices/[...]/xkdisk@3,0:a
/dev/dsk/c0t3d0s1 -> ../../devices/[...]/xkdisk@3,0:b
/dev/dsk/c0t3d0s2 -> ../../devices/[...]/xkdisk@3,0:c
/dev/dsk/c0t3d0s3 -> ../../devices/[...]/xkdisk@3,0:d
/dev/dsk/c0t3d0s4 -> ../../devices/[...]/xkdisk@3,0:e
/dev/dsk/c0t3d0s5 -> ../../devices/[...]/xkdisk@3,0:f
/dev/dsk/c0t3d0s6 -> ../../devices/[...]/xkdisk@3,0:g
/dev/dsk/c0t3d0s7 -> ../../devices/[...]/xkdisk@3,0:h
```

and /dev/rdsk will contain the disk entries for the character device nodes in `/devices`

EXAMPLE 1 Creating Block and Character Minor Devices *(Continued)*

```
# ls -l /dev/rdsk
/dev/rdsk/c0t3d0s0 -> ../../devices/[...]/xkdisk@3,0:a,raw
/dev/rdsk/c0t3d0s1 -> ../../devices/[...]/xkdisk@3,0:b,raw
/dev/rdsk/c0t3d0s2 -> ../../devices/[...]/xkdisk@3,0:c,raw
/dev/rdsk/c0t3d0s3 -> ../../devices/[...]/xkdisk@3,0:d,raw
/dev/rdsk/c0t3d0s4 -> ../../devices/[...]/xkdisk@3,0:e,raw
/dev/rdsk/c0t3d0s5 -> ../../devices/[...]/xkdisk@3,0:f,raw
/dev/rdsk/c0t3d0s6 -> ../../devices/[...]/xkdisk@3,0:g,raw
/dev/rdsk/c0t3d0s7 -> ../../devices/[...]/xkdisk@3,0:h,raw
```

Files

`/dev/dsk/*` Disk entries (block device interface)

`/dev/rdsk/*` Disk entries (character device interface)

`/devices/*` Device special files (minor device nodes)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `add_drv(1M)`, `devfsadm(1M)`, `fdisk(1M)`, `attributes(5)`, `isp(7D)`, `devfs(7FS)`, `dkio(7I)`, `attach(9E)`, `ddi_create_minor_node(9F)`

Writing Device Drivers

Bugs `disks` silently ignores malformed minor device names.

Name diskscan – perform surface analysis

Synopsis `diskscan [-W] [-n] [-y]` *raw_device*

Description `diskscan` is used by the system administrator to perform surface analysis on a portion of a hard disk. The disk portion may be a raw partition or slice; it is identified using its raw device name. By default, the specified portion of the disk is read (non-destructive) and errors reported on standard error. In addition, a progress report is printed on standard out. The list of bad blocks should be saved in a file and later fed into `addbadsec(1M)`, which will remap them.

Options The following options are supported:

`-n` Causes `diskscan` to suppress linefeeds when printing progress information on standard out.

`-W` Causes `diskscan` to perform write and read surface analysis. This type of surface analysis is destructive and should be invoked with caution.

`-y` Causes `diskscan` to suppress the warning regarding destruction of existing data that is issued when `-W` is used.

Operands The following operands are supported:

raw_device The address of the disk drive (see `FILES`).

Files The raw device should be `/dev/rdsk/c?[t?]d?[ps]?`. See `disks(1M)` for an explanation of SCSI and IDE device naming conventions.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWcsu

See Also `addbadsec(1M)`, `disks(1M)`, `fdisk(1M)`, `fmthard(1M)`, `format(1M)`, `attributes(5)`

Notes The `format(1M)` utility is available to format, label, analyze, and repair SCSI disks. This utility is included with the `diskscan`, `addbadsec(1M)`, `fdisk(1M)`, and `fmthard(1M)` commands available for x86. To format an IDE disk, use the DOS `format` utility; however, to label, analyze, or repair IDE disks on x86 systems, use the Solaris `format(1M)` utility.

Name dispadmin – process scheduler administration

Synopsis

```
dispadmin -l

dispadmin -c class -g [-r res]

dispadmin -d [class]
```

Description The `dispadmin` command displays or changes process scheduler parameters while the system is running.

`dispadmin` does limited checking on the values supplied in *file* to verify that they are within their required bounds. The checking, however, does not attempt to analyze the effect that the new values have on the performance of the system. Inappropriate values can have a negative effect on system performance. (See *System Administration Guide: Basic Administration*

Options The following options are supported:

`-c` *class* Specifies the class whose parameters are to be displayed or changed. Valid *class* values are: `RT` for the real-time class, `TS` for the time-sharing class, `IA` for the inter-active class, `FSS` for the fair-share class, and FX for the fixed-priority class. The time-sharing and inter-active classes share the same scheduler, so changes to the scheduling parameters of one will change those of the other.

`-d` [*class*] Sets or displays the name of the default scheduling class to be used on reboot when starting `svc:/system/scheduler:default`. If class name is not specified, the name and description of the current default scheduling class is displayed. If class name is specified and is a valid scheduling class name, then it is saved in `dispadmin`'s private configuration file `/etc/dispadmin.conf`. Only super-users can set the default scheduling class.

`-g` Gets the parameters for the specified class and writes them to the standard output. Parameters for the real-time class are described in `rt_dptbl(4)`. Parameters for the time-sharing and inter-active classes are described in `ts_dptbl(4)`. Parameters for the fair-share class are described in `FSS(7)`. Parameters for the fixed-priority class are described in `fx_dptbl(4)`.

The `-g` and `-s` options are mutually exclusive: you may not retrieve the table at the same time you are overwriting it.

`-l` Lists the scheduler classes currently configured in the system.

`-r` *res* When using the `-g` option you may also use the `-r` option to specify a resolution to be used for outputting the time quantum values. If no resolution is specified, time quantum values are in milliseconds. If *res* is specified it must be a positive integer between 1 and 1000000000 inclusive, and the resolution used is the reciprocal of *res* in seconds. For example, a *res* value of 10 yields time quantum values expressed in tenths of a second; a *res* value of 1000000 yields time quantum values expressed in microseconds. If the time quantum cannot be

expressed as an integer in the specified resolution, it is rounded up to the next integral multiple of the specified resolution.

`-s` *file* Sets scheduler parameters for the specified class using the values in *file*. These values overwrite the current values in memory—they become the parameters that control scheduling of processes in the specified class. The values in *file* must be in the format output by the `-g` option. Moreover, the values must describe a table that is the same size (has same number of priority levels) as the table being overwritten. Super-user privileges are required in order to use the `-s` option.

Specify time quantum values for scheduling classes in system clock ticks, and not in constant-time units. Time quantum values are based on the value of the kernel's `hz` variable. If kernel variable `hires_tick` is set to `1` to get higher resolution clock behavior, the actual time quanta will be reduced by the order of `10`.

The `-g` and `-s` options are mutually exclusive: you may not retrieve the table at the same time you are overwriting it.

Examples

EXAMPLE 1 Retrieving the Current Scheduler Parameters for the real-time class

The following command retrieves the current scheduler parameters for the real-time class from kernel memory and writes them to the standard output. Time quantum values are in microseconds.

```
dispadmin -c RT -g -r 1000000
```

EXAMPLE 2 Overwriting the Current Scheduler Parameters for the Real-time Class

The following command overwrites the current scheduler parameters for the real-time class with the values specified in `rt.config`.

```
dispadmin -c RT -s rt.config
```

EXAMPLE 3 Retrieving the Current Scheduler Parameters for the Time-sharing Class

The following command retrieves the current scheduler parameters for the time-sharing class from kernel memory and writes them to the standard output. Time quantum values are in nanoseconds.

```
dispadmin -c TS -g -r 1000000000
```

EXAMPLE 4 Overwriting the Current Scheduler Parameters for the Time-sharing Class

The following command overwrites the current scheduler parameters for the time-sharing class with the values specified in `ts.config`.

```
dispadmin -c TS -s ts.config
```

Files `/etc/dispadmin.conf`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `priocntl(1)`, `svcs(1)`, `svcadm(1M)`, `priocntl(2)`, `fx_dptbl(4)`, `rt_dptbl(4)`, `ts_dptbl(4)`, `attributes(5)`, `smf(5)`, `FSS(7)`

System Administration Guide: Basic Administration Programming Interfaces Guide

Diagnostics `dispadmin` prints an appropriate diagnostic message if it fails to overwrite the current scheduler parameters due to lack of required permissions or a problem with the specified input file.

Notes The default scheduling class setting facility is managed by the service management facility,`smf(5)`, under the service identifier:

```
svc:/system/scheduler:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Note that disabling the service while it is running will not change anything. The service's status can be queried using the `svcs(1)` command.

Name dladm – configure data-link interfaces

Synopsis

```
dladm show-link [-s [-i interval]] [-p] [name]

dladm show-dev [-s [-i interval]] [-p] [dev]

dladm create-aggr [-t] [-R root-dir] [-P policy] [-l mode]
     [-T time] [-u address] -d dev [-d dev] ... key

dladm delete-aggr [-t] [-R root-dir] key

dladm add-aggr [-t] [-R root-dir] -d dev [-d dev] ... key

dladm remove-aggr [-t] [-R root-dir] -d dev [-d dev] ... key

dladm modify-aggr [-t] [-R root-dir] [-P policy] [-l mode]
     [-T time] [-u address] key

dladm show-aggr [-L] [-s [-i interval]] [-p] [key]

dladm set-linkprop [-t] [-R root-dir] -p prop=value[,...] name

dladm reset-linkprop [-t] [-R root-dir] [-p prop,...] name

dladm show-linkprop [-cP] [-p prop,...] [name]

dladm -?
```

Description The `dladm` command is used to configure data-links. A configured data-link is represented in the system as a STREAMS DLPI (v2) interface which may be plumbed under protocol stacks such as TCP/IP. Each data-link relies on either a single network device or an aggregation of devices to send packets to or receive packets from a network.

The `dladm` command operates on the following kinds of object:

link
: Data-links, identified by a name. A name is a maximum of 30 characters. The first character must be alphabetic, the last numeric.

aggr
: Aggregations of network devices, identified by a key.

dev
: Network devices, identified by concatenation of a driver name and an instance number.

The behavior of the `linkprop` subcommands depends on the type of link and underlying device, currently only one `linkprop`, "zone" is supported.

Some devices do not support configurable data-links or aggregations. The fixed data-links provided by such devices can be viewed using `dladm`, but can not be configured.

SUBCOMMANDS The following subcommands are supported:

`show-link`
Show configuration information for all data-links or the specified data-link. By default, the system is configured to have one data-link for each known network device.

`show-dev`
Shows information for all devices or the specified device.

`create-aggr`
Creates an aggregation using the given key value from as many *dev* objects as are specified. A data-link is created by default, and is given a name which is the concatenation of "aggr" and the key value of the aggregation.

`delete-aggr`
Deletes the specified aggregation.

`add-aggr`
Adds as many *dev* objects as are specified to the given aggregation.

`remove-aggr`
Removes as many *dev* objects as are specified from the given aggregation.

`modify-aggr`
Modifies the parameters of the given aggregation.

`show-aggr`
Shows configuration information for all aggregations or the specified aggregation.

`set-linkprop`
Sets the values of one or more properties on the link specified by *name*. The list of properties and their possible values depend on the link type, the network device driver, and networking hardware, but can be retrieved using `show-linkprop`.

`reset-linkprop`
Resets one or more properties to their values on the link specified by *name*. If no properties are specified, all properties are reset.

`show-linkprop`
Show the current values of one or more properties, either for all data-links or for the specified link name. If no properties are specified, all available link properties are displayed.

Options The following options are supported:

`-k` *key*
`--key=`key
The key of an aggregation. This must be an integer value between 1 and 999.

`-d` *dev*
`--dev=`*dev*
A device specifier. This must be a concatenation of the name and instance of the driver bound to the device.

`-P` *policy*
`--policy=`policy
Specifies the port selection policy to use for load spreading of outbound traffic. The policy specifies which *dev* object is used to send packets. A policy consists of a list of one or more layers specifiers separated by commas. A layer specifier is one of the following:

L2
Select outbound device according to source and destination MAC addresses of the packet.

L3
Select outbound device according to source and destination IP addresses of the packet.

L4
Select outbound device according to the upper layer protocol information contained in the packet. For TCP and UDP, this includes source and destination ports. For *IPsec*, this includes the SPI (Security Parameters Index.)

In the absence of a policy specification, `dladm` uses the default, L4.

As an example of use of the L*num* identifiers, to use upper layer protocol information, specify the following policy:

```
-P L4
```

Note that, as the default, specification of L4 is superfluous.

To use the source and destination MAC addresses as well as the source and destination IP addresses, the following policy can be used:

```
-P L2,L3
```

`-l` *mode*
`--lacp-mode=`*mode*
Specifies whether LACP should be used and, if used, the mode in which it should operate. Legal values are `off`, `active` or `passive`.

`-T` *time*
`--lacp-timer=`*time*
Specifies the LACP timer value. The legal values are short or long.

`-u` *address*
`--unicast=`address
Specifies a fixed unicast address to be used for the aggregation. If this option is not specified then an address is automatically chosen from the set of addresses of the component devices.

`-L`
`--lacp`
Specifies whether detailed LACP information should be displayed.

`-s`
`--statistics`
Used with the `show-link`, `show-aggr`, or `show-dev` subcommands to show the statistics of data-links, aggregations or devices, respectively.

`-i` *interval*
`--interval=`interval
Used with the `-s` option to specify an interval, in seconds, at which statistics should be displayed. If this option is not specified, statistics will only be displayed once.

`-t`
`--temporary`
Specifies that the change is temporary. Temporary changes last until the next reboot.

`-R` *root-dir*
`--root-dir=`root-dir
Specifies an alternate root directory where `dladm` applies changes. This can be useful in JumpStart scripts, where the root directory of the system being modified is mounted elsewhere.

`-p`
`--parseable`
Specifies that configuration information should be displayed in parseable format.

`-?`
`--help`
Displays help information. (Stops interpretation of subsequent arguments).

LINK PROPERTIES

The following link properties listed below are supported. Note that these properties can be modified only temporarily through `dladm`, and thus the `-t` option must be specified. See the `NOTES` section for instructions on how to make property values persistent.

`zone`
Specifies the zone to which the link belongs. Possible values consist of any exclusive-IP zone currently running on the system. By default, the zone binding is as per `zonecfg(1M)`.

`tagmode`
This link property controls the conditions in which 802.1Q VLAN tags will be inserted in packets being transmitted on the link. Two mode values can be assigned to this property:

`normal`
Insert a VLAN tag in outgoing packets under the following conditions:

- The packet belongs to a VLAN.
- The user requested priority tagging.

`vlanonly`
Insert a VLAN tag only when the outgoing packet belongs to a VLAN. If a tag is being inserted in this mode and the user has also requested a non-zero priority, the priority is honored and included in the VLAN tag.

The default value is `vlanonly`.

Examples

EXAMPLE 1 Configuring an Aggregation

To configure a data-link over an aggregation of devices *bge0* and *bge1* with key 1, enter the following command:

```
# dladm create-aggr -d bge0 -d bge1 1
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

`/usr/sbin`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

`/sbin`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr
Interface Stability	Evolving

See Also

`ifconfig(1M)`, `zonecfg(1M)`, `attributes(5)`, `smf(5)`, `dlpi(7P)`

Notes

There are two ways to make values for link properties persistent across reboots:

- Using `rc` scripts.
- Writing a transient `smf(5)` service.

The second method, using `smf(5)`, is preferred.

To use the `rc` feature, perform steps such as the following:

1. Create a shell script, with permissions 744.
2. Store the script in `/etc/rc3.d`.
3. Inside the script, enter a command such as the following:

   ```
   /usr/sbin/dladm set-linkprop -t -p tagmode=vlanonly ce1
   ```

   In this example, it is the value for the `tagmode` property that is being made persistent. The interface `ce1` is also particular to this example. Your interface name might be different.

See `/etc/rc3.d/README` for further guidance.

The second, preferred means of making values persistent is to use the smf(5) facility. To do so, perform steps such as the following:

1. Compose a manifest file. The following is an example of such a file:

```
<service_bundle type='manifest'name='apply_linkprop'>
<service
        name='network/apply_linkprop'
        type='service'
        version='1'>

        <instance name='default' enabled='true'>

        <dependency
                name='dlmgmtd'
                grouping='require_all'
                restart_ov='none'
                type='service'>
        <service_fmri value='svc:/network/datalink-management:default' />
        </dependency>

        <exec_method
                type='method'
                name='stop'
                exec=':true'
                timeout_seconds='3' />

        <property_group name='startd' type='framework'>
                <propval name='duration' type='astring' value='transient' />
        </property_group>

        </instance>

           <stability value='Evolving' />
</service>

</service_bundle>
```

   Store this file in `/lib/svc/manifest/network/`.

2. Create a shell script in `/lib/svc/method` that contains:

```
/usr/sbin/dladm set-linkprop -t tagmode=vlanonly ce1
```

   The property, `tagmode`, and interface name, `ce1`, are examples. Use the names appropriate for your system.

Any additional properties you want to make persistent should be added, as separate commands, to the preceding shell script. There is no need to create an additional manifest file.

Name dmesg – collect system diagnostic messages to form error log

Synopsis

```
/usr/bin/dmesg

/usr/sbin/dmesg
```

Description `dmesg` is made obsolete by `syslogd(1M)` for maintenance of the system error log.

`dmesg` looks in a system buffer for recently printed diagnostic messages and prints them on the standard output.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu

See Also `syslogd(1M)`, `attributes(5)`

Name dmi_cmd – DMI command line interface utility

Synopsis

```
dmi_cmd  -AL -c compId -g groupId [-dp] [-a attrId] [-m max-count]
 [-r req-mode] [-s hostname]

dmi_cmd  -CD -c compId [-s hostname]

dmi_cmd  -CI mif-file [-s hostname]

dmi_cmd  -CL [-dp] [-c compId] [-m max-count] [-r req-mode]
 [-s hostname]

dmi_cmd  -GD -c compId -g groupId [-s hostname]

dmi_cmd  -GI schema-file -c compId [-s hostname]

dmi_cmd  -GL -c compId -g groupId [-dp] [-m max-count]
 [-r req-mode] [-s hostname]

dmi_cmd  -GM -c compId [-m max-count] [-s hostname]

dmi_cmd  -h

dmi_cmd  -ND -c compId -l language-string [-s hostname]

dmi_cmd  -NI schema-file -c compId [-s hostname]

dmi_cmd  -NL -c compId [-s hostname]

dmi_cmd  -V [-s hostname]

dmi_cmd  -W config-file [-s hostname]

dmi_cmd  -X [-s hostname]
```

Description The `dmi_cmd` utility provides the ability to:

- Obtain version information about the DMI Service Provider
- Set the configuration to describe the language required by the management application
- Obtain configuration information describing the current language in use for the session
- Install components into the database
- List components in a system to determine what is installed
- Delete an existing component from the database
- Install group schemas to an existing component in the database
- List class names for all groups in a component
- List the groups within a component
- Delete a group from a component
- Install a language schema for an existing component in the database
- List the set of language mappings installed for a specified component
- Delete a specific language mapping for a component

- List the properties for one or more attributes in a group

Note that this command is Obsolete.

Options The following options are supported:

`-a` *attrId*	Specify an attribute by its ID (positive integer). The default value is `0`.
`-AL`	List the attributes for the specified component.
`-c` *compId*	Specify a component by its ID (positive integer). The default value is `0`.
`-CD`	Delete the specified component.
`-CI` *mif-file*	Install the component described in the *mif-file*.
`-CL`	List component information.
`-d`	Display descriptions.
`-g` *groupId*	Specify a group by its ID (positive integer). The default value is `0`.
`-GD`	Delete a group for the specified component.
`-GI` *schema-file*	Install the group schema specified in *schema-file*.
`-GL`	List the groups for the specified component.
`-GM`	List the class names for the specified component.
`-h`	Help. Print the command line usage.
`-l` *language-string*	Specify a language mapping.
`-m` *max-count*	Specify the maximum number of components to display.
`-ND`	Delete a language mapping for the specified component.
`-NI` *schema-file*	Install the language schema specified in *schema-file*.
`-NL`	List the language mappings for a specified component.
`-p`	Display the pragma string.
`-r` *req-mode*	Specify the request mode. The valid values are: `1` DMI_UNIQUE - access the specified item (or table row). `2` DMI_FIRST - access the first item. `3` DMI_NEXT - access the next item.

The default request mode is 1 DMI_UNIQUE.

`-s` *hostname* — Specify the host machine on which `dmispd` is running. The default host is the local host.

`-V` — Version. Prints version information about the DMI Service Provider.

`-W` *config-file* — Set the configuration specified in *config-file* to `dmispd`.

`-X` — Retrieve configuration information describing the current language in use.

Exit Status The following error values are returned:

`0` Successful completion.

`−1` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi
Interface Stability	Obsolete

See Also `dmiget(1M)`, `dmispd(1M)`, `attributes(5)`

Name dmiget – DMI command line retrieval utility

Synopsis

```
dmiget  -c compId [-a attrId] [-g groupId] [-s hostname]
dmiget  -h
```

Description The `dmiget` utility retrieves the table information of a specific component in the DMI Service Provider.

Options The following options are supported:

`-a` *attrId*	Display the attribute information for the component specified with the `-c` argument.
`-c` *compId*	Display all the table information for the specified component.
`-g` *groupId*	Display all the attribute information in the group specified with *groupId* for the component specified with the `-c` argument
`-h`	Help. Print the command line usage.
`-s` *hostname*	Specify the host machine on which `dmispd` is running. The default host is the local host.

Exit Status The following error values are returned:

`0`	Successful completion.
`–1`	An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi
Interface Stability	Obsolete

See Also `dmi_cmd(1M)`, `dmispd(1M)`, `attributes(5)`

Name dminfo – report information about a device entry in a device maps file

Synopsis

```
dminfo [-v] [-a] [-f pathname]
dminfo [-v] [-a] [-f pathname] -n dev -name...
dminfo [-v] [-a] [-f pathname] -d dev -path...
dminfo [-v] [-a] [-f pathname] -t dev -type...
dminfo [-v] [-f pathname] -u dm -entry
```

Description `dminfo` reports and updates information about the `device_maps(4)` file.

Options The following options are supported

`-a`	Succeed if any of the requested entries are found. If used with `-v`, all entries that match the requested case(s) are printed.
`-d` *dev–path*	Search by *dev–path*. Search `device_maps(4)` for a device special pathname in the *device_list* field matching the *dev–path* argument. This option cannot be used with `-n`, `-t` or `-u`.
`-f` *pathname*	Use a device_maps file with *pathname* instead of `/etc/security/device_maps`.
`-n` *dev–name*	Search by *dev–name*. Search `device_maps(4)` for a *device_name* field matching *dev–name*. This option cannot be used with `-d`, `-t` or `-u`.
`-t` *dev–type*	Search by *dev–type*. Search `device_maps(4)` for a *device_type* field matching the given *dev–type*. This option cannot be used with `-d`, `-n` or `-u`.
`-u` *dm–entry*	Update the `device_maps(4)` file. This option is provided to add entries to the `device_maps(4)` file. The *dm–entry* must be a complete `device_maps(4)` file entry. The *dm–entry* has fields, as in the `device_maps` file. It uses the colon (`:`) as a field separator, and white space as the *device_list* subfield separators. The *dm–entry* is not made if any fields are missing, or if the *dm–entry* would be a duplicate. The default device maps file can be updated only by the super user.
`-v`	Verbose. Print the requested entry or entries, one line per entry, on the standard output. If no entries are specified, all are printed.

Exit Status

0	Successful completion.
1	Request failed.
2	Incorrect syntax.

Files `/etc/security/device_maps`

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also bsmconv(1M), device_maps(4), attributes(5)

Notes The functionality described in this man page is available only if the Solaris Auditing feature has been enabled. See bsmconv(1M) for more information.

Name dmispd – Sun Solstice Enterprise DMI Service Provider

Synopsis `/usr/lib/dmi/dmispd [-h] [-c` *config-dir*`] [-d` *debug-level*`]`

Description The DMI Service Provider, `dmispd`, is the core of the DMI solution. Management applications and Component instrumentations communicate with each other through the Service Provider. The Service Provider coordinates and arbitrates requests from the management application to the specified component instrumentations. The Service Provider handles runtime management of the Component Interface (CI) and the Management Interface (MI), including component installation, registration at the MI and CI level, request serialization and synchronization, event handling for CI, and general flow control and housekeeping.

The Service Provider is invoked from a start-up script at boot time only if contents of the DMI Service Provider configuration file `/etc/dmi/conf/dmispd.conf` are non-trivial.

Options The following options are supported:

`-c` *config-dir* Specify the full path of the directory containing the `dmispd.conf` configuration file. The default directory is `/etc/dmi/conf`.

`-d` *debug-level* Debug. Levels from 0 to 5 are supported, giving various levels of debug information. The default is 0, meaning no debug information is given.

If this option is omitted, then `dmispd` is run as a daemon process.

`-h` Help. Print the command line usage.

Exit Status The following error values are returned:

`0` Successful completion.

`1` An error occurred.

Files `/etc/dmi/conf/dmispd.conf` DMI Service Provider configuration file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi
Interface Stability	Obsolete

See Also `snmpXdmid(1M)`, `attributes(5)`

Name dnssec-dsfromkey – DNSSEC DS RR generation tool

Synopsis

```
dnssec-dsfromkey [-v level] [-1] [-2] [-a alg] keyfile

dnssec-dsfromkey -s [-v level] [-1] [-2] [-a alg] [-c class]
     [-d dir] keyfile
```

Description `dnssec-dsfromkey`

Options The following options are supported:

`-1`
Use `SHA-1` as the digest algorithm. The default is to use both `SHA-1` and `SHA-256`.

`-2`
Use SHA-256 as the digest algorithm.

`-a` *algorithm*
Select the digest algorithm. The value of *algorithm* must be one of `SHA-1` (`SHA1`) or `SHA-256` (`SHA256`). These values are case-insensitive.

`-v` *level*
Sets the debugging level.

`-s`
Keyset mode: in place of the keyfile name, the argument is the DNS domain name of a keyset file. The `-c` and `-d` options have meaning only in this mode.

`-c` *class*
Specifies the DNS class (default is `IN`); useful only in the keyset mode.

`-d` *directory*
Look for keyset files in directory as the directory; ignored when not in the keyset mode.

Examples To build the SHA-256 DS RR from the `Kexample.com.+003+26160` keyfile name, use a command such as the following:

```
# dnssec-dsfromkey -2 Kexample.com.+003+26160
```

This command would produce output similar to the following:

```
example.com. IN DS 26160 5 2
3A1EADA7A74B8D0BA86726B0C227AA85AB8BBD2B2004F41A868A54F0
C5EA0B94
```

Files The keyfile can be designated by the key identification K*nnnn*.+*aaa*+*iiiii*, or the full file name K*nnnn*.+*aaa*+*iiiii*.key, as generated by `dnssec-keygen(1M)`.

The keyset file name is built from the directory, the string `keyset-` and the *dnsname*.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also dnssec-keygen(1M), dnssec-signzone(1M), attributes(5)

RFC 3658, *RFC 4509*

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at https://www.isc.org/software/bind/documentation.

Caution A keyfile error can produce a "file not found" message, even if the file exists.

Name dnssec-keyfromlabel – DNSSEC key generation tool

Synopsis

```
dnssec-keyfromlabel -a algorithm -l label [-c class] [-f flag] [-k]
     [-n nametype] [-p protocol] [-t type] [-v level] name
```

Description `dnssec-keyfromlabel` retrieves keys with a specified label from a crypto hardware device and builds key files for DNSSEC (Secure DNS), as defined in RFC 2535 and RFC 4034.

Options The following options are supported:

`-a` *algorithm*
Selects the cryptographic algorithm. The value of *algorithm* must be one of `RSAMD5` (RSA) or `RSASHA1`, `DSA`, `NSEC3RSASHA1`, `NSEC3DSA`, or `DH` (Diffie-Hellman). These values are case-insensitive.

Note that for `DNSSEC`, `RSASHA1` is a mandatory-to-implement algorithm, and DSA is recommended. Note also that `DH` automatically sets the `-k` flag.

`-l` *label*
Specifies the label of keys in the crypto hardware (PKCS#11) device.

`-n` *nametype*
Specifies the owner type of the key. The value of *nametype* must either be `ZONE` (for a `DNSSEC` zone key (`KEY/DNSKEY`)), `HOST` or `ENTITY` (for a key associated with a host (`KEY`)), `USER` (for a key associated with a user (`KEY`)), or `OTHER` (`DNSKEY`). These values are case-insensitive.

`-c` *class*
Indicates that the DNS record containing the key should have the specified class. If not specified, class `IN` is used.

`-f` *flag*
Set the specified flag in the flag field of the `KEY/DNSKEY` record. The only recognized flag is `KSK` (Key Signing Key) `DNSKEY`.

`-h`
Displays a short summary of the options and arguments to `dnssec-keyfromlabel`.

`-k`
Generate `KEY` records rather than `DNSKEY` records.

`-p` *protocol*
Sets the protocol value for the generated key. The protocol is a number between 0 and 255. The default is `3` (`DNSSEC`). Other possible values for this argument are listed in RFC 2535 and its successors.

`-t` *type*
Indicates the use of the key. *type* must be one of `AUTHCONF`, `NOAUTHCONF`, `NOAUTH`, or `NOCONF`. The default is `AUTHCONF`. `AUTH` refers to the ability to authenticate data, and `CONF` the ability to encrypt data.

-v *level*
Sets the debugging level.

Generated Key Files

When `dnssec-keyfromlabel` completes successfully, it displays a string of the form K*nnnn*.+*aaa*+*iiiii* to the standard output. This is an identification string for the key files it has generated, which translates as follows.

- *nnnn* is the key name.
- *aaa* is the numeric representation of the algorithm.
- *iiiii* is the key identifier (or footprint).

`dnssec-keyfromlabel` creates two files, with names based on the displayed string. K*nnnn*.+*aaa*+*iiiii*.`key` contains the public key, and K*nnnn*.+*aaa*+*iiiii*.`private` contains the private key.

The first file contains a `DNS KEY` record that can be inserted into a zone file (directly or with an `$INCLUDE` statement).

The second file contains algorithm-specific fields. For obvious security reasons, this file does not have general read permission.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also

`dnssec-keygen(1M)`, `dnssec-signzone(1M)`, `attributes(5)`

RFC 2539, *RFC 2845*, *RFC 4033*

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name dnssec-keygen – DNSSEC key generation tool

Synopsis

```
dnssec-keygen -a algorithm -b keysize -n nametype [-ehk]
     [-c class] [-f flag] [-g generator] [-p protocol]
     [-r randomdev] [-s strength] [-t type] [-v level] name
```

Description The `dnssec-keygen` utility generates keys for DNSSEC (Secure DNS), as defined in RFC 2535 and RFC 4034. It can also generate keys for use with TSIG (Transaction Signatures), as defined in RFC 2845.

Options The following options are supported:

`-a` *algorithm*
Select the cryptographic algorithm. The value of algorithm must be one of RSAMD5 (RSA) or RSASHA1, DSA, NSEC3RSASHA1, NSEC3DSA, `DH` (Diffie-Hellman), or HMAC-MD5. These values are case insensitive.

For DNSSEC, RSASHA1 is a mandatory-to-implement algorithm and DSA is recommended. For TSIG, HMAC-MD5 is mandatory.

Note – HMAC-MD5 and DH automatically set the `-k` flag.

`-b` *keysize*
Specify the number of bits in the key. The choice of key size depends on the algorithm used. RSAMD5 and RSASHA1 keys must be between 512 and 2048 bits. Diffie-Hellman keys must be between 128 and 4096 bits. DSA keys must be between 512 and 1024 bits and an exact multiple of 64. HMAC-MD5 keys must be between 1 and 512 bits.

`-c` *class*
Indicate that the DNS record containing the key should have the specified class. If not specified, class IN is used.

`-e`
Use a large exponent if generating an RSAMD5 or RSASHA1 key.

`-f` *flag*
Set the specified flag in the flag field of the KEY/DNSKEY record. The only recognized flag is KSK (Key Signing Key) DNSKEY.

`-g` *generator*
Use this *generator* if generating a Diffie Hellman key. Allowed values are 2 and 5. If no generator is specified, a known prime from RFC 2539 will be used if possible; otherwise the default is 2.

`-h`
Print a short summary of the options and arguments to `dnssec-keygen`.

`-k`
Generate KEY records rather than DNSKEY records.

`-n` *nametype*
Specify the owner type of the key. The value of *nametype* must either be `ZONE` (for a DNSSEC zone key (KEY/DNSKEY)), `HOST` or `ENTITY` (for a key associated with a host (KEY)), USER (for a key associated with a user(KEY)), or `OTHER` (DNSKEY). These values are case insensitive. Defaults to ZONE for DNSKEY generation.

`-p` *protocol*
Set the protocol value for the generated key. The *protocol* argument is a number between 0 and 255. The default is 3 (DNSSEC) Other possible values for this argument are listed in RFC 2535 and its successors.

`-r` *randomdev*
Specify the source of randomness. If the operating system does not provide a `/dev/random` or equivalent device, the default source of randomness is keyboard input. *randomdev* specifies the name of a character device or file containing random data to be used instead of the default. The special value "`keyboard`" indicates that keyboard input should be used.

`-s` *strength*
Specify the strength value of the key. The *strength* argument is a number between 0 and 15, and currently has no defined purpose in DNSSEC.

`-t` *type*
Indicate the use of the key. `type` must be one of `AUTHCONF`, `NOAUTHCONF`, `NOAUTH`, or `NOCONF`. The default is `AUTHCONF`. `AUTH` refers to the ability to authenticate data, and `CONF` the ability to encrypt data.

`-v` *level*
Set the debugging level.

Generated Keys When `dnssec-keygen` completes successfully, it prints a string of the form K*nnnn*`.+`*aaa*`+`*iiiii* to the standard output. This is an identification string for the key it has generated.

- *nnnn* is the key name.
- *aaa* is the numeric representation of the algorithm.
- *iiiii* is the key identifier (or footprint).

The `dnssec-keygen` utility creates two files, with names based on the printed string.

- K*nnnn*`.+`*aaa*`+`*iiiii*`.key` contains the public key.
- K*nnnn*`.+`*aaa*`+`*iiiii*`.private` contains the private key.

The `.key` file contains a DNS `KEY` record that can be inserted into a zone file (directly or with a `$INCLUDE` statement).

The `.private` file contains algorithm specific fields. For obvious security reasons, this file does not have general read permission.

Both `.key` and `.private` files are generated for symmetric encryption algorithm such as HMAC-MD5, even though the public and private key are equivalent.

Examples

EXAMPLE 1 Generating a 768-bit DSA Key

To generate a 768-bit DSA key for the domain `example.com`, the following command would be issued:

```
dnssec-keygen -a DSA -b 768 -n ZONE example.com
```

The command would print a string of the form:

```
Kexample.com.+003+26160
```

The following files would be created:

```
Kexample.com.+003+26160.key
Kexample.com.+003+26160.private
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also

`dnssec-signzone(1M)`, `attributes(5)`

RFC 2539, *RFC 2845*, *RFC 4033*

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name dnssec-signzone – DNSSEC zone signing tool

Synopsis

```
dnssec-signzone [-Aaghptz] [-c class] [-d directory]
     [-e end-time] [-f output-file] [-H iterations] [-I input_format]
     [-i interval] [-k key] [-l domain] [-N soa-serial-format] [-n ncpus]
     [-O output_format] [-o origin] [-r randomdev] [-s start-time]
     [-v level] [-3 salt] zonefile [key]...
```

Description The `dnssec-signzone` utility signs a zone. It generates `NSEC` and `RRSIG` records and produces a signed version of the zone. The security status of delegations from the signed zone (that is, whether the child zones are secure or not) is determined by the presence or absence of a `keyset` file for each child zone.

Options The following options are supported:

`-A`
When generating an NSEC3 chain, set the `OPTOUT` flag on all NSEC3 records and do not generate NSEC3 records for insecure delegations.

`-a`
Verify all generated signatures.

`-c` *class*
Specify the `DNS` class of the zone.

`-d` *directory*
Look for `keyset` files in *directory*.

`-e` *end-time*
Specify the date and time when the generated `RRSIG` records expire. As with `start-time`, an absolute time is indicated in `YYYYMMDDHHMMSS` notation. A time relative to the start time is indicated with +*N*, which is *N* seconds from the start time. A time relative to the current time is indicated with `now`+*N*. If no *end-time* is specified, 30 days from the start time is used as a default.

`-f` *output-file*
The name of the output file containing the signed zone. The default is to append `.signed` to the input file name.

`-g`
Generate DS records for child zones from `keyset` files. Existing DS records will be removed.

`-H` *iterations*
When generating a NSEC3 chain use the number of interations specified by *iterations*. The default is 100.

`-h`
Prints a short summary of the options and arguments to `dnssec-signzone()`.

`-I` *input-format*
The format of the input zone file. Possible formats are `text` (default) and `raw`. This option is primarily intended for dynamic signed zones so that the dumped zone file in a non-text format containing updates can be signed directly. The use of this option serves no purpose for non-dynamic zones.

`-i` *interval*
Specify the cycle interval as an offset from the current time (in seconds). When a previously signed zone is passed as input, records could be resigned. If an `RRSIG` record expires after the cycle interval, it is retained. Otherwise, it is considered to be expiring soon and will be replaced.

The default cycle interval is one quarter of the difference between the signature end and start times. If neither *end-time* or *start-time* are specified, `dnssec-signzone` generates signatures that are valid for 30 days, with a cycle interval of 7.5 days. Any existing `RRSIG` records due to expire in less than 7.5 days would be replaced.

`-j` *jitter*
When signing a zone with a fixed signature lifetime, all `RRSIG` records issued at the time of signing expire simultaneously. If the zone is incrementally signed, that is, a previously-signed zone is passed as input to the signer, all expired signatures have to be regenerated at about the same time. The jitter option specifies a jitter window that will be used to randomize the signature-expire time, thus spreading incremental signature regeneration over time.

Signature lifetime jitter also benefits, to some extent, validators and servers by spreading out cache expiration. That is, if large numbers of `RRSIG`s from all caches do not expire at the same time, there will be less congestion than if all validators needed to refetch at almost the same time.

`-k` *key*
Treat specified *key* as a key-signing key, ignoring any key flags. This option can be specified multiple times.

`-l` *domain*
Generate a DLV set in addition to the key (DNSKEY) and DS sets. The domain is appended to the name of the records.

`-N` *soa-serial-format*
The SOA serial number format of the signed zone. Possible formats are `keep` (default), `increment` and `unixtime`, described as follows.

`keep`
Do not modify the SOA serial number.

`increment`
Increment the SOA serial number using RFC 1982 arithmetic.

`unixtime`
Set the SOA serial number to the number of seconds since epoch.

`-n` *nthreads*
Specifies the number of threads to use. By default, one thread is started for each detected CPU.

`-O` *output_format*
The format of the output file containing the signed zone. Possible formats are `text` (default) and `raw`.

`-o` *origin*
Specify the zone origin. If not specified, the name of the zone file is assumed to be the origin.

`-p`
Use pseudo-random data when signing the zone. This is faster, but less secure, than using real random data. This option may be useful when signing large zones or when the entropy source is limited.

`-r` *randomdev*
Specifies the source of randomness. If the operating system does not provide a `/dev/random` or equivalent device, the default source of randomness is keyboard input. *randomdev* specifies the name of a character device or file containing random data to be used instead of the default `/dev/random`. The special value `keyboard` indicates that keyboard input should be used.

`-s` *start-time*
Specify the date and time when the generated `RRSIG` records become valid. This can be either an absolute or relative time. An absolute start time is indicated by a number in *YYYYMMDDHHMMSS* notation; 20000530144500 denotes 14:45:00 UTC on May 30th, 2000. A relative start time is indicated by *+N*, which is *N* seconds from the current time. If no *start-time* is specified, the current time minus one hour (to allow for clock skew) is used.

`-t`
Print statistics at completion.

`-v` *level*
Set the debugging level.

`-z`
Ignore KSK flag on key when determining what to sign.

`-3` *salt*
Generate a NSEC3 chain with the specified hex-encoded *salt*. A dash (`-`) can be used to indicate that no salt is to be used when generating the NSEC3 chain.

Operands The following operands are supported:

zonefile
The file containing the zone to be signed.

key
Specify which keys should be used to sign the zone. If no keys are specified, then the zone will be examined for `DNSKEY` records at the zone apex. If these are found and there are matching private keys in the current directory, these will be used for signing.

Examples **EXAMPLE 1** Signing a Zone with a DSA Key

The following command signs the `example.com` zone with the DSA key generated in the example in the `dnssec-keygen(1M)` manual page (`Kexample.com.+003+17247`). The zone's keys must be in the master file (`db.example.com`). This invocation looks for keyset files in the current directory, so that DS records can be generated from them (`-g`).

```
% dnssec-signzone -g -o example.com db.example.com \
Kexample.com.+003+17247
db.example.com.signed
%
```

In the above example, `dnssec-signzone` creates the file `db.example.com.signed`. This file should be referenced in a zone statement in a `named.conf` file.

EXAMPLE 2 Re-signing a Previously Signed Zone

The following commands re-sign a previously signed zone with default parameters. The private keys are assumed to be in the current directory.

```
% cp db.example.com.signed db.example.com
% dnssec-signzone -o example.com db.example.com \
db.example.com.signed
%
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also `dnssec-keygen(1M)`, `attributes(5)`

RFC 4033

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name domainname – set or display name of the current domain

Synopsis

```
domainname [name-of-domain]
```

Description Without an argument, `domainname` displays the name of the current domain name used in RPC exchanges, usually referred to as the NIS or NIS+ domain name. This name typically encompasses a group of `hosts` or `passwd` entries under the same administration. The `domainname` command is used by various components of Solaris to resolve names for entries such as are found in `passwd`, `hosts` and `aliases`. By default, naming services such as NIS and NIS+ use `domainname` to resolve names.

With appropriate privileges (root or an equivalent role [see `rbac(5)`]), you can set the name of the domain by specifying the name as an argument to the `domainname` command.

The domain name for various naming services can also be set by other means. For example, `ypinit` can be used to specify a different domain for all NIS calls. The domain name of the machine is usually set during boot time through the `domainname` command by the `svc:/system/identity:domain` service. If the new domain name is not saved in the `/etc/defaultdomain` file, the machine reverts to the old domain after it reboots.

The `sendmail(1M)` daemon, as shipped with Solaris, and the `sendmail` implementation provided by `sendmail.org` (formerly referred to as "Berkeley 8.x sendmail") both attempt to determine a local host's fully qualified host name at startup and both pursue follow-up actions if the initial search fails. It is in these follow-up actions that the two implementations differ.

Both implementations use a standard Solaris or Unix system call to determine its fully qualified host name at startup, following the name service priorities specified in `nsswitch.conf(4)`. To this point, the Solaris and `sendmail.org` versions behave identically.

If the request for a fully qualified host name fails, the `sendmail.org` `sendmail` sleeps for 60 seconds, tries again, and, upon continuing failure, resorts to a short name. The Solaris version of `sendmail` makes the same initial request, but then, following initial failure, calls `domainname`. If successful, the sleep is avoided.

On a Solaris machine, if you run the `sendmail.org` version of sendmail, you get the startup behavior (omitting the `domainname` call) described above. If you run the Solaris `sendmail`, the `domainname` call is made if needed.

If the Solaris `sendmail` cannot determine the fully qualified host name, use `check-hostname(1M)` as a troubleshooting aid. This script can offer guidance as to appropriate corrective action.

Files `/etc/defaultdomain`

`/etc/nsswitch.conf`

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also NIS+(1), nischown(1), nispasswd(1), svcs(1), check-hostname(1M), hostconfig(1M), named(1M), nisaddcred(1M), sendmail(1M), svcadm(1M), ypinit(1M), sys-unconfig(1M), aliases(4), defaultdomain(4), hosts(4), nsswitch.conf(4), passwd(4), attributes(5), rbac(5), smf(5)

Notes The domainname service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/identity:domain
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name drd – Logical Domain Dynamic Reconfiguration daemon

Synopsis

```
/usr/lib/ldoms/drd
```

Description The `drd` daemon is part of the framework that enables the addition and removal of resources from a Logical Domain. This framework is collectively called Dynamic Reconfiguration (DR).

`drd` is responsible for various aspects of DR on a Logical Domain and must be enabled to ensure proper DR functionality. It is started at boot time and has no configuration options.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWldomu
Interface Stability	Unstable

See Also `svcs(1)`, `svcadm(1M)`, `syslog(3C)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`

Errors `drd` uses `syslog(3C)` to report status and error messages. All of the messages are logged with the `LOG_DAEMON` facility. Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities, and informational messages are logged with the `LOG_INFO` priority. The default entries in the `/etc/syslog.conf` file log all the `drd` error messages to the `/var/adm/messages` log.

Notes The `drd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/platform/sun4v/drd:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name drvconfig – apply permission and ownership changes to devices

Synopsis

```
drvconfig [-bn] [-a alias_name] [-c class_name]
     [-i drivername] [-m major_num] [-r root_dir]
```

Description devfsadm(1M) is now the preferred command and should be used instead of drvconfig.

The default operation of drvconfig is to apply permission and ownership changes to devices. Normally, this command is run automatically after a new driver has been installed (with add_drv(1M)) and the system has been rebooted.

Options The following options are supported:

-a*alias_name*	Add the name *alias_name* to the list of aliases that this driver is known by. This option, if used, must be used with the -m *major_num*, the -b and the -i *drivername* options.
-b	Add a new major number to name binding into the kernel's internal name_to_major tables. This option is not normally used directly, but is used by other utilities such as add_drv(1M). Use of the -b option requires that -i and -m be used also. No /devices entries are created.
-c*class_name*	The driver being added to the system exports the class *class_name*. This option is not normally used directly, but is used by other utilities. It is only effective when used with the -b option.
-i*drivername*	Only configure the devices for the named driver. The following options are used by the implementation of add_drv(1M) and rem_drv(1M), and may not be supported in future versions of Solaris:
-m*major_num*	Specify the major number *major_num* for this driver to add to the kernel's name_to_major binding tables.
-n	Do not try to load and attach any drivers, or if the -i option is given, do not try to attach the driver named *drivername*.
-r*root_dir*	Perform operations under *root_dir*, rather than directly under root.

Exit Status

0	Successful completion.
non-zero	An error occurred.

Files

/devices	Device nodes directory
/etc/minor_perm	Minor mode permissions
/etc/name_to_major	Major number binding
/etc/driver_classes	Driver class binding file

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also sh(1), add_drv(1M), modinfo(1M), modload(1M), modunload(1M), rem_drv(1M), update_drv(1M), path_to_inst(4), attributes(5), devfs(7FS)

Name dsvclockd – DHCP service lock daemon

Synopsis

```
/usr/lib/inet/dsvclockd [-d 1 | 2] [-f ] [-v]
```

Description The `dsvclockd` daemon is a lock manager that works in conjunction with the Dynamic Host Configuration Protocol (DHCP) Data Service Library (`libdhcpsvc`). It provides shared or exclusive access to the `dhcp_network(4)` and `dhcptab(4)` tables. This service is used by the `SUNWbinfiles` and `SUNWfiles` DHCP data store modules. See `dhcp_modules(5)`.

`dsvclockd` is started on demand by `libdhcpsvc`. The `dsvclockd` daemon should be started manually only if command line options need to be specified.

Options The following options are supported:

`-d 1 | 2` Set debug level. Two levels of debugging are currently available, `1` and `2`. Level `2` is more verbose.

`-f` Run in the foreground instead of as a daemon process. When this option is used, messages are sent to standard error instead of to `syslog(3C)`.

`-v` Provide verbose output.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdhcsu
Interface Stability	Unstable

See Also `syslog(3C)`, `dhcp_network(4)`, `dhcptab(4)`, `dhcp_modules(5)`, `attributes(5)`

Name dtrace – DTrace dynamic tracing compiler and tracing utility

Synopsis

```
dtrace [-32 | -64] [-aACeFGHhlqSvVwZ] [-b bufsz] [-c cmd]
     [-D name [=value]] [-I path] [-L path] [-o output]
     [-s script] [-U name] [-x arg [=val]]
     [-X a | c | s | t] [-p pid]
     [-P provider [[predicate] action]]
     [-m [provider:] module [[predicate] action]]
     [-f [[provider:] module:] function [[predicate] action]]
     [-n [[[provider:] module:] function:] name [[predicate] action]]
     [-i probe-id [[predicate] action]]
```

Description DTrace is a comprehensive dynamic tracing framework for the Solaris Operating System. DTrace provides a powerful infrastructure that permits administrators, developers, and service personnel to concisely answer arbitrary questions about the behavior of the operating system and user programs.

The *Solaris Dynamic Tracing Guide* describes how to use DTrace to observe, debug, and tune system behavior. Refer to this book for a detailed description of DTrace features, including the bundled DTrace observability tools, instrumentation providers, and the D programming language.

The `dtrace` command provides a generic interface to the essential services provided by the DTrace facility, including:

- Options that list the set of probes and providers currently published by DTrace
- Options that enable probes directly using any of the probe description specifiers (provider, module, function, name)
- Options that run the D compiler and compile one or more D program files or programs written directly on the command line
- Options that generate anonymous tracing programs
- Options that generate program stability reports
- Options that modify DTrace tracing and buffering behavior and enable additional D compiler features

You can use `dtrace` to create D scripts by using it in a `#!` declaration to create an interpreter file. You can also use `dtrace` to attempt to compile D programs and determine their properties without actually enabling tracing using the `-e` option. See `OPTIONS`. See the *Solaris Dynamic Tracing Guide* for detailed examples of how to use the `dtrace` utility to perform these tasks.

Options The arguments accepted by the `-P`, `-m`, `-f`, `-n`, and `-i` options can include an optional D language *predicate* enclosed in slashes `//` and optional D language *action* statement list enclosed in braces `{}`. D program code specified on the command line must be appropriately quoted to avoid interpretation of meta-characters by the shell.

The following options are supported:

`-32 | -64`
The D compiler produces programs using the native data model of the operating system kernel. You can use the `isainfo -b` command to determine the current operating system data model. If the `-32` option is specified, `dtrace` forces the D compiler to compile a D program using the 32-bit data model. If the `-64` option is specified, `dtrace` forces the D compiler to compile a D program using the 64-bit data model. These options are typically not required as `dtrace` selects the native data model as the default. The data model affects the sizes of integer types and other language properties. D programs compiled for either data model can be executed on both 32-bit and 64-bit kernels. The `-32` and `-64` options also determine the ELF file format (ELF32 or ELF64) produced by the `-G` option.

`-a`
Claim anonymous tracing state and display the traced data. You can combine the `-a` option with the `-e` option to force `dtrace` to exit immediately after consuming the anonymous tracing state rather than continuing to wait for new data. See the *Solaris Dynamic Tracing Guide* for more information about anonymous tracing.

`-A`
Generate `driver.conf(4)` directives for anonymous tracing. This option constructs a set of `dtrace(7D)` configuration file directives to enable the specified probes for anonymous tracing and then exits. By default, `dtrace` attempts to store the directives to the file `/kernel/drv/dtrace.conf`. You can modify this behavior if you use the `-o` option to specify an alternate output file.

`-b` *bufsz*
Set principal trace buffer size (*bufsz*). The trace buffer size can include any of the size suffixes `k`, `m`, `g`, or `t`. If the buffer space cannot be allocated, `dtrace` attempts to reduce the buffer size or exit depending on the setting of the `bufresize` property.

`-c` *cmd*
Run the specified command *cmd* and exit upon its completion. If more than one `-c` option is present on the command line, `dtrace` exits when all commands have exited, reporting the exit status for each child process as it terminates. The process-ID of the first command is made available to any D programs specified on the command line or using the `-s` option through the `$target` macro variable. Refer to the *Solaris Dynamic Tracing Guide* for more information on macro variables.

`-C`
Run the C preprocessor `cpp(1)` over D programs before compiling them. You can pass options to the C preprocessor using the `-D`, `-U`, `-I`, and `-H` options. You can select the degree of C standard conformance if you use the `-X` option. For a description of the set of tokens defined by the D compiler when invoking the C preprocessor, see `-X`.

`-D` *name* `[=`*value*`]`
Define *name* when invoking `cpp(1)` (enabled using the `-C` option). If you specify the equals sign (`=`) and additional *value*, the name is assigned the corresponding value. This option passes the `-D` option to each `cpp` invocation.

`-e`
Exit after compiling any requests and consuming anonymous tracing state (`-a` option) but prior to enabling any probes. You can combine this option with the `-a` option to print anonymous tracing data and exit. You can also combine this option with D compiler options. This combination verifies that the programs compile without actually executing them and enabling the corresponding instrumentation.

`-f`[[*provider*`:`]*module*`:`]*function*[[*predicate*]*action*]]
Specify function name to trace or list (`-l` option). The corresponding argument can include any of the probe description forms *provider:module:function*, *module:function*, or *function*. Unspecified probe description fields are left blank and match any probes regardless of the values in those fields. If no qualifiers other than *function* are specified in the description, all probes with the corresponding *function* are matched. The `-f` argument can be suffixed with an optional D probe clause. You can specify more than one `-f` option on the command line at a time.

`-F`
Coalesce trace output by identifying function entry and return. Function entry probe reports are indented and their output is prefixed with `->`. Function return probe reports are unindented and their output is prefixed with `<-`. System call entry probe reports are indented and their output is prefixed with `=>`. System call return probe reports are unindented and their output is prefixed with `<=`.

`-G`
Generate an ELF file containing an embedded DTrace program. The DTrace probes specified in the program are saved inside of a relocatable ELF object which can be linked into another program. If the `-o` option is present, the ELF file is saved using the pathname specified as the argument for this operand. If the `-o` option is not present and the DTrace program is contained with a file whose name is *filename*`.d`, then the ELF file is saved using the name *filename*`.o`. Otherwise the ELF file is saved using the name `d.out`.

`-H`
Print the pathnames of included files when invoking cpp(1) (enabled using the `-C` option). This option passes the `-H` option to each `cpp` invocation, causing it to display the list of pathnames, one for each line, to `stderr`.

`-h`
Generate a header file containing macros that correspond to probes in the specified provider definitions. This option should be used to generate a header file that is included by other source files for later use with the `-G` option. If the `-o` option is present, the header file is saved using the pathname specified as the argument for that option. If the `-o` option is not present and the DTrace program is contained with a file whose name is *filename*`.d`, then the header file is saved using the name *filename*`.h`.

`-i` *probe-id*[[*predicate*] *action*]
Specify probe identifier (*probe-id*) to trace or list (`-l` option). You can specify probe IDs using decimal integers as shown by `dtrace -l`. The `-i` argument can be suffixed with an optional D probe clause. You can specify more than one `-i` option at a time.

`-I` *path*
Add the specified directory *path* to the search path for `#include` files when invoking cpp(1) (enabled using the `-C` option). This option passes the `-I` option to each `cpp` invocation. The specified *path* is inserted into the search path ahead of the default directory list.

`-L` *path*
Add the specified directory *path* to the search path for DTrace libraries. DTrace libraries are used to contain common definitions that can be used when writing D programs. The specified *path* is added after the default library search path.

`-l`
List probes instead of enabling them. If the `-l` option is specified, `dtrace` produces a report of the probes matching the descriptions given using the `-P`, `-m`, `-f`, `-n`, `-i`, and `-s` options. If none of these options are specified, this option lists all probes.

`-m` [[*provider:*] *module:* [[*predicate*] *action*]]
Specify module name to trace or list (`-l` option). The corresponding argument can include any of the probe description forms *provider:module* or *module*. Unspecified probe description fields are left blank and match any probes regardless of the values in those fields. If no qualifiers other than *module* are specified in the description, all probes with a corresponding *module* are matched. The `-m` argument can be suffixed with an optional D probe clause. More than one `-m` option can be specified on the command line at a time.

`-n` [[[*provider:*] *module:*] *function:*] *name* [[*predicate*] *action*]
Specify probe name to trace or list (`-l` option). The corresponding argument can include any of the probe description forms *provider:module:function:name*, *module:function:name*, *function:name*, or *name*. Unspecified probe description fields are left blank and match any probes regardless of the values in those fields. If no qualifiers other than *name* are specified in the description, all probes with a corresponding *name* are matched. The `-n` argument can be suffixed with an optional D probe clause. More than one `-n` option can be specified on the command line at a time.

`-o` *output*
Specify the *output* file for the `-A`, `-G`, `-h`, and `-l` options, or for the traced data itself. If the `-A` option is present and `-o` is not present, the default output file is `/kernel/drv/dtrace.conf`. If the `-G` option is present and the `-s` option's argument is of the form *filename*`.d` and `-o` is not present, the default output file is *filename*`.o`. Otherwise the default output file is `d.out`.

`-p` *pid*
Grab the specified process-ID *pid*, cache its symbol tables, and exit upon its completion. If more than one `-p` option is present on the command line, `dtrace` exits when all commands have exited, reporting the exit status for each process as it terminates. The first process-ID is made available to any D programs specified on the command line or using the `-s` option through the `$target` macro variable. Refer to the *Solaris Dynamic Tracing Guide* for more information on macro variables.

`-P` *provider* [[*predicate*] *action*]
Specify provider name to trace or list (`-l` option). The remaining probe description fields module, function, and name are left blank and match any probes regardless of the values in those fields. The `-P` argument can be suffixed with an optional D probe clause. You can specify more than one `-P` option on the command line at a time.

`-q`
Set quiet mode. `dtrace` suppresses messages such as the number of probes matched by the specified options and D programs and does not print column headers, the CPU ID, the probe ID, or insert newlines into the output. Only data traced and formatted by D program statements such as `trace()` and `printf()` is displayed to `stdout`.

`-s`
Compile the specified D program source file. If the `-e` option is present, the program is compiled but instrumentation is not enabled. If the `-l` option is present, the program is compiled and the set of probes matched by it is listed, but instrumentation is not enabled. If none of `-e`, `-l`, `-G`, or `-A` are present, the instrumentation specified by the D program is enabled and tracing begins.

`-S`
Show D compiler intermediate code. The D compiler produces a report of the intermediate code generated for each D program to `stderr`.

`-U` *name*
Undefine the specified *name* when invoking cpp(1) (enabled using the `-C` option). This option passes the `-U` option to each `cpp` invocation.

`-v`
Set verbose mode. If the `-v` option is specified, `dtrace` produces a program stability report showing the minimum interface stability and dependency level for the specified D programs. DTrace stability levels are explained in further detail in the *Solaris Dynamic Tracing Guide*.

`-V`
Report the highest D programming interface version supported by `dtrace`. The version information is printed to `stdout` and the `dtrace` command exits. Refer to the *Solaris Dynamic Tracing Guide* for more information about DTrace versioning features.

`-w`
Permit destructive actions in D programs specified using the `-s`, `-P`, `-m`, `-f`, `-n`, or `-i` options. If the `-w` option is not specified, `dtrace` does not permit the compilation or enabling of a D program that contains destructive actions.

`-x` *arg* [=*val*]
Enable or modify a DTrace runtime option or D compiler option. The list of options is found in the *Solaris Dynamic Tracing Guide*. Boolean options are enabled by specifying their name. Options with values are set by separating the option name and value with an equals sign (=).

`-X a | c | s | t`

Specify the degree of conformance to the ISO C standard that should be selected when invoking cpp(1) (enabled using the `-C` option). The `-X` option argument affects the value and presence of the `__STDC__` macro depending upon the value of the argument letter.

The `-X` option supports the following arguments:

`a` Default. ISO C plus K&R compatibility extensions, with semantic changes required by ISO C. This is the default mode if `-X` is not specified. The predefined macro `__STDC__` has a value of 0 when `cpp` is invoked in conjunction with the `-Xa` option.

`c` Conformance. Strictly conformant ISO C, without K&R C compatibility extensions. The predefined macro `__STDC__` has a value of 1 when `cpp` is invoked in conjunction with the `-Xc` option.

`s` K&R C only. The macro `__STDC__` is not defined when `cpp` is invoked in conjunction with the `-Xs` option.

`t` Transition. ISO C plus K&R C compatibility extensions, without semantic changes required by ISO C. The predefined macro `__STDC__` has a value of 0 when `cpp` is invoked in conjunction with the `-Xt` option.

As the `-X` option only affects how the D compiler invokes the C preprocessor, the `-Xa` and `-Xt` options are equivalent from the perspective of D and both are provided only to ease re-use of settings from a C build environment.

Regardless of the `-X` mode, the following additional C preprocessor definitions are always specified and valid in all modes:

- `__sun`
- `__unix`
- `__SVR4`
- `__sparc` (on SPARC systems only)
- `__sparcv9` (on SPARC systems only when 64-bit programs are compiled)
- `__i386` (on x86 systems only when 32-bit programs are compiled)
- `__amd64` (on x86 systems only when 64-bit programs are compiled)
- `__`*‘uname -s‘*`_`*‘uname -r‘* (for example, `__SunOS_5_10`)
- `__SUNW_D=1`
- `__SUNW_D_VERSION=0x`*MMmmmuuu*

  Where *MM* is the major release value in hexadecimal, *mmm* is the minor release value in hexadecimal, and *uuu* is the micro release value in hexadecimal. Refer to the *Solaris Dynamic Tracing Guide* for more information about DTrace versioning.

`-Z`

Permit probe descriptions that match zero probes. If the `-Z` option is not specified, `dtrace` reports an error and exits if any probe descriptions specified in D program files (`-s` option) or on the command line (`-P`, `-m`, `-f`, `-n`, or `-i` options) contain descriptions that do not match any known probes.

Operands You can specify zero or more additional arguments on the `dtrace` command line to define a set of macro variables (`$1`, `$2`, and so forth). The additional arguments can be used in D programs specified using the `-s` option or on the command line. The use of macro variables is described further in the *Solaris Dynamic Tracing Guide*.

Exit Status The following exit values are returned:

0 Successful completion.

For D program requests, an exit status of `0` indicates that programs were successfully compiled, probes were successfully enabled, or anonymous state was successfully retrieved. `dtrace` returns `0` even if the specified tracing requests encountered errors or drops.

1 An error occurred.

For D program requests, an exit status of `1` indicates that program compilation failed or that the specified request could not be satisfied.

2 Invalid command line options or arguments were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdtrc
Interface Stability	See below.

The command-line syntax is Committed. The human-readable output is Uncommitted.

See Also `cpp(1)`, `isainfo(1)`, `ssh(1)`, `libdtrace(3LIB)`, `driver.conf(4)`, `attributes(5)`, `dtrace(7D)`

Solaris Dynamic Tracing Guide

Usage When using the `-p` flag, `dtrace` stops the target processes while it is inspecting them and reporting results. A process can do nothing while it is stopped. This means that, if , for example, the X server is inspected by `dtrace` running in a window under the X server's control, the whole window system can become deadlocked, because the `proc` tool would be attempting to display its results to a window that cannot be refreshed. In such a case, logging in from another system using `ssh(1)` and killing the offending `proc` tool clears the deadlock.

Name dumpadm – configure operating system crash dump

Synopsis

```
/usr/sbin/dumpadm [-nuy] [-c content-type] [-d dump-device]
     [-m mink | minm | min%] [-s savecore-dir]
     [-r root-dir] [-z on | off]
```

Description The `dumpadm` program is an administrative command that manages the configuration of the operating system crash dump facility. A crash dump is a disk copy of the physical memory of the computer at the time of a fatal system error. When a fatal operating system error occurs, a message describing the error is printed to the console. The operating system then generates a crash dump by writing the contents of physical memory to a predetermined dump device, which is typically a local disk partition. The dump device can be configured by way of `dumpadm`. Once the crash dump has been written to the dump device, the system will reboot.

Fatal operating system errors can be caused by bugs in the operating system, its associated device drivers and loadable modules, or by faulty hardware. Whatever the cause, the crash dump itself provides invaluable information to your support engineer to aid in diagnosing the problem. As such, it is vital that the crash dump be retrieved and given to your support provider. Following an operating system crash, the `savecore(1M)` utility is executed automatically during boot to retrieve the crash dump from the dump device and write it to your file system in compressed form, to a file name `vmdump.`*X*, where *X* is an integer identifying the dump. Afterwards, `savecore(1M)` can be invoked on the same or another system to expand the compressed crash dump to a pair of files named `unix.`*X* and `vmcore.`*X*. The directory in which the crash dump is saved on reboot can be configured using `dumpadm`.

For systems with a UFS root file system, the default dump device is configured to be an appropriate swap partition. Swap partitions are disk partitions reserved as virtual memory backing store for the operating system. Thus, no permanent information resides in swap to be overwritten by the dump. See `swap(1M)`. For systems with a ZFS root file system, dedicated ZFS volumes are used for swap and dump areas. For further information about setting up a dump area with ZFS, see the *ZFS Administration Guide*. To view the current dump configuration, use the `dumpadm` command with no arguments:

```
example# dumpadm

      Dump content: kernel pages
       Dump device: /dev/dsk/c0t0d0s1 (swap)
Savecore directory: /var/crash/saturn
  Savecore enabled: yes
   Save compressed: yes
```

When no options are specified, `dumpadm` displays the current crash dump configuration. The example above shows the set of default values: the dump content is set to kernel memory pages only, the dump device is a swap disk partition, the directory for `savecore` files is set to `/var/crash/`*hostname*. `savecore` is set to run automatically on reboot and save the crash dump in a compressed format.

When one or more options are specified, `dumpadm` verifies that your changes are valid, and if so, reconfigures the crash dump parameters and displays the resulting configuration. You must be `root` to view or change dump parameters.

Options The following options are supported:

`-c` *content-type*
Modify the dump configuration so that the crash dump consists of the specified dump content. The content should be one of the following:

`kernel`
Kernel memory pages only.

`all`
All memory pages.

`curproc`
Kernel memory pages, and the memory pages of the process whose thread was currently executing on the CPU on which the crash dump was initiated. If the thread executing on that CPU is a kernel thread not associated with any user process, only kernel pages will be dumped.

`-d` *dump-device*
Modify the dump configuration to use the specified dump device. The dump device may one of the following:

dump-device
A specific dump device specified as an absolute pathname, such as `/dev/dsk/`*cNtNdNsN* when the system is running a UFS root file system. Or, specify a ZFS volume, such as `/dev/zvol/dsk/rpool/dump`, when the system is running a ZFS root file system.

`swap`
If the special token `swap` is specified as the dump device, `dumpadm` examines the active swap entries and selects the most appropriate entry to configure as the dump device. See `swap(1M)`. Refer to the `NOTES` below for details of the algorithm used to select an appropriate swap entry. When the system is first installed with a UFS root file system, `dumpadm` uses the value for `swap` to determine the initial dump device setting. A given ZFS volume cannot be configured for both the swap area and the dump device.

`-m` *min*`k` | *min*`m` | *min*`%`
Create a `minfree` file in the current savecore directory indicating that `savecore` should maintain at least the specified amount of free space in the file system where the savecore directory is located. The `min` argument can be one of the following:

`k`
A positive integer suffixed with the unit `k` specifying kilobytes.

`m`
A positive integer suffixed with the unit `m` specifying megabytes.

`%`
A % symbol, indicating that the `minfree` value should be computed as the specified percentage of the total current size of the file system containing the savecore directory.

The `savecore` command will consult the `minfree` file, if present, prior to writing the dump files. If the size of these files would decrease the amount of free disk space below the `minfree` threshold, no dump files are written and an error message is logged. The administrator should immediately clean up the savecore directory to provide adequate free space, and re-execute the `savecore` command manually. The administrator can also specify an alternate directory on the `savecore` command-line.

`-n`
Modify the dump configuration to not run `savecore` automatically on reboot. This is not the recommended system configuration; if the dump device is a swap partition, the dump data will be overwritten as the system begins to swap. If `savecore` is not executed shortly after boot, crash dump retrieval may not be possible.

`-r` *root-dir*
Specify an alternate root directory relative to which `dumpadm` should create files. If no `-r` argument is specified, the default root directory `/` is used.

`-s` *savecore-dir*
Modify the dump configuration to use the specified directory to save files written by `savecore`. The directory should be an absolute path and exist on the system. If upon reboot the directory does not exist, it will be created prior to the execution of `savecore`. See the `NOTES` section below for a discussion of security issues relating to access to the savecore directory. The default savecore directory is `/var/crash/`*hostname* where *hostname* is the output of the `-n` option to the `uname(1)` command.

`-u`
Forcibly update the kernel dump configuration based on the contents of `/etc/dumpadm.conf`. Normally this option is used only on reboot when starting `svc:/system/dumpadm:default`, when the `dumpadm` settings from the previous boot must be restored. Your dump configuration is saved in the configuration file for this purpose. If the configuration file is missing or contains invalid values for any dump properties, the default values are substituted. Following the update, the configuration file is resynchronized with the kernel dump configuration.

`-y`
Modify the dump configuration to automatically run `savecore` on reboot. This is the default for this dump setting.

`-z on | off`
Modify the dump configuration to control the operation of `savecore` on reboot. The options are `on`, to enable saving core files in a compressed format, and `off`, to automatically uncompress the crash dump file. The default is `on`, because crash dump files can be very large and require less file system space if saved in a compressed format.

Examples

EXAMPLE 1 Reconfiguring The Dump Device To A Dedicated Dump Device:

The following command reconfigures the dump device to a dedicated dump device:

```
example# dumpadm –d /dev/dsk/c0t2d0s2

          Dump content: kernel pages
           Dump device: /dev/dsk/c0t2d0s2 (dedicated)
    Savecore directory: /var/crash/saturn
      Savecore enabled: yes
       Save compressed: yes
```

Exit Status

The following exit values are returned:

`0`
Dump configuration is valid and the specified modifications, if any, were made successfully.

`1`
A fatal error occurred in either obtaining or modifying the dump configuration.

`2`
Invalid command line options were specified.

Files

`/dev/dump`
Dump device.

`/etc/dumpadm.conf`
Contains configuration parameters for `dumpadm`. Modifiable only through that command.

savecore-directory`/minfree`
Contains minimum amount of free space for *savecore-directory*. See `savecore(1M)`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs

See Also

`svcs(1)`, `uname(1)`, `savecore(1M)`, `svcadm(1M)`, `swap(1M)`, `attributes(5)`, `smf(5)`

Notes

The system crash dump service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/dumpadm:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Dump Device Selection

When the special `swap` token is specified as the argument to `dumpadm -d` the utility will attempt to configure the most appropriate swap device as the dump device. `dumpadm` configures the largest swap block device as the dump device; if no block devices are available for swap, the largest swap entry is configured as the dump device. If no swap entries are present, or none can be configured as the dump device, a warning message will be displayed. While local and remote swap files can be configured as the dump device, this is not recommended.

Dump Device/Swap Device Interaction (UFS File Systems Only)

In the event that the dump device is also a swap device, and the swap device is deleted by the administrator using the `swap -d` command, the `swap` command will automatically invoke `dumpadm -d swap` in order to attempt to configure another appropriate swap device as the dump device. If no swap devices remain or none can be configured as the dump device, the crash dump will be disabled and a warning message will be displayed. Similarly, if the crash dump is disabled and the administrator adds a new swap device using the `swap -a` command, `dumpadm -d swap` will be invoked to re-enable the crash dump using the new swap device.

Once `dumpadm -d swap` has been issued, the new dump device is stored in the configuration file for subsequent reboots. If a larger or more appropriate swap device is added by the administrator, the dump device is not changed; the administrator must re-execute `dumpadm -d swap` to reselect the most appropriate device fom the new list of swap devices.

Minimum Free Space

If the `dumpadm -m` option is used to create a `minfree` file based on a percentage of the total size of the file system containing the `savecore` directory, this value is not automatically recomputed if the file system subsequently changes size. In this case, the administrator must re-execute `dumpadm -m` to recompute the `minfree` value. If no such file exists in the `savecore` directory, `savecore` will default to a free space threshold of one megabyte. If no free space threshold is desired, a minfree file containing size 0 can be created.

Security Issues

If, upon reboot, the specified `savecore` directory is not present, it will be created prior to the execution of `savecore` with permissions 0700 (read, write, execute by owner only) and owner `root`. It is recommended that alternate savecore directories also be created with similar permissions, as the operating system crash dump files themselves may contain secure information.

Name editmap – query and edit single records in database maps for sendmail

Synopsis

```
editmap -C file [-N] [-f] [-q | -u | -x] maptype mapname key
     ["value"]...
```

Description The `editmap` command queries or edits one record in a database maps used by the keyed map lookups in `sendmail(1M)`. Arguments are passed on the command line and output (for queries) is directed to standard output.

Depending on how it is compiled, `editmap` handles up to three different database formats, selected using the *maptype* parameter. See `OPERANDS`.

If the `TrustedUser` option is set in the `sendmail` configuration file and `editmap` is invoked as root, the generated files are owned by the specified `TrustedUser`.

Options The following options are supported:

`-C` *file* Use the specified `sendmail` configuration file (*file*) to look up the `TrustedUser` option.

`-f` Disable the folding of all upper case letters in the key to lower case. Normally, all upper case letters in the key are folded to upper case. This is intended to mesh with the `-f` flag in the `K` line in `sendmail.cf`. The value is never case folded.

`-N` Include the null byte that terminates strings in the map (for alias maps).

`-q` Query the map for the specified key. If found, print value to standard output and exit with `0`. If not found then print an error message to `stdout` and exit with `EX_UNAVAILABLE`.

`-u` Update the record for key with value or inserts a new record if one doesn't exist. Exits with `0` on success or `EX_IOERR` on failure.

`-x` Delete the specific key from the map. Exit with `0` on success or `EX_IOERR` on failure.

Operands The following operands are supported:

key The left hand side of a record.

Each record is of the form:

key value

key and *value* are separated by white space.

mapname File name of the database map being created.

maptype Specifies the database format. The following *maptype* parameters are available:

`dbm` Specifies DBM format maps.

`btree` Specifies B-Tree format maps.

`hash` Specifies hash format maps.

value The right hand side of a record.

Each record is of the form:

key value

key and *value* are separated by white space.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsndmu

See Also `makemap(1M)`, `sendmail(1M)`, `attributes(5)`

Name edquota – edit user quotas for ufs file system

Synopsis

```
edquota [-p proto_user] username...

edquota -t
```

Description `edquota` is a quota editor. One or more users may be specified on the command line. For each user a temporary file is created with an ASCII representation of the current disk quotas for that user for each mounted ufs file system that has a `quotas` file, and an editor is then invoked on the file. The quotas may then be modified, new quotas added, etc. Upon leaving the editor, `edquota` reads the temporary file and modifies the binary quota files to reflect the changes made.

The editor invoked is `vi(1)` unless the `EDITOR` environment variable specifies otherwise.

Only the super-user may edit quotas. In order for quotas to be established on a file system, the root directory of the file system must contain a file, owned by root, called `quotas`. (See `quotaon(1M)`.)

proto_user and *username* can be numeric, corresponding to the UID of a user. Unassigned UIDs may be specified; unassigned names may not. In this way, default quotas can be established for users who are later assigned a UID.

If no options are specified, the temporary file created will have one or more lines of the format, where a block is considered to be a 1024 byte (1K) block:

```
fs mount_point blocks (soft =number, \
     hard =number ) inodes (soft =number, \
     hard =number)
```

The *number* fields may be modified to reflect desired values.

Options The following options are supported:

`-p` Duplicate the quotas of the *proto_user* specified for each *username* specified. This is the normal mechanism used to initialize quotas for groups of users.

`-t` Edit the soft time limits for each file system. If the time limits are zero, the default time limits in `/usr/include/sys/fs/ufs_quota.h` are used. The temporary file created will have one or more lines of the form

fs *mount_point* blocks time limit = *number tmunit*, files time limit = *number tmunit*

tmunit may be one of "month", "week", "day", "hour", "min" or "sec"; characters appended to these keywords are ignored, so you may write "months" or "minutes" if you prefer. The *number* and *tmunit* fields may be modified to set desired values. Time limits are printed in the greatest possible time unit such that the value is greater than or equal to one. If "default" is printed after the *tmunit*, this indicates that the value shown is zero (the default).

Usage See `largefile(5)` for the description of the behavior of `edquota` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`quotas`	quota file at the file system root
`/etc/mnttab`	table of mounted file systems

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `vi(1)`, `quota(1M)`, `quotacheck(1M)`, `quotaon(1M)`, `repquota(1M)`, `attributes(5)`, `largefile(5)`, `quotactl(7I)`

Notes All UIDs can be assigned quotas.

Name eeprom – EEPROM display and load utility

Synopsis `/usr/sbin/eeprom [-] [-f` *device*`] [`*parameter*`[=`*value*`]]`

Description `eeprom` displays or changes the values of parameters in the EEPROM. It processes parameters in the order given. When processing a *parameter* accompanied by a *value*, `eeprom` makes the indicated alteration to the EEPROM; otherwise, it displays the *parameter*'s value. When given no parameter specifiers, `eeprom` displays the values of all EEPROM parameters. A '`–`' (hyphen) flag specifies that parameters and values are to be read from the standard input (one *parameter* or *parameter*=*value* per line).

Only the super-user may alter the EEPROM contents.

`eeprom` verifies the EEPROM checksums and complains if they are incorrect.

platform-name is the name of the platform implementation and can be found using the `-i` option of `uname(1)`.

SPARC SPARC based systems implement firmware password protection with `eeprom`, using the `security-mode`, `security-password` and `security-#badlogins` properties.

x86 EEPROM storage is simulated using a file residing in the platform-specific boot area. The `/boot/solaris/bootenv.rc` file simulates EEPROM storage.

Because x86 based systems typically implement password protection in the system BIOS, there is no support for password protection in the `eeprom` program. While it is possible to set the `security-mode`, `security-password` and `security-#badlogins` properties on x86 based systems, these properties have no special meaning or behavior on x86 based systems.

Options

`-f` *device*
Use *device* as the EEPROM device.

Operands

x86 Only *acpi-user-options*
A configuration variable that controls the use of Advanced Configuration and Power Interface (ACPI), a power management specification. The acceptable values for this variable depend on the release of the Solaris operating system you are using.

For all releases of Solaris 10 and Solaris 11, a value of of `0x0` means that there will be an attempt to use ACPI if it is available on the system. A value of `0x2` disables the use of ACPI.

For the Solaris 10 1/06 release, a value of `0x8` means that there will be an attempt to use ACPI in a mode compatible with previous releases of Solaris 10 if it is available on the system. The default for Solaris 10 1/06 is `0x8`.

For releases of Solaris 10 after the 1/06 release and for Solaris 11, the default is `0x0`.

Most users can safely accept the default value, which enables ACPI if available. If issues related to the use of ACPI are suspected on releases of Solaris after Solaris 1/06, it is suggested to first try a value of `0x8` and then, if you do not obtain satisfactory results, `0x02`.

console
Specifies the console device. Possible values are `ttya`, `ttyb`, and `text`. In `text` mode, console output goes to the frame buffer and input comes from the keyboard. When this property is not present, the console device falls back to the device specified by `input-device` and `output-device`. When neither the console property or the `input-device` and `output-device` property pair are present, the console defaults to the frame buffer and keyboard.

Nvram Configuration Parameters

Not all OpenBoot systems support all parameters. Defaults vary depending on the system and the PROM revision. See the output in the "Default Value" column of the `printenv` command, as entered at the `ok` (OpenBoot) prompt, to determine the default for your system.

auto-boot?
If `true`, boots automatically after power-on or reset. Defaults to `true`. On x86, this parameter is controlled by the grub menu file. See `installgrub(1M)`.

ansi-terminal?
Configuration variable used to control the behavior of the terminal emulator. The value `false` makes the terminal emulator stop interpreting ANSI escape sequences; instead, echoes them to the output device. Defaults to `true`.

boot-args
Holds a string of arguments that are passed to the boot subsystem. For example, you can use `boot-args='-install dhcp'` to request a customer jumpstart installation. See `boot(1M)`, `kadb(1M)` and `kernel(1M)`.

boot-command
Command executed if `auto-boot?` is `true`. Defaults to `boot`.

boot-device
Device from which to boot. *boot-device* may contain 0 or more device specifiers separated by spaces. Each device specifier may be either a prom device alias or a prom device path. The boot prom will attempt to open each successive device specifier in the list beginning with the first device specifier. The first device specifier that opens successfully will be used as the device to boot from. Defaults to `disk net`.

boot-file
File to boot (an empty string lets the secondary booter choose default). Defaults to empty string.

boot-from
Boot device and file (OpenBoot PROM version 1.*x* only). Defaults to `vmunix`.

boot-from-diag
Diagnostic boot device and file (OpenBoot PROM version 1.*x* only). Defaults to `le( )unix`.

boot-ncpus
Configuration variable that controls the number of processors with which the system should boot. By default, the system boots with maximum supported number of processors.

com*X*-noprobe
Where *X* is the number of the serial port, prevents device probe on serial port *X*.

diag-device
Diagnostic boot source device. Defaults to `net`.

diag-file
File from which to boot in diagnostic mode. Defaults to empty string.

diag-level
Diagnostics level. Values include `off`, `min`, `max` and `menus`. There may be additional platform-specific values. When set to `off`, POST is not called. If POST is called, the value is made available as an argument to, and is interpreted by POST. Defaults to `platform-dependent`.

diag-switch?
If `true`, run in diagnostic mode. Defaults to `false` on most desktop systems, `true` on most servers.

error-reset-recovery
Recover after an error reset trap. Defaults to platform-specific setting.

On platforms supporting this variable, it replaces the `watchdog-reboot?`, `watchdog-sync?`, `redmode-reboot?`, `redmode-sync?`, `sir-sync?`, and `xir-sync?` parameters.

The options are:

none
Print a message describing the reset trap and go to OpenBoot PROM's user interface, *aka* `OK` prompt.

sync
Invoke OpenBoot PROM's `sync` word after the reset trap. Some platforms may treat this as `none` after an externally initiated reset (XIR) trap.

boot
Reboot after the reset trap. Some platforms may treat this as `none` after an XIR trap.

fcode-debug?
If `true`, include name parameter for plug-in device FCodes. Defaults to `false`.

hardware-revision
System version information.

input-device
Input device used at power-on (usually `keyboard`, `ttya`, or `ttyb`). Defaults to `keyboard`.

keyboard-click?
If `true`, enable keyboard click. Defaults to `false`.

keyboard-layout
A string that specifies the layout name for non-self-identifying keyboards (type 7c). Invoke `kbd -s` to obtain a list of acceptable layout names. See `kbd(1)`.

keymap
Keymap for custom keyboard.

last-hardware-update
System update information.

load-base
Default load address for client programs. Default value is `16384`.

local-mac-address?
If true, network drivers use their own MAC address, not the system's. Defaults to `false`.

mfg-mode
Manufacturing mode argument for POST. Possible values include `off` or `chamber`. The value is passed as an argument to POST. Defaults to `off`.

mfg-switch?
If true, repeat system self-tests until interrupted with STOP-A. Defaults to `false`.

nvramrc
Contents of NVRAMRC. Defaults to empty.

network-boot-arguments
Arguments to be used by the PROM for network booting. Defaults to an empty string. `network-boot-arguments` can be used to specify the boot protocol (RARP/DHCP) to be used and a range of system knowledge to be used in the process.

The syntax for arguments supported for network booting is:

`[`*protocol*`,] [`*key*`=`*value*`,]*`

All arguments are optional and can appear in any order. Commas are required unless the argument is at the end of the list. If specified, an argument takes precedence over any default values, or, if booting using DHCP, over configuration information provided by a DHCP server for those parameters.

protocol, above, specifies the address discovery protocol to be used.

Configuration parameters, listed below, are specified as *key*=*value* attribute pairs.

`tftp-server`
IP address of the TFTP server

`file`
file to download using TFTP or URL for WAN boot

`host-ip`
IP address of the client (in dotted-decimal notation)

`router-ip`
IP address of the default router (in dotted-decimal notation)

`subnet-mask`
subnet mask (in dotted-decimal notation)

`client-id`
DHCP client identifier

`hostname`
hostname to use in DHCP transactions

`http-proxy`
HTTP proxy server specification (IPADDR[:PORT])

`tftp-retries`
maximum number of TFTP retries

`dhcp-retries`
maximum number of DHCP retries

If no parameters are specified (that is, `network-boot-arguments` is an empty string), the PROM will use the platform-specific default address discovery protocol.

Absence of the protocol parameter when other configuration parameters are specified implies manual configuration.

Manual configuration requires that the client be provided with all the information necessary for boot. If using manual configuration, information required by the PROM to load the second-stage boot program must be provided in `network-boot-arguments` while information required for the second-stage boot program can be specified either as arguments to the `boot` program or by means of the `boot` program's interactive command interpreter.

Information required by the PROM when using manual configuration includes the booting client's IP address, name of the boot file, and the address of the server providing the boot file image. Depending on network configuration, it might be required that the subnet mask and address of the default router to use also be specified.

oem-banner
Custom OEM banner (enabled by setting `oem-banner?` to `true`). Defaults to empty string.

oem-banner?
If `true`, use custom OEM banner. Defaults to `false`.

oem-logo
Byte array custom OEM logo (enabled by setting `oem-logo?` to `true`). Displayed in hexadecimal.

oem-logo?
If `true`, use custom OEM logo (else, use Sun logo). Defaults to `false`.

pci-mem64?
If true, the OpenBoot PROM allocates 64-bit PCI memory addresses to a PCI device that can support 64-bit addresses.

This variable is available on SPARC platforms only and is optional. Some versions of SunOS do not support PCI `MEM64` addresses and will fail in unexpected ways if the OpenBoot PROM allocates PCI `MEM64` addresses.

The default value is system-dependent. If the variable exists, the default value is appropriate to the lowest version of the SunOS that shipped with a specific platform.

output-device
Output device used at power-on (usually `screen`, `ttya`, or `ttyb`). Defaults to `screen`.

redmode-reboot?
Specify `true` to reboot after a redmode reset trap. Defaults to `true`. (Sun Enterprise 10000 only.)

redmode-sync?
Specify `true` to invoke OpenBoot PROM's `sync` word after a redmode reset trap. Defaults to `false`. (Sun Enterprise 10000 only.)

rootpath
Specifies the root device of the operating system.

sbus-probe-list
Designate which SBus slots are probed and in what order. Defaults to `0123`.

screen-#columns
Number of on-screen columns (characters/line). Defaults to `80`.

screen-#rows
Number of on-screen rows (lines). Defaults to `34`.

scsi-initiator-id
SCSI bus address of host adapter, range 0-7. Defaults to `7`.

sd-targets
Map SCSI disk units (OpenBoot PROM version 1.*x* only). Defaults to `31204567`, which means that unit 0 maps to target `3`, unit 1 maps to target `1`, and so on.

security-#badlogins
Number of incorrect security password attempts.This property has no special meaning or behavior on x86 based systems.

security-mode
Firmware security level (options: `none`, `command`, or `full`). If set to `command` or `full`, system will prompt for PROM security password. Defaults to `none`.This property has no special meaning or behavior on x86 based systems.

security-password
Firmware security password (never displayed). Can be set only when `security-mode` is set to `command` or `full`.This property has no special meaning or behavior on x86 based systems.

```
example# eeprom security-password=
Changing PROM password:
New password:
Retype new password:
```

selftest-#megs
Megabytes of RAM to test. Ignored if `diag-switch?` is `true`. Defaults to `1`.

sir-sync?
Specify `true` to invoke OpenBoot PROM's `sync` word after a software-initiated reset (SIR) trap. Defaults to `false`. (Sun Enterprise 10000 only.)

skip-vme-loopback?
If `true`, POST does not do VMEbus loopback tests. Defaults to `false`.

st-targets
Map SCSI tape units (OpenBoot PROM version 1.*x* only). Defaults to `45670123`, which means that unit 0 maps to target `4`, unit 1 maps to target `5`, and so on.

sunmon-compat?
If `true`, display Restricted Monitor prompt (>). Defaults to `false`.

testarea
One-byte scratch field, available for read/write test. Defaults to `0`.

tpe-link-test?
Enable 10baseT link test for built-in twisted pair Ethernet. Defaults to `true`.

ttya-mode
TTYA (baud rate, #bits, parity, #stop, handshake). Defaults to `9600,8,n,1,-`.

Fields, in left-to-right order, are:

Baud rate:
110, 300, 1200, 4800, 9600 . . .

Data bits:
5, 6, 7, 8

Parity:
n(none), e(even), o(odd), m(mark), s(space)

Stop bits:
1, 1.5, 2

Handshake:
–(none), h(hardware:rts/cts), s(software:xon/xoff)

ttyb-mode
TTYB (baud rate, #bits, parity, #stop, handshake). Defaults to `9600,8,n,1,–`.

Fields, in left-to-right order, are:

Baud rate:
110, 300, 1200, 4800, 9600 . . .

Data bits:
5, 6, 7, 8

Stop bits:
1, 1.5, 2

Parity:
n(none), e(even), o(odd), m(mark), s(space)

Handshake:
–(none), h(hardware:rts/cts), s(software:xon/xoff)

ttya-ignore-cd
If `true`, operating system ignores carrier-detect on TTYA. Defaults to `true`.

ttyb-ignore-cd
If `true`, operating system ignores carrier-detect on TTYB. Defaults to `true`.

ttya-rts-dtr-off
If `true`, operating system does not assert DTR and RTS on TTYA. Defaults to `false`.

ttyb-rts-dtr-off
If `true`, operating system does not assert DTR and RTS on TTYB. Defaults to `false`.

use-nvramrc?
If `true`, execute commands in NVRAMRC during system start-up. Defaults to `false`.

verbosity
Controls the level of verbosity of PROM messages. Can be one of `debug`, `max`, `normal`, `min`, or `none`. Defaults to `normal`.

version2?
If `true`, hybrid (1.*x*/2.*x*) PROM comes up in version 2.*x*. Defaults to `true`.

watchdog-reboot?
If `true`, reboot after watchdog reset. Defaults to `false`.

watchdog-sync?
Specify `true` to invoke OpenBoot PROM's `sync` word after a watchdog reset trap. Defaults to `false`. (Sun Enterprise 10000 only.)

xir-sync?
Specify `true` to invoke OpenBoot PROM's `sync` word after an XIR trap. Defaults to `false`. (Sun Enterprise 10000 only.)

Examples

EXAMPLE 1 Changing the Number of Megabytes of RAM.

The following example demonstrates the method for changing from one to two the number of megabytes of RAM that the system will test.

```
example# eeprom selftest-#megs
selftest-#megs=1

example# eeprom selftest-#megs=2

example# eeprom selftest-#megs
selftest-#megs=2
```

EXAMPLE 2 Setting the `auto-boot?` Parameter to `true`.

The following example demonstrates the method for setting the `auto-boot?` parameter to `true`.

```
example# eeprom auto-boot?=true
```

When the `eeprom` command is executed in user mode, the parameters with a trailing question mark (?) need to be enclosed in double quotation marks (“ “) to prevent the shell from interpreting the question mark. Preceding the question mark with an escape character (\) will also prevent the shell from interpreting the question mark.

```
example% eeprom "auto-boot?"=true
```

EXAMPLE 3 Using `network-boot-arguments`

To use DHCP as the boot protocol and a hostname of `abcd.example.com` for network booting, set these values in `network-boot-arguments` as:

```
example# eeprom network-boot-arguments="dhcp,hostname=abcd.example.com"
```

...then boot using the command:

```
ok boot net
```

Note that network boot arguments specified from the PROM command line cause the contents of `network-boot-arguments` to be ignored. For example, with `network-boot-arguments` set as shown above, the `boot` command:

```
ok boot net:dhcp
```

EXAMPLE 3 Using `network-boot-arguments` *(Continued)*

...causes DHCP to be used, but the `hostname` specified in `network-boot-arguments` will not be used during network boot.

EXAMPLE 4 Setting System Console to Auxiliary Device

The command below assigns the device `/dev/term/a` as the system console device. You would make such an assignment prior to using `tip(1)` to establish a `tip` connection to a host.

On a SPARC machine:

```
# eeprom output-device=/dev/term/a
```

On an x86 machine:

```
# eeprom console=ttya
```

On a SPARC machine, the preceding command would be sufficient for assigning the console to an auxiliary device. For an x86 machine, you might, in addition, need to set the characteristics of the serial line, for which you would have to consult the BIOS documentation for that machine. Also, on some x86 machines, you might use a device other than device `a`, as shown above. For example, you could set console to `ttyb` if the second serial port is present.

Files

`/boot/solaris/bootenv.rc`
File storing `eeprom` values on x86 machines.

`/dev/openprom`
Device file

`/usr/platform/`*platform-name*`/sbin/eeprom`
Platform-specific version of `eeprom`. Use `uname -i` to obtain *platform-name*.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`passwd(1)`, `sh(1)`, `svcs(1)`, `tip(1)`, `uname(1)`, `boot(1M)`, `kadb(1M)`, `kernel(1M)`, `init(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

OpenBoot 3.x Command Reference Manual

ONC+ Developer's Guide

Name efdaemon – embedded FCode interpreter daemon

Synopsis

```
/usr/lib/efcode/sparcv9/efdaemon [-d]
```

Description `efdaemon`, the embedded `FCode` interpreter daemon, invokes the embedded `FCode` interpreter when the daemon receives an interpretation request. A new session of the interpreter is started for each unique request by invoking the script `/usr/lib/efcode/efcode`.

`efdaemon` is used on selected platforms as part of the processing of some dynamic reconfiguration events.

Options The following option is supported:

`-d` Set debug output. Log debug messages as `LOG_DEBUG` level messages by using `syslog()`. See `syslog(3C)`.

Files

`/dev/fcode`	`FCode` interpreter pseudo device, which is a portal for receipt of `FCode` interpretation requests
`/usr/lib/efcode/efcode`	Shell script that invokes the embedded `FCode` interpreter
`/usr/lib/efcode/interpreter`	Embedded `FCode` interpreter
`/usr/lib/efcode/sparcv9/interpreter`	Embedded `FCode` interpreter

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWefcx, SUNWefcux, SUNWefcr, SUNWefclx

See Also `svcs(1)`, `prtconf(1M)`, `svcadm(1M)`, `syslog(3C)`, `attributes(5)`, `smf(5)`

Notes The `efdaemon` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/platform/sun4u/efdaemon:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name embedded_su – allow an application to prompt for credentials and execute commands as the super user or another user

Synopsis

```
/usr/lib/embedded_su [-] [username [arg...]]
```

Description The `embedded_su` command allows an application to prompt the user for security credentials and then use those credentials to execute a program as another user or role (see `rbac(5)` for information on role-based access control). The default *username* is `root` (super user).

`embedded_su` is identical to `su(1M)`, except that the user interaction is packaged in a form suitable for another program to interpret and display. Typically, `embedded_su` would be used to allow a graphical program to prompt for the super user password and execute a command as the super user, without requiring that the requesting program be run as the super user.

PROTOCOL `embedded_su` implements a simple protocol over standard input, standard output, and standard error. This protocol consists of three phases, roughly corresponding to PAM initialization, the PAM dialog, and PAM completion.

Phase 1: Initialization

After starting `embedded_su`, the application must send an initialization block on `embedded_su`'s standard input. This block is a text block, as described under "Text Blocks". There are currently no initialization parameters defined; the application should send an empty block by sending a line consisting solely of a period (.).

Phase 2: Conversation

`embedded_su` then emits zero or more conversation blocks on its standard output. Each conversation block may require zero or more responses.

A conversation block starts with a line consisting of the word `CONV`, followed by whitespace, followed by the number of messages in the conversation block as a decimal integer. The number of messages may be followed by whitespace and additional data. This data, if present, must be ignored.

Each message consists of a line containing a header followed by a text block, as described under "Text Blocks". A single newline is appended to each message, allowing the message to end with a line that does not end with a newline.

A message header line consists of a PAM message style name, as described in `pam_start(3PAM)`. The message header values are:

`PAM_PROMPT_ECHO_OFF` The application is to prompt the user for a value, with echoing disabled.

`PAM_PROMPT_ECHO_ON` The application is to prompt the user for a value, with echoing enabled.

`PAM_ERROR_MSG` The application is to display the message in a form appropriate for displaying an error.

PAM_TEXT_INFO	The application is to display the message in a form appropriate for general information.

The PAM message style may be followed by whitespace and additional data. This data, if present, must be ignored.

After writing all of the messages in the conversation block, if any of them were PAM_PROMPT_ECHO_OFF or PAM_PROMPT_ECHO_ON, embedded_su waits for the response values. It expects the response values one per line, in the order the messages were given.

Phase 3: Completion

After zero or more conversation blocks, embedded_su emits a result block instead of a conversation block.

Upon success, embedded_su emits a single line containing the word "SUCCESS". The word SUCCESS may be followed by whitespace and additional data. This data, if present, must be ignored.

Upon failure, embedded_su emits a single line containing the word "ERROR", followed by a text block as described under "Text Bocks". The text block gives an error message. The word ERROR may be followed by whitespace and additional data. This data, if present, must be ignored.

Text Blocks

Initialization blocks, message blocks, and error blocks are all text blocks. These are blocks of text that are terminated by a line containing a single period (.). Lines in the block that begin with a "." have an extra "." prepended to them.

Internationalization — All messages are localized to the current locale; no further localization is required.

SECURITY — embedded_su uses pam(3PAM) for authentication, account management, and session management. Its primary function is to export the PAM conversation mechanism to an unprivileged program. Like su(1M), the PAM configuration policy can be used to control embedded_su. The PAM service name used is "embedded_su".

embedded_su is almost exactly equivalent to su(1M) for security purposes. The only exception is that it is slightly easier to use embedded_su in writing a malicious program that might trick a user into providing secret data. For those sites needing maximum security, potentially at the expense of application functionality, the EXAMPLES section shows how to disable embedded_su.

Examples

In the following examples, left angle brackets (<<<) indicate a line written by embedded_su and read by the invoking application. Right angle brackets (>>>) indicate a line written by the application and read by embedded_su.

EXAMPLE 1 Executing a command with the Correct Password

The following example shows an attempt to execute "somecommand" as "someuser", with the correct password supplied:

```
/usr/lib/embedded_su someuser -c somecommand
   >>>.
   <<<CONV 1
   <<<PAM_PROMPT_ECHO_OFF
   <<<Password:
   <<<.
   >>>[ correct password ]
   <<<SUCCESS
   [ somecommand executes  ]
```

EXAMPLE 2 Executing a command with the Incorrect Password

The following example shows an attempt to execute "somecommand" as "someuser", with the incorrect password supplied:

```
/usr/lib/embedded_su someuser -c somecommand
   >>>.
   <<<CONV 1
   <<<PAM_PROMPT_ECHO_OFF
   <<<Password:
   <<<.
   >>>[ incorrect password ]
   [ delay ]
   <<<ERROR
   <<<embedded_su:Sorry
   <<<.
   [ exit ]
```

EXAMPLE 3 Message Examples

A `pam_message` structure with *msg_style* equal to `PAM_TEXT_INFO` and `msg` equal to "foo" produces:

```
PAM_TEXT_INFO
foo
.
```

A `pam_message` structure with *msg_style* equal to `PAM_ERROR_MESSAGE` and `msg` equal to "bar\n" produces:

```
PAM_ERROR_MESSAGE
bar
[ blank line ]
.
```

EXAMPLE 3 Message Examples *(Continued)*

A `pam_message` structure with *msg_style* equal to `PAM_ERROR_MESSAGE` and `msg` equal to "aaa\nbbb" produces:

```
PAM_ERROR_MESSAGE
aaa
bbb
.
```

A `pam_message` structure with *msg_style* equal to `PAM_TEXT_INFO` and `msg` equal to "" produces:

```
PAM_TEXT_INFO
[ blank line ]
.
```

A `pam_message` structure with *msg_style* equal to `PAM_TEXT_INFO` and `msg` equal to NULL produces:

```
PAM_TEXT_INFO
.
```

EXAMPLE 4 Disabling embedded_su

To disable `embedded_su`, add a line to the `/etc/pam.conf` file similar to:

```
embedded_su  auth  requisite  pam_deny.so.1
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Stable

See Also

`su(1M)`, `pam(3PAM)`, `pam_start(3PAM)`, `attributes(5)`, `rbac(5)`

Name etrn – start mail queue run

Synopsis

```
etrn [-b] [-v] server-host [client-hosts]
```

Description SMTP's ETRN command allows an SMTP client and server to interact, giving the server an opportunity to start the processing of its queues for messages to go to a given host. This is meant to be used in start-up conditions, as well as for mail nodes that have transient connections to their service providers.

The etrn utility initiates an SMTP session with the host *server-host* and sends one or more ETRN commands as follows: If no *client-hosts* are specified, etrn looks up every host name for which sendmail(1M) accepts email and, for each name, sends an ETRN command with that name as the argument. If any *client-hosts* are specified, etrn uses each of these as arguments for successive ETRN commands.

Options The following options are supported:

`-b` System boot special case. Make sure localhost is accepting SMTP connections before initiating the SMTP session with server-host.

This option is useful because it prevents race conditions between sendmail(1M) accepting connections and *server-host* attempting to deliver queued mail. This check is performed automatically if no *client-hosts* are specified.

`-v` The normal mode of operation for etrn is to do all of its work silently. The `-v` option makes it verbose, which causes etrn to display its conversations with the remote SMTP server.

Environment Variables No environment variables are used. However, at system start-up, `svc:/network/smtp:sendmail` reads `/etc/default/sendmail`. In this file, if the variable ETRN_HOSTS is set, `svc:/network/smtp:sendmail` parses this variable and invokes etrn appropriately. ETRN_HOSTS should be of the form:

```
"s1:c1.1,c1.2        s2:c2.1 s3:c3.1,c3.2,c3.3"
```

That is, white-space separated groups of *server:client* where *client* can be one or more comma-separated names. The *:client* part is optional. *server* is the name of the server to prod; a mail queue run is requested for each *client* name. This is comparable to running:

```
/usr/lib/sendmail -qR client
```

on the host *server*.

Examples **EXAMPLE 1** Using etrn

Inserting the line:

```
ETRN_HOSTS="s1.domain.com:clnt.domain.com s2.domain.com:clnt.domain.com"
```

EXAMPLE 1 Using `etrn` *(Continued)*

in `/etc/default/sendmail` results in `svc:/network/smtp:sendmail` invoking `etrn` such that ETRN commands are sent to both `s1.domain.com` and `s2.domain.com`, with both having `clnt.domain.com` as the ETRN argument.

The line:

```
ETRN_HOSTS="server.domain.com:client1.domain.com,client2.domain.com"
```

results in two ETRN commands being sent to `server.domain.com`, one with the argument `client1.domain.com`, the other with the argument `client2.domain.com`.

The line:

```
ETRN_HOSTS="server1.domain.com server2.domain.com"
```

results in set of a ETRN commands being sent to both `server1.domain.com` and `server2.domain.com`; each set contains one ETRN command for each host name for which `sendmail(1M)` accepts email, with that host name as the argument.

Files

`/etc/mail/sendmail.cf`	`sendmail` configuration file
`/etc/default/sendmail`	Variables used by `svc:/network/smtp:sendmail`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsndmu
Interface Stability	Stable

See Also `sendmail(1M)`, `attributes(5)`

RFC 1985

Notes Not all SMTP servers support ETRN.

Name fbconfig – Frame Buffer configuration utility

Synopsis

```
fbconfig  [-list | -gui | -help]

fbconfig  [-dev device_filename] [-prconf] [-propt]
     [-res \?]

fbconfig  [-dev device_filename] [-res resolution-specification]
     device_specific_options
```

Description `fbconfig` is the generic command line interface to query and configure frame buffer attributes.

The following form of `fbconfig` is the interface for the device independent operations performed by `fbconfig`:

```
fbconfig  [-list | -gui | -help]
```

The following form of `fbconfig` is the interface for configuring a frame buffer:

```
fbconfig  [-dev device_filename] [-prconf] [-propt]
 [-res]
```

If the `-dev` option is omitted, the default frame buffer (`/dev/fb` or /dev/fb0) is assumed. In the absence of specific options, the response will depend upon the device specific configuration program and how it responds to no options

Options The following options are supported:

`-dev` *device_filename* — Specify the FFB special file. The default is `/dev/fbs/ffb0`.

`-gui` — Invoke the Graphical User Interface (GUI). The GUI is available if the SUNWdcm package is installed. All other arguments are ignored.

The GUI can configure devices (as an alternative to the `fbconfig` command line) and can update the `Xservers` file without directly editing the file. The GUI allows the user that is logged in on the graphics device or devices to configure which graphics displays the window system should use, their screen layout (where they appear on the user's desktop), and screen properties (X attributes).

In addition, the GUI allows advanced users to create a new video format (resolution) that some graphics devices can select from `fbconfig` command line or from the device-dependent portion of the GUI. The GUI's online help explains all options and features.

`-help` — Print the `fbconfig` command usage summary. This is the default option.

`-list` Print the list of installed frame buffers and associated device specific configuration routines.

```
Device Filename    Specific Config Program
---------------    -----------------------
 /dev/fbs/ffb0     SUNWffb_config
 /dev/fbs/ffb1     SUNWffb_config
 /dev/fbs/m640     SUNWm64_config
 /dev/fbs/cgsix0   not configurable
```

`-prconf` Print the current hardware configuration.

`-propt` Print the current software configuration.

`-res \?` Print the current hardware resolution.

Operands The following operands are supported:

device_specific_options *device_specific_options* are specified in the format shown by the `-help` output, or the corresponding device-specific man page.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfbc

See Also `SUNWgfb_config(1M)`, `SUNWifb_config(1M)`, `SUNWpfb_config(1M)`, `SUNWzulu_config(1M)`, `afbconfig(1M)`, `ffbconfig(1M)`, `m64config(1M)`, `pgxconfig(1M)`, `attributes(5)`

Limitations Because of limitations in the `m64` kernel driver and related software, `fbconfig` (with the `-prconf` option) is unable to distinguish between a current depth of 24 or 8+24. The `-propt` option returns the depth specified in the `OWconfig` file, which will be in effect following the next restart of the window system. The `xwininfo` utility, usually shipped in the package containing frame buffer software (such as `SUNWxwplt`), reports current depth of a specified window.

Name fcinfo – Fibre Channel HBA Port Command Line Interface

Synopsis

```
fcinfo hba-port [-l] [HBA_port_WWN]...

fcinfo remote-port [-ls] [-p HBA_port_WWN]
     [REMOTE_port_WWN]...

fcinfo [-V]

fcinfo [-?]
```

Description `fcinfo` is a command line interface that collects administrative information on fibre channel host bus adapter (HBA) ports on a host. It also collects data on any fibre channel targets that may be connected to those ports in a Storage Area Network (SAN).

SUBCOMMANDS The following subcommands are supported:

hba-port Lists information for the HBA port referenced by the specified *HBA_port_WWN*. If *HBA_port_WWN* is not specified, all fibre channel HBA ports on the host will be listed.

remote-port Lists the *remote-port* information for those remote ports that are specified. If no *REMOTE_port_WWN* is specified, all remote ports that are visible through *HBA_port_WWN* are listed.

Options The following options are supported:

`-l`, `--linkstat` Lists the link error statistics information for the port referenced by the specified *HBA_port_WWN* or *REMOTE_port_WWN*.

`-p` *HBA_port_WWN*, `--port` *HBA_port_WWN* Retrieve remote port information from the *HBA_port_WWN* of the local HBA port on the host. The `-p` option can only be used with the `remote-port` subcommand and is a mandatory option.

`-s`, `--scsi-target` Lists the SCSI target information for all remote ports the user has asked for. The `-p`, `--port` option must always be specified and must be a valid HBA port on the host. This HBA port will be used as the initiator for which to retrieve the SCSI level target information. Note that this will only function on remote port fibre channel World-Wide Names that support an FC4 type of SCSI.

`-V`, `--version`	Prints the version information.
`-?`, `--help`	Prints the usage information.

Examples

EXAMPLE 1 Listing all HBA ports

The following command lists all fibre channel HBA ports on the host:

```
# fcinfo hba-port

HBA Port WWN: 210000e08b074cb5
        OS Device Name: /dev/cfg/c1
        Manufacturer: QLogic Corp.
        Model: 375-3108-xx
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Type: N-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200000e08b074cb5
HBA Port WWN: 210100e08b274cb5
        OS Device Name: /dev/cfg/c2
        Manufacturer: QLogic Corp.
        Model: 375-3108-xx
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Type: N-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200100e08b274cb5
HBA Port WWN: 210000e08b072ab5
        OS Device Name: /dev/cfg/c3
        Manufacturer: QLogic Corp.
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Model: 375-3108-xx
        Type: L-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200000e08b072ab5
HBA Port WWN: 210100e08b272ab5
        OS Device Name: /dev/cfg/c4
        Manufacturer: QLogic Corp.
        Model: 375-3108-xx
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
```

EXAMPLE 1 Listing all HBA ports *(Continued)*

```
        Type: N-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200100e08b272ab5
```

EXAMPLE 2 Listing HBA ports and link statistics

The following command lists information for the HBA ports and the link statistics for those ports:

```
# fcinfo hba-port -l 210000e08b074cb5 210100e08b274cb5

HBA Port WWN: 210000e08b074cb5
        OS Device Name: /dev/cfg/c1
        Manufacturer: QLogic Corp.
        Model: 375-3108-xx
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Type: N-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200000e08b074cb5
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
HBA Port WWN: 210100e08b274cb5
        OS Device Name: /dev/cfg/c2
        Manufacturer: QLogic Corp.
        Model: 375-3108-xx
        Firmware Version: 3.3.116
        FCode/BIOS Version: 1.13.08
        Type: N-port
        State: online
        Supported Speeds: 1Gb 2Gb
        Current Speed: 2Gb
        Node WWN: 200100e08b274cb5
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
```

EXAMPLE 2 Listing HBA ports and link statistics *(Continued)*

```
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
```

EXAMPLE 3 Listing all remote ports

The following command lists all remote ports that are visible through the given HBA port:

```
# fcinfo remote-port -p 210100e08b274cb5

Remote Port WWN: 50020f230000b4af
        Active FC4 Types: SCSI
        SCSI Target: yes
        Node WWN: 50020f200000b4af
Remote Port WWN: 210000e08b07daa6
        Active FC4 Types: SCSI
        SCSI Target: no
        Node WWN: 200000e08b07daa6
Remote Port WWN: 20030003ba27c788
        Active FC4 Types: SCSI
        SCSI Target: yes
        Node WWN: 10000003ba27c788
Remote Port WWN: 210000e08b096a60
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Node WWN: 200000e08b096a60
```

EXAMPLE 4 Listing remote ports and link statistics

The following command lists information for the remote ports and the link statistics for those ports:

```
# fcinfo remote-port -l -p 210100e08b272ab5

Remote Port WWN: 210100e08b296a60
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Node WWN: 200100e08b296a60
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 20030003ba27d56d
        Active FC4 Types: SCSI
```

EXAMPLE 4 Listing remote ports and link statistics *(Continued)*

```
        SCSI Target: yes
        Node WWN: 10000003ba27d56d
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 4765165
                Loss of Signal Count: 4765165
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210100e08b27f7a6
        Active FC4 Types: SCSI
        SCSI Target: no
        Node WWN: 200100e08b27f7a6
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 50020f230000b897
        Active FC4 Types: SCSI
        SCSI Target: yes
        Node WWN: 50020f200000b897
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 7
                Loss of Signal Count: 7
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210100e08b27daa6
        Active FC4 Types: SCSI
        SCSI Target: no
        Node WWN: 200100e08b27daa6
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210000e08b074cb5
        Active FC4 Types: SCSI,IP
        SCSI Target: no
```

EXAMPLE 4 Listing remote ports and link statistics *(Continued)*

```
        Node WWN: 200000e08b074cb5
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210100e08b296060
        Active FC4 Types: SCSI
        SCSI Target: no
        Node WWN: 200100e08b296060
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
```

EXAMPLE 5 Listing all SCSI targets and link statistics

The following command lists all remote ports as well as the link statistics and *scsi-target* information:

```
# fcinfo remote-port  -sl -p 210100e08b272ab5

Remote Port WWN: 210100e08b296a60
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Node WWN: 200100e08b296a60
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 20030003ba27d56d
        Active FC4 Types: SCSI
        SCSI Target: yes
        Node WWN: 10000003ba27d56d
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 4765165
                Loss of Signal Count: 4765165
```

EXAMPLE 5 Listing all SCSI targets and link statistics *(Continued)*

```
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
        LUN: 0
          Vendor: SUN
          Product: T4
          OS Device Name: /dev/rdsk/c4t20030003BA27D56Dd0s2
        LUN: 1
          Vendor: SUN
          Product: T4
          OS Device Name: /dev/rdsk/c4t20030003BA27D56Dd1s2
Remote Port WWN: 210100e08b27f7a6
        Active FC4 Types: SCSI
        SCSI Target: no
        Node WWN: 200100e08b27f7a6
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 50020f230000b897
        Active FC4 Types: SCSI
        SCSI Target: yes
        Node WWN: 50020f200000b897
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 7
                Loss of Signal Count: 7
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
        LUN: 0
          Vendor: SUN
          Product: T300
          OS Device Name: Unknown
        LUN: 1
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d1s2
        LUN: 2
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d2s2
```

EXAMPLE 5 Listing all SCSI targets and link statistics *(Continued)*

```
        LUN: 3
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d3s2
        LUN: 4
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d4s2
        LUN: 5
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d5s2
        LUN: 6
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d6s2
        LUN: 7
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d7s2
        LUN: 8
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d8s2
        LUN: 9
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d9s2
        LUN: 10
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d10s2
        LUN: 11
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d11s2
        LUN: 12
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d12s2
        LUN: 13
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d13s2
        LUN: 14
```

EXAMPLE 5 Listing all SCSI targets and link statistics *(Continued)*

```
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d14s2
        LUN: 15
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d15s2
Remote Port WWN: 210100e08b27daa6
        Active FC4 Types: SCSI
        SCSI Target: no
        Node WWN: 200100e08b27daa6
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210000e08b074cb5
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Node WWN: 200000e08b074cb5
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
Remote Port WWN: 210100e08b296060
        Active FC4 Types: SCSI
        SCSI Target: no
        Node WWN: 200100e08b296060
        Link Error Statistics:
                Link Failure Count: 0
                Loss of Sync Count: 0
                Loss of Signal Count: 0
                Primitive Seq Protocol Error Count: 0
                Invalid Tx Word Count: 0
                Invalid CRC Count: 0
```

EXAMPLE 6 Listing SCSI target information

The following command lists all remote ports as well as the *scsi-target* information:

EXAMPLE 6 Listing SCSI target information *(Continued)*

```
# fcinfo remote-port  -s -p 210100e08b272ab5

Remote Port WWN: 210100e08b296a60
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Node WWN: 200100e08b296a60
Remote Port WWN: 20030003ba27d56d
        Active FC4 Types: SCSI
        SCSI Target: yes
        Node WWN: 10000003ba27d56d
        LUN: 0
          Vendor: SUN
          Product: T4
          OS Device Name: /dev/rdsk/c4t20030003BA27D56Dd0s2
        LUN: 1
          Vendor: SUN
          Product: T4
          OS Device Name: /dev/rdsk/c4t20030003BA27D56Dd1s2
Remote Port WWN: 210100e08b27f7a6
        Active FC4 Types: SCSI
        SCSI Target: no
        Node WWN: 200100e08b27f7a6
Remote Port WWN: 50020f230000b897
        Active FC4 Types: SCSI
        SCSI Target: yes
        Node WWN: 50020f200000b897
        LUN: 0
          Vendor: SUN
          Product: T300
          OS Device Name: Unknown
        LUN: 1
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d1s2
        LUN: 2
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d2s2
        LUN: 3
          Vendor: SUN
          Product: T300
          OS Device Name: /dev/rdsk/c4t50020F230000B897d3s2
        LUN: 4
          Vendor: SUN
          Product: T300
```

EXAMPLE 6 Listing SCSI target information *(Continued)*

```
  OS Device Name: /dev/rdsk/c4t50020F230000B897d4s2
LUN: 5
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d5s2
LUN: 6
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d6s2
LUN: 7
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d7s2
LUN: 8
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d8s2
LUN: 9
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d9s2
LUN: 10
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d10s2
LUN: 11
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d11s2
LUN: 12
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d12s2
LUN: 13
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d13s2
LUN: 14
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d14s2
LUN: 15
  Vendor: SUN
  Product: T300
  OS Device Name: /dev/rdsk/c4t50020F230000B897d15s2
```

EXAMPLE 6 Listing SCSI target information *(Continued)*

```
Remote Port WWN: 210100e08b27daa6
        Active FC4 Types: SCSI
        SCSI Target: no
        Node WWN: 200100e08b27daa6
Remote Port WWN: 210000e08b074cb5
        Active FC4 Types: SCSI,IP
        SCSI Target: no
        Node WWN: 200000e08b074cb5
Remote Port WWN: 210100e08b296060
        Active FC4 Types: SCSI
        SCSI Target: no
        Node WWN: 200100e08b296060
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNW
Interface Stability	Evolving

See Also

`attributes(5)`

Error Messages

Errors that can occur in addition to the errors normally associated with system administration commands:

HBA_port_WWN`: not found`

REMOTE_port_WWN`: not found`

Name fdetach – detach a name from a STREAMS-based file descriptor

Synopsis

```
fdetach path
```

Description The `fdetach` command detaches a STREAMS-based file descriptor from a name in the file system. Use the *path* operand to specify the path name of the object in the file system name space, which was previously attached. See `fattach(3C)`.

The user must be the owner of the file or a user with the appropriate privileges. All subsequent operations on *path* will operate on the underlying file system entry and not on the STREAMS file. The permissions and status of the entry are restored to the state they were in before the STREAMS file was attached to the entry.

Operands The following operands are supported:

path Specifies the path name of the object in the file system name space, which was previously attached.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `fattach(3C)`, `fdetach(3C)`, `attributes(5)`, `streamio(7I)`

STREAMS Programming Guide

Name fdisk – create or modify fixed disk partition table

Synopsis

```
fdisk [-o offset] [-s size] [-P fill_patt] [-S geom_file]
     [-w | -r | -d | -n | -I | -B | -t | -T | -g | -G | -R | -E]
     [--F fdisk_file] [ [-v] -W {fdisk_file | –}]
     [-h] [-b masterboot]
     [-A id : act : bhead : bsect : bcyl : ehead : esect :
         ecyl : rsect : numsect]
     [-D id : act : bhead: bsect : bcyl : ehead: esect :
         ecyl : rsect : numsect] rdevice
```

Description This command is used to do the following:

- Create and modify an `fdisk` partition table on x86 systems
- Create and modify an `fdisk` partition table on removable media on SPARC or x86 systems
- Install the master boot record that is put in the first sector of the fixed disk on x86 systems only

This table is used by the first-stage bootstrap (or firmware) to identify parts of the disk reserved for different operating systems, and to identify the partition containing the second-stage bootstrap (the *active* Solaris partition). The *rdevice* argument must be used to specify the raw device associated with the fixed disk, for example, `/dev/rdsk/c0t0d0p0`.

The program can operate in three different modes. The first is interactive mode. In interactive mode, the program displays the partition table as it exists on the disk, and then presents a menu allowing the user to modify the table. The menu, questions, warnings, and error messages are intended to be self-explanatory.

In interactive mode, if there is no partition table on the disk, the user is given the options of creating a default partitioning or specifying the initial table values. The default partitioning allocates the entire disk for the Solaris system and makes the Solaris system partition active. In either case, when the initial table is created, `fdisk` also writes out the first-stage bootstrap (x86 only) code along with the partition table. In this mode (x86 only), when creating an entry for a non-EFI partition on a disk that is larger than 2 TB (terabytes), `fdisk` warns that the maximum size of the partition is 2 TB. Under these conditions percentages displayed by `fdisk` are based on 2 TB.

The second mode of operation is used for automated entry addition, entry deletion, or replacement of the entire `fdisk` table. This mode can add or delete an entry described on the command line. In this mode the entire `fdisk` table can be read in from a file replacing the original table. `fdisk` can also be used to create this file. There is a command line option that will cause `fdisk` to replace any `fdisk` table with the default of the whole disk for the Solaris system.

The third mode of operation is used for disk diagnostics. In this mode, a section of the disk can be filled with a user specified pattern, and mode sections of the disk can also be read or written.

When `fdisk` creates a partition, the space is allocated in the `fdisk` partition table, but the allocated disk space is not initialized. `newfs(1M)` is required to create and write file system metadata to the new partition, and `format(1M)` is required to write the VTOC or EFI/GPT metadata.

Menu Options

The menu options for interactive mode given by the `fdisk` program are:

`Create a partition`
: This option allows the user to create a new partition. The maximum number of partitions is 4. The program will ask for the type of the partition (SOLARIS, MS-DOS, UNIX, or other). It will then ask for the size of the partition as a percentage of the disk. The user may also enter the letter `c` at this point, in which case the program will ask for the starting cylinder number and size of the partition in cylinders. If a `c` is not entered, the program will determine the starting cylinder number where the partition will fit. In either case, if the partition would overlap an existing partition or will not fit, a message is displayed and the program returns to the original menu.

`Change Active (Boot from) partition`
: This option allows the user to specify the partition where the first-stage bootstrap will look for the second-stage bootstrap, otherwise known as the *active* partition.

`Delete a partition`
: This option allows the user to delete a previously created partition. Note that this will destroy all data in that partition.

`Change between Solaris and Solaris2 Partition IDs`
: This option allows the user to switch between the current `fdisk` operating system partition identifier and the previous one. This does not affect any data in the disk partition and is provided for compatibility with older software.

Use the following options to include your modifications to the partition table at this time or to cancel the session without modifying the table:

`Exit` This option writes the new version of the table created during this session with `fdisk` out to the fixed disk, and exits the program.

`Cancel` This option exits without modifying the partition table.

Options

The following options apply to `fdisk`:

`-A` *id:act:bhead:bsect:bcyl:ehead:esect:ecyl:rsect:numsect*
: Add a partition as described by the argument (see the `-F` option below for the format). Use of this option will zero out the VTOC on the Solaris partition if the `fdisk` table changes.

`-b` *master_boot*
: Specify the file *master_boot* as the master boot program. The default master boot program is `/usr/lib/fs/ufs/mboot`.

`-B`
Default to one Solaris partition that uses the whole disk. On an x86 machine, if the disk is larger than 2 TB (terabytes), the default size of the Solaris partition will be limited to 2 TB.

`-d`
Turn on verbose *debug* mode. This will cause `fdisk` to print its state on stderr as it is used. The output from this option should not be used with `-F`.

`-D` *id:act:bhead:bsect:bcyl:ehead:esect:ecyl:rsect:numsect*
Delete a partition as described by the argument (see the `-F` option below for the format). Note that the argument must be an exact match or the entry will not be deleted! Use of this option will zero out the VTOC on the Solaris partition if the `fdisk` table changes.

`-E`
Create an EFI partition that uses the entire disk.

`-F` *fdisk_file*
Use fdisk file *fdisk_file* to initialize table. Use of this option will zero out the VTOC on the Solaris partition if the `fdisk` table changes.

The *fdisk_file* contains up to four specification lines. Each line is delimited by a new-line character (`\n`). If the first character of a line is an asterisk (`*`), the line is treated as a comment. Each line is composed of entries that are position-dependent, are separated by ‘‘white space'' or colons, and have the following format:

id act bhead bsect bcyl ehead esect ecyl rsect numsect

where the entries have the following values:

id	This is the type of partition and the correct numeric values may be found in `fdisk.h`.
act	This is the active partition flag; `0` means not active and `128` means active.
bhead	This is the head where the partition starts. If this is set to `0`, `fdisk` will correctly fill this in from other information.
bsect	This is the sector where the partition starts. If this is set to `0`, `fdisk` will correctly fill this in from other information.
bcyl	This is the cylinder where the partition starts. If this is set to `0`, `fdisk` will correctly fill this in from other information.
ehead	This is the head where the partition ends. If this is set to `0`, `fdisk` will correctly fill this in from other information.
esect	This is the sector where the partition ends. If this is set to `0`, `fdisk` will correctly fill this in from other information.
ecyl	This is the cylinder where the partition ends. If this is set to `0`, `fdisk` will correctly fill this in from other information.

rsect The relative sector from the beginning of the disk where the partition starts. This must be specified and can be used by `fdisk` to fill in other fields.

numsect The size in sectors of this disk partition. This must be specified and can be used by `fdisk` to fill in other fields.

`-g`
Get the label geometry for disk and display on stdout (see the `-S` option for the format).

`-G`
Get the physical geometry for disk and display on stdout (see the `-S` option for the format).

`-h`
Issue verbose message; message will list all options and supply an explanation for each.

`-I`
Forgo device checks. This is used to generate a file image of what would go on a disk without using the device. Note that you must use `-S` with this option (see above).

`-n`
Don't update `fdisk` table unless explicitly specified by another option. If no other options are used, `-n` will only write the master boot record to the disk. In addition, note that `fdisk` will not come up in interactive mode if the `-n` option is specified.

`-o` *offset*
Block offset from start of disk. This option is used for `-P`, `-r`, and `-w`. Zero is assumed when this option is not used.

`-P` *fill_patt*
Fill disk with pattern *fill_patt*. *fill_patt* can be decimal or hex and is used as number for constant long word pattern. If *fill_patt* is `#`, then pattern is block # for each block. Pattern is put in each block as long words and fills each block (see `-o` and `-s`).

`-r`
Read from disk and write to stdout. See `-o` and `-s`, which specify the starting point and size of the operation.

`-R`
Treat disk as read-only. This is for testing purposes.

`-s` *size*
Number of blocks to perform operation on (see `-o`).

`-S` *geom_file*
Set the label geometry to the content of the *geom_file*. The *geom_file* contains one specification line. Each line is delimited by a new-line character (`\n`). If the first character of a line is an asterisk (*), the line is treated as a comment. Each line is composed of entries that are position-dependent, are separated by white space, and have the following format:

pcyl ncyl acyl bcyl nheads nsectors sectsiz

where the entries have the following values:

pcyl This is the number of physical cylinders for the drive.

ncyl This is the number of usable cylinders for the drive.

acyl This is the number of alt cylinders for the drive.

bcyl This is the number of offset cylinders for the drive (should be zero).

nheads The number of heads for this drive.

nsectors The number of sectors per track.

sectsiz The size in bytes of a sector.

`-t`
Adjust incorrect slice table entries so that they will not cross partition table boundaries.

`-T`
Remove incorrect slice table entries that span partition table boundaries.

`-v`
Output the HBA (virtual) geometry dimensions. This option must be used in conjunction with the `-W` flag. This option will work for platforms which support virtual geometry. (x86 only)

`-w`
Write to disk and read from stdin. See `-o` and `-s`, which specify the starting point and size of the operation.

`-W –`
Output the disk table to `stdout`.

`-W` *fdisk_file*
Create an `fdisk` file *fdisk_file* from disk table. This can be used with the `-F` option below.

Files

`/dev/rdsk/c0t0d0p0`	Raw device associated with the fixed disk.
`/usr/lib/fs/ufs/mboot`	Default master boot program.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86 and SPARC
Availability	SUNWcsu

See Also `uname(1)`, `fmthard(1M)`, `format(1M)`, `newfs(1M)`, `prtvtoc(1M)`, `attributes(5)`

Diagnostics Most messages will be self-explanatory. The following may appear immediately after starting the program:

`Fdisk: cannot open <device>`
This indicates that the device name argument is not valid.

`Fdisk: unable to get device parameters for device <device>`
This indicates a problem with the configuration of the fixed disk, or an error in the fixed disk driver.

`Fdisk: error reading partition table`
This indicates that some error occurred when trying initially to read the fixed disk. This could be a problem with the fixed disk controller or driver, or with the configuration of the fixed disk.

`Fdisk: error writing boot record`
This indicates that some error occurred when trying to write the new partition table out to the fixed disk. This could be a problem with the fixed disk controller, the disk itself, the driver, or the configuration of the fixed disk.

Name ff – list file names and statistics for a file system

Synopsis `ff [-F` *FSType*`] [-V] [`*generic_options*`] [-o` *specific_options*`]` *special*`...`

Description `ff` prints the pathnames and inode numbers of files in the file system which resides on the special device *special*. Other information about the files may be printed using options described below. Selection criteria may be used to instruct `ff` to only print information for certain files. If no selection criteria are specified, information for all files considered will be printed (the default); the `-i` option may be used to limit files to those whose inodes are specified.

Output is sorted in ascending inode number order. The default line produced by `ff` is:

path-name i-number

The maximum information the command will provide is:

path-name i-number size uid

Options

`-F` Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table, or by consulting `/etc/default/fs`.

`-V` Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option may be used to verify and validate the command line.

generic_options Options that are supported by most FSType-specific modules of the command. The following options are available:

`-I` Do not print the i-node number after each path name.

`-l` Generate a supplementary list of all path names for multiply-linked files.

`-p` *prefix* The specified *prefix* will be added to each generated path name. The default is '.' (dot).

`-s` Print the file size, in bytes, after each path name.

`-u` Print the owner's login name after each path name.

`-a -`*n* Select if the file has been accessed in *n* days.

`-m -`*n* Select if the file has been written or created in *n* days.

`-c -`*n* Select if file's status has been changed in *n* days.

`-n` *file* Select if the file has been modified more recently than the argument `file`.

`-i` *i-node-list* Generate names for only those i-nodes specified in *i-node-list*. *i-node-list* is a list of numbers separated by commas (with no intervening spaces).

`-o` Specify *FSType*-specific options in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the *FSType*-specific module of the command.

Operands *special* A special device.

Usage See `largefile(5)` for the description of the behavior of `ff` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files `/etc/default/fs` default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`

`LOCAL` The default partition for a command if no FSType is specified.

`/etc/vfstab` list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `find(1)`, `ncheck(1M)`, `stat(2)`, `vfstab(4)`, `attributes(5)`, `largefile(5)` Manual pages for the *FSType*-specific modules of `ff`.

Notes This command may not be supported for all *FSTypes*.

The `-a`, `-m`, and `-c` flags examine the `st_atime`, `st_mtime`, and `st_ctime` fields of the `stat` structure respectively. (See `stat(2)`.)

Name ffbconfig, SUNWffb_config – configure the FFB Graphics Accelerator

Synopsis

```
/usr/sbin/ffbconfig [-dev device-filename]
     [-res video-mode [now | try] [noconfirm | nocheck]]
     [-file | machine | system]
     [-deflinear | true | false]
     [-defoverlay | true | false]
     [-linearorder | first | last]
     [-overlayorder | first | last]
     [-expvis | enable | disable]
     [-sov | enable | disable] [-maxwids n]
     [-extovl | enable | disable]
     [-g gamma-correction-value]
     [-gfile gamma-correction-file] [-propt] [-prconf]
     [-defaults]

/usr/sbin/ffbconfig [-propt ] [-prconf]

/usr/sbin/ffbconfig [-help] [-res ?]
```

Description `ffbconfig` configures the FFB Graphics Accelerator and some of the X11 window system defaults for FFB.

The first form of `ffbconfig` stores the specified options in the OWconfig file. These options will be used to initialize the FFB device the next time the window system is run on that device. Updating options in the OWconfig file provides persistence of these options across window system sessions and system reboots.

The second and third forms of `ffbconfig`, which invoke only the `-prconf`, `-propt`, `-help`, and `-res ?` options do not update the OWconfig file. Additionally, for the third form all other options are ignored.

Options may be specified for only one FFB device at a time. Specifying options for multiple FFB devices requires multiple invocations of `ffbconfig`.

Only FFB-specific options can be specified through `ffbconfig`. The normal window system options for specifying default depth, default visual class and so forth are still specified as device modifiers on the `openwin` command line. See the *OpenWindows Desktop Reference Manual* for details.

The user can also specify the OWconfig file that is to be updated. By default, the machine-specific file in the `/etc/openwin` directory tree is updated. The `-file` option can be used to specify an alternate file to use. For example, the system-global OWconfig file in the `/usr/openwin` directory tree can be updated instead.

Both of these standard OWconfig files can only be written by root. Consequently, the `ffbconfig` program, which is owned by the root user, always runs with setuid root permission.

Options `-dev` *device-filename*
Specifies the FFB special file. The default is `/dev/fbs/ffb0`.

`-file machine|system`
Specifies which `OWconfig` file to update. If *machine* is specified, the machine-specific `OWconfig` file in the `/etc/openwin` directory tree is updated. If *system* is specified, the global `OWconfig` file in the `/usr/openwin` directory tree is updated. If the specified file does not exist, it is created. This option has no effect unless other options are specified. The default is *machine*.

`-res` *video-mode* [`now` | `try` [`noconfirm` | `nocheck`]]
Specifies the video mode used to drive the monitor connected to the specified FFB device.

video-mode has the format of *width*x*height*x*rate* where *width* is the screen width in pixels, *height* is the screen height in pixels, and *rate* is the vertical frequency of the screen refresh.

The `s` suffix, as in `960x680x112s` and `960x680x108s`, indicates stereo video modes. The `i` suffix, as in `640x480x60i` and `768x575x50i`, indicates interlaced video timing. If absent, non-interlaced timing will be used.

`-res` (the third form in the `SYNOPSIS`) also accepts formats with `@` (at sign) in front of the refresh rate instead of `x`. `1280x1024@76` is an example of this format.

Some video-modes are supported only on certain revisions of FFB. Also, some video-modes, supported by FFB, may not be supported by the monitor. The list of video-modes supported by the FFB device and the monitor can be obtained by running `ffbconfig` with the `-res ?` option.

The following table lists all possible video modes supported on FFB:

Name	Description
`1024x768x60`	
`1024x768x70`	
`1024x768x75`	
`1024x768x77`	
`1024x800x84`	
`1152x900x66`	
`1152x900x76`	
`1280x800x76`	
`1280x1024x60`	
`1280x1024x67`	

Name	Description
1280x1024x76	
960x680x112s	(stereo)
960x680x108s	(stereo)
640x480x60	
640x480x60i	(interlaced)
768x575x50i	(interlaced)
1440x900x76	(hi-res)
1600x1000x66	(hi-res)
1600x1000x76i	(hi-res)
1600x1280x76	(hi-res)
1920x1080x72	(hi-res)
1920x1200x70	(hi-res)

Symbolic names

For convenience, some video modes have symbolic names defined for them. Instead of the form *width*x*height*x*rate,* one of these names may be supplied as the argument to `-res`. The meaning of the symbolic name `none` is that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

Name	Corresponding Video Mode
svga	1024x768x60
1152	1152x900x76
1280	1280x1024x76
stereo	960x680x112s
ntsc	640x480x60i
pal	768x575x50i
none	(video mode currently programmed in device)

The `-res` option also accepts additional, optional arguments immediately following the video mode specification. Any or all of these may be present.

now
Specifies that the FFB device will be immediately programmed to display this video mode, in addition to updating the video mode in the OWconfig file. This option is useful for changing the video mode before starting the window system.

It is inadvisable to use this suboption with ffbconfig while the configured device is being used (for example, while running the window system); unpredictable results may occur. To run ffbconfig with the now suboption, first bring the window system down. If the now suboption is used within a window system session, the video mode will be changed immediately, but the width and height of the affected screen won't change until the window system is exited and re-entered. In addition, the system may not recognize changes in stereo mode. Consequently, this usage is strongly discouraged.

noconfirm
Instructs ffbconfig to bypass confirmation and and warning messages and to program the requested video mode anyway.

Using the -res option, the user could potentially put the system into an unusable state, a state where there is no video output. This can happen if there is ambiguity in the monitor sense codes for the particular code read. To reduce the chance of this, the default behavior of ffbconfig is to print a warning message to this effect and to prompt the user to find out if it is okay to continue. This option is useful when ffbconfig is being run from a shell script.

nocheck
Suspends normal error checking based on the monitor sense code. The video mode specified by the user will be accepted regardless of whether it is appropriate for the currently attached monitor. This option is useful if a different monitor is to be connected to the FFB device. Note: Use of this option implies noconfirm as well.

try
Programs the specified video mode on a trial basis. The user will be asked to confirm the video mode by typing y within 10 seconds. The user may also terminate the trial before 10 seconds are up by typing any character. Any character other than y or RETURN is considered a no and the previous video mode will be restored and ffbconfig will not change the video mode in the OWconfig file and other options specified will still take effect. If a RETURN is pressed, the user is prompted for a yes or no answer on whether to keep the new video mode.

This sub-option should not be used with ffbconfig while the configured device is being used (for example, while running the window system) as unpredictable results may occur. To run fbconfig with the try sub-option, the window system should be brought down first.

`-deflinear true | false`
FFB possesses two types of visuals: linear and nonlinear. Linear visuals are gamma corrected and nonlinear visuals are not. There are two visuals that have both linear and nonlinear versions: 24-bit TrueColor and 8-bit StaticGray.

`-deflinear true` sets the default visual to the linear visual that satisfies other specified default visual selection options. Specifically, the default visual selection options are those set by the `Xsun` (1) `defdepth` and `defclass` options. See *OpenWindows Desktop Reference Manual* for details.

`-deflinear false` (or if there is no linear visual that satisfies the other default visual selection options) sets the default visual to the non-linear visual as the default.

This option cannot be used when the `-defoverlay` option is present, because FFB does not possess a linear overlay visual.

`-defoverlay true | false`
FFB provides an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the FFB visuals. This is called the overlay visual. Windows created in this visual will not damage windows created in other visuals. The converse, however, is not true. Windows created in other visuals will damage overlay windows. This visual has `256` `maxwids` of opaque color values. See `-maxwids` in `OPTIONS`.

If `-defoverlay` is `true`, the overlay visual will be made the default visual. If `-defoverlay` is `false`, the nonoverlay visual that satisfies the other default visual selection options, such as `defdepth` and `defclass`, will be chosen as the default visual. See the *OpenWindows Desktop Reference Manual* for details.

Whenever `-defoverlay true` is used, the default depth and class chosen on the openwin command line must be 8-bit PseudoColor. If not, a warning message will be printed and the `-defoverlay` option will be treated as false. This option cannot be used when the `-deflinear` option is present, because FFB doesn't possess a linear overlay visual.

`-linearorder first | last`
If `first`, linear visuals will come before their non-linear counterparts on the X11 screen visual list for the FFB screen. If `last`, the nonlinear visuals will come before the linear ones.

`-overlayorder first | last`
If `true`, the depth 8 PseudoColor Overlay visual will come before the non-overlay visual on the X11 screen visual list for the FFB screen. If `false`, the non-overlay visual will come before the overlay one.

`-expvis enable | disable`
If enabled, OpenGL Visual Expansion will be activated. Multiple instances of selected visual groups (8-bit PseudoColor, 24-bit TrueColor and so forth) can be found in the screen visual list.

`-sov enable | disable`
Advertises the root window's `SERVER_OVERLAY_VISUALS` property. SOV visuals will be exported and their transparent types, values and layers can be retrieved through this property. If `-sov disable` is specified, the `SERVER_OVERLAY_VISUALS` property will not be defined. SOV visuals will not be exported.

`-maxwids` *n*
Specifies the maximum number of FFB X channel pixel values that are reserved for use as window sIDs (WIDs). The remainder of the pixel values in overlay colormaps are used for normal X11 opaque color pixels. The reserved WIDs are allocated on a first-come first-serve basis by 3D graphics windows (such as XGL), MBX windows, and windows that have a non-default visual. The X channel codes `0` to (`255`-*n*) will be opaque color pixels. The X channel codes (`255`-*n*+`1`) to `255` will be reserved for use as WIDs. Legal values on FFB, FFB2 are: `1`, `2`, `4`, `8`, `16`, and `32`. Legal values on FFB2+ are: `1`, `2`, `4`, `8`, `16`, `32`, and `64`.

`-extovl enable | disable`
This option is available only on FFB2+. If enabled, extended overlay is available. The overlay visuals will have 256 opaque colors. The SOV visuals will have 255 opaque colors and 1 transparent color. This option enables hardware supported transparency which provides better performance for windows using the SOV visuals.

`-g` *gamma-correction value*
This option is available only on FFB2+. This option allows changing the gamma correction value. All linear visuals provide gamma correction. By default the gamma correction value is 2.22. Any value less than zero is illegal. The gamma correction value is applied to the linear visual, which then has an effective gamma value of 1.0, which is the value returned by `XSolarisGetVisualGamma`(3). See `XSolarisGetVisualGamma`(3) for a description of that function.

This option can be used while the window system is running. Changing the gamma correction value will affect all the windows being displayed using the linear visuals.

`-gfile` *gamma-correction file*
This option is available only on FFB2+. This option loads gamma correction table from the specified file. This file should be formatted to provide the gamma correction values for R, G and B channels on each line. This file should provide 256 triplet values, each in hexadecimal format and separated by at least 1 space. Following is an example of this file:

```
0x00 0x00 0x00
0x01 0x01 0x01
0x02 0x02 0x02
...
...
0xff 0xff 0xff
```

Using this option, the gamma correction table can be loaded while the window system is running. The new gamma correction will affect all the windows being displayed using the linear visuals. Note, when gamma correction is being done using user specified table, the

gamma correction value is undefined. By default, the window system assumes a gamma correction value of 2.22 and loads the gamma table it creates corresponding to this value.

`-defaults`
Resets all option values to their default values.

`-propt`
Prints the current values of all FFB options in the OWconfig file specified by the `-file` option for the device specified by the `-dev` option. Prints the values of options as they will be in the OWconfig file after the call to `ffbconfig` completes. The following is a typical display using the `-propt` option:

```
--- OpenWindows Configuration for /dev/fbs/ffb0 ---
OWconfig: machine
Video Mode: NONE
Default Visual: Non-Linear Normal Visual
Visual Ordering: Linear Visuals are last
                 Overlay Visuals are last
OpenGL Visuals: disabled
SOV: disabled
Allocated WIDs: 32
```

`-prconf`
Prints the FFB hardware configuration. The following is a typical display using the `-prconf` option:

```
--- Hardware Configuration for /dev/fbs/ffb0 ---
Type: double-buffered FFB2 with Z-buffer
Board: rev x
PROM Information: @(#)ffb2.fth x.x xx/xx/xx
FBC: version x
DAC: Brooktree 9068, version x
3DRAM: Mitsubishi 1309, version x
EDID Data: Available - EDID version 1 revision x
Monitor Sense ID: 4  (Sun 37x29cm RGB color monitor)
Monitor possible resolutions: 1024x768x60, 1024x768x70,
      1024x768x75, 1152x900x66, 1152x900x76,
      1280x1024x67, 1280x1024x76,
      960x680x112s, 640x480x60
Current resolution setting: 1280x1024x76
```

`-help`
Prints a list of the `ffbconfig` command line options, along with a brief explanation of each.

Defaults For a given invocation of `ffbconfig` command line if an option does not appear on the command line, the corresponding OWconfig option is not updated; it retains its previous value.

When the window system is run, if an FFB option has never been specified via `ffbconfig`, a default value is used. The option defaults are listed in the following table:

Option	Default
-dev	/dev/fbs/ffb0
-file	machine
-res	none
-deflinear	false
-defoverlay	false
-linearorder	last
-overlayorder	last
-expvis	enabled
-sov	enabled
-maxwids	32

The default for the -res option of none means that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

This provides compatibility for users who are used to specifying the device resolution through the PROM. On some devices (for example, GX) this is the only way of specifying the video mode. This means that the PROM ultimately determines the default FFB video mode.

Examples

EXAMPLE 1 Changing The Monitor Type

The following example switches the monitor type to the resolution of 1280 × 1024 at 76 Hz:

```
example% /usr/sbin/ffbconfig -res 1280x1024x76
```

Files

/dev/fbs/ffb0 device special file

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWffbcf

See Also

mmap(2), attributes(5), fbio(7I), ffb(7D)

OpenWindows Desktop Reference Manual

Name ff_ufs – list file names and statistics for a ufs file system

Synopsis

```
ff -F ufs [generic_options] [-o a,m,s] special...
```

Description `ff` prints the pathnames and inode numbers of files in the file system which resides on the special device *special*.

See `ff(1M)` for information regarding the `ff` command. See `OPTIONS` for information regarding the ufs-specific options.

Options The following options are supported:

`-o` Specify `ufs` file system specific options. The following options available are:

`a` Print the '.' and '. .' directory entries.

`m` Print mode information. This option must be specified in conjunction with the `-i` *i-node-list* option (see `ff(1M)`).

`s` Print only special files and files with set-user-ID mode.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `find(1)`, `ff(1M)`, `ncheck(1M)`, `attributes(5)`

Name fiocompress – file compression utility

Synopsis

```
/sbin/fiocompress -c [-m] [-b block_size] input_file output_file

/sbin/fiocompress -d input_file output_file
```

Description The `fiocompress` utility is a file compression tool that works together with the `dcfs`(7FS) file system to perform per-file compression. You can use `fiocompress` to decompress a compressed file or mark a compressed file as compressed, causing automatic decompression on read. The primary use of `fiocompress` is to compress files in the boot archive.

Note that this utility is not a Committed interface. See `attributes`(5).

Options The following options are supported:

`-b` *block_size*
Specify a block size for compression. The default block size is 8192.

`-c`
Compress the specified file.

`-d`
Decompress the specified file.

`-m`
Mark the compressed file for automatic decompression on read. Can be used only in conjunction with `-c`.

Exit Status

`0`
The command completed successfully.

`–1`
The command exited due to an error.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr
Interface Stability	Private

See Also `boot`(1M), `bootadm`(1M), `dcfs`(7FS), `ufs`(7FS), `attributes`(5)

Notes This compression/decompression utility works only with files stored in a UFS file system.

There is no obvious way to determine whether a given file is compressed, other than copying the file and comparing the number of disk blocks of the copy against the original.

Name flar – administer flash archives

Synopsis

```
flar create -n name [-R root] [-A system_image] [-H] [-I]
     [-M] [-S] [-c] [-t [-p posn] [-b blocksize]]
     [-i date] [-u section]... [-m master]
     [-f [filelist | -] [-F]] [-a author]  [-D dataset]
     [-e descr | -E descr_file] [-L archiver] [-T type]
     [-U key=value]... [-x exclude]... [-y include]...
     [-z filelist]... [-X filelist]... archive

flar combine [-d dir] [-u section]...
     [-t [-p posn] [-b blocksize]] archive

flar split [-d dir] [-u section]... [-f] [-S section]
     [-t [-p posn] [-b blocksize]] archive

flar info [-l] [-k keyword] [-t [-p posn] [-b blocksize]] archive
```

Description The `flar` command is used to administer flash archives. A flash archive is an easily transportable version of a reference configuration of the Solaris operating environment, plus optional other software. Such an archive is used for the rapid installation of Solaris on large numbers of machines. You can create a flash archive using either `flar` with the `create` subcommand or the `flarcreate(1M)` command. See `flash_archive(4)`.

In flash terminology, a system on which an archive is created is called a *master*. The system image stored in the archive is deployed to systems that are called *clones*.

A flash archive can be created on a system that is running a UFS root file system or a ZFS root file system. A flash archive of a ZFS root pool contains the entire pool hierarchy except for the swap and dump volumes and any excluded datasets. The swap and dump volumes are created when the flash archive is installed.

There are two types of flash archives: full and differential. Both are created with the `create` subcommand. A full archive contains all the files that are in a system image. A differential archive contains only differences between two system images. Installation of a differential archive is faster and consumes fewer resources than installation of a full archive.

In creating a differential archive, you compare two system images. A system image can be any of:

- a Live Upgrade boot environment, mounted on some directory using `lumount(1M)` (see `live_upgrade(5)`)
- a clone system mounted over NFS with root permissions
- a full flash archive expanded into some local directory

To explain the creation of a differential flash archive, the following terminology is used:

old The image prior to upgrade or other modification. This is likely the image as it was installed on clone systems.

new The old image, plus possible additions or changes and minus possible deletions. This is likely the image you want to duplicate on clone systems.

The `flar` command compares *old* and *new*, creating a differential archive as follows:

- files on *new* that are not in *old* are added to the archive;
- files of the same name that are different between *old* and *new* are taken from *new* and added to the archive;
- files that are in *old* and not in *new* are put in list of files to be deleted when the differential archive is installed on clone systems.

When creating a differential flash archive, the currently running image is, by default, the new image and a second image, specified with the `-A` option, is the old image. You can use the `-R` option to designate an image other than the currently running system as the new image. These options are described below.

Note that differential flash archives are not supported for a ZFS root file system.

You can run `flarcreate` in multi- or single-user mode. You can also use the command when the master system is booted from the first Solaris software CD or from a Solaris net image. Archive creation should be performed when the master system is in as stable a state as possible.

Following creation of a flash archive, you can use JumpStart to clone the archive on multiple systems. Using JumpStart to install one or more systems is the only installation method available for installing a flash archive of a ZFS root pool.

Subcommands The `flar` command includes subcommands for creating, combining, splitting, and providing information about archives. A subcommands is the first argument in a `flar` command line. These subcommands are as follows:

`create` Create a new flash archive, of a name you specify with the `-n` argument, based on the currently running system. Use the `-A` option (described below) to create a differential flash archive.

`combine` Combine the individual sections that make up an archive into the archive. If *dir* is specified (see `-d` option below), the sections will be gathered from *dir*; otherwise, they will be gathered from the current directory. Each section is assumed to be in a separate file, the names of which are the section names. At a minimum, the archive cookie (`cookie`), archive identification (`identification`), and archive files (`archive`) sections must be present. If `archive` is a directory, its contents are archived using `files_archived_method` (`cpio` and `pax`) prior to inclusion in the archive. If so specified in the `identification` section, the contents are compressed.

Note that no validation is performed on any of the sections. In particular, no fields in the `identification` section are validated or updated. See `flash_archive(4)` for a description of the archive sections.

`info` Extract information on an archive. This subcommand is analogous to `pkginfo`.

`split` Split an archive into one file for each section of the archive. Each section is copied into a separate file in *dir*, if *dir* is specified (see `-d` option below), or the current directory if it is not. The files resulting from the split are named after the sections. The archive cookie is stored in a file named `cookie`. If *section* is specified (see `-u` option below), only the named section is copied.

The `create` subcommand requires root privileges.

The options for each subcommand are described below.

Options The options for the `create` subcommand are below. Many of these options supply values for keywords in the identification section of a file containing a flash archive. See `flash_archive(4)` for a description of these keywords.

`-a` *author* *author* is used to provide an author name for the archive identification section of the new flash archive. If you do not specify `-a`, no author name is included in the identification section.

`-A` *system_image* Create a differential flash archive by comparing a new system image (see DESCRIPTION) with the image specified by the *system_image* argument. By default, the new system image is the currently running system. You can change the default with the `-R` option, described below. *system_image* is a directory containing an image. It can be accessible through UFS, NFS, or `lumount(1M)`.

The rules for inclusion and exclusion of files in a differential archive are described in DESCRIPTION. You can modify the effect of these rules with the use of the `-x`, `-X`, `-y`, and `-z` options, described below.

This option is not supported when creating a flash archive of a ZFS root pool.

`-c` Compress the archive using `compress(1)`

`-D` *dataset* Exclude specified datasets from the flash archive. This option can be used multiple times to exclude multiple datasets.

`-f` *filelist* Use the contents of *filelist* as a list of files to include in the archive. The files are included in addition to the normal file list, unless `-F` is specified (see below). If *filelist* is `-`, the list is taken from standard input. *filelist* can include directories. If a directory is listed, all files and subdirectories under that directory are included.

This option is not supported when creating a flash archive of a ZFS root pool.

`-e` *descr*
: The description to be included in the archive as the value of the `content_description` archive identification key. This option is incompatible with `-E`.

`-E` *descr_file*
: The description to be used as the value of the archive identification `content_description` key is retrieved from the file *descr_file*. This option is incompatible with `-e`.

`-F`
: Include only files in the list specified by `-f`. This option makes `-f` *filelist* an absolute list, rather than a list that is appended to the normal file list.

  This option is not supported when creating a flash archive of a ZFS root pool.

`-H`
: Do not generate hash identifier.

`-I`
: Ignore integrity check. To prevent you from excluding important system files from an archive, `flar` runs an integrity check. This check examines all files registered in a system package database and stops archive creation if any of them are excluded. Use this option to override this integrity check.

  This option is not supported when creating a flash archive of a ZFS root pool.

`-i` *date*
: By default, the value for the `creation_date` field in the identification section is generated automatically, based on the current system time and date. If you specify the `-i` option, *date* is used instead. *date* is in the format *YYYYMMDDhhmmss*, so that, for example:

  20060905031000

  ...stands for 3:10 AM on September 5, 2006.

`-L` *archiver*
: By default, the value for the `files_archived_method` field in the identification section is `pax(1)`. If you specify `-L`, the archiver (`cpio(1)` and `pax`) is used instead.

`-m` *master*
: By default, the value for the `creation_master` field in the identification section is the name of the system on which you run `flarcreate`, as reported by `uname -n`. If you specify `-m`, *master* is used instead.

`-M`
: Used only when you are creating a differential flash archive. When creating a differential archive, `flar` creates a long list of the files in the system that remain the same, are changed, and are to be deleted on clone systems. This list is stored in the manifest section of the archive (see

flash_archive(4)). When the differential archive is deployed, the flash software uses this list to perform a file-by-file check, ensuring the integrity of the clone system. Use of this option to avoids such a check and saves the space used by the manifest section in a differential archive. However, you must weigh the savings in time and disk space against the loss of an integrity check upon deployment. Because of this loss, use of this option is not recommended.

This option is not supported when creating a flash archive of a ZFS root pool.

`-n` *name*

name is supplied as the value of the `content_name` keyword. See flash_archive(4).

The `-n` *name* option, with *name* argument, is required for the `create` subcommand.

`-R` *root*

Create the archive from the file system tree rooted at *root*. If you do not specify this option, `flar` creates an archive from a file system rooted at `/`. When creating a differential flash archive, the system image specified by `-R` replaces the currently running system as the *new* image. See DESCRIPTION.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See zones(5).

`-S`

Skip the disk space check and do not write archive size data to the archive. Without `-S`, `flar` builds a compressed archive in memory before writing the archive to disk, to ensure you have sufficient disk space. Use `-S` to skip this step. The result of the use of `-S` is a significant decrease in the time it takes to create an archive.

`-T` *type*

Content type included in the archive as the value of the `content_type` archive identification key. If you do not specify `-T`, the `content_type` keyword is not included.

`-U` *key=value...*

Include the user-defined keyword(s) and values in the archive identification section. See flash_archive(4).

`-u` *section...*

Include the user-defined section located in the file *section* in the archive. *section* must be a blank-separated list of section names as described in flash_archive(4).

`-x` *exclude ...*

Exclude the file or directory *exclude* from the archive. Note that the *exclude* file or directory is assumed to be relative to the alternate root specified using `-R`. If the parent directory of the file *exclude* is included

with the `-y` option (see `-y` *include*), then only the specific file or directory specified by *exclude* is excluded. Conversely, if the parent directory of an included file is specified for exclusion, then only the file *include* is included. For example, if you specify:

```
-x /a -y /a/b
```

all of `/a` except for `/a/b` is excluded. If you specify:

```
-y /a -x /a/b
```

all of `/a` except for `/a/b` is included.

This option is not supported when creating a flash archive of a ZFS root pool.

`-y` *include* ... Include the file or directory *include* in the archive. Note that the *exclude* file or directory is assumed to be relative to the alternate root specified using `-R`. See the description of the `-x` option, above, for a description of the interaction of the `-x` and `-y` options.

This option is not supported when creating a flash archive of a ZFS root pool.

`-X` *filelist* ... Use the contents of *filelist* as a list of files to exclude from the archive. If *filelist* is –, the list is taken from standard input.

This option is not supported when creating a flash archive of a ZFS root pool.

`-z` *filelist* ... *filelist* is a list of files prefixed with a plus (+) or minus (-). A plus indicates that a file should be included in the archive; the minus indicates exclusion. If *filelist* is –, the list is taken from standard input.

This option is not supported when creating a flash archive of a ZFS root pool.

The options for `flar info` subcommand are as follows:

`-k` *keyword* Only the value of the keyword *keyword* is returned.

`-l` List all files in the archive. Does not process content from any sections other than the `archive` section.

The following are `flar info` options used with tape archives:

`-b` *blocksize* The block size to be used when creating the archive. If not specified, a default block size of 64K is used.

`-p` *posn* Specifies the position on the tape device where the archive should be created. If not specified, the current position of the tape device is examined.

`-t` The archive to be analyzed is located on a tape device. The path to the device is specified by *archive* (see OPERANDS).

The options for `flar split` and `combine` (split and combine archives) subcommands are as follows:

`-d` *dir* Retrieve sections from *dir*, rather than from the current directory.

`-f` (Used with `split` only.) Extract the `archive` section into directory called `archive`, rather than placing it in a file of the same name as the section.

`-S` *section* (Used with `split` only.) Extract only the section named *section* from the archive.

`-u` *section...* Appends *section* to the list of sections to be included. The default list includes the `cookie`, `identification`, and `archive` sections. *section* can be a single section name or a space-separated list of section names.

The following options are used with tape archives (with both `split` and `combine`):

`-b` *blocksize* The block size to be used when creating the archive. If not specified, a default block size of 64K is used.

`-p` *posn* Used only with `-t`. Specifies the position on the tape device where the archive should be created. If not specified, the current position of the tape device is used.

`-t` Create an archive on or read an archive from a tape device. The *archive* operand (see OPERANDS) is assumed to be the name of the tape device.

Operands The following operand is supported:

archive Path to tape device if the `-t` option was used. Otherwise, the complete path name of a flash archive. By convention, a file containing a flash archive has a file extension of `.flar`.

Examples **EXAMPLE 1** Creating a Flash Archive

The command below creates a flash archive named `pogoS9` and stores it in `/export/home/archives/s9fcs.flar`. The currently running system is the basis for the new archive.

```
# flar create -n pogoS9 /export/home/archives/s9fcs.flar
```

EXAMPLE 2 Creating Differential Flash Archives

The command below creates a differential flash archive.

EXAMPLE 2 Creating Differential Flash Archives *(Continued)*

```
# flar create -n diff_pogoS9 -A /images \
/export/home/archives/diff_s9fcs.flar
```

In the following example the *old* system image is accessed through `lumount`.

```
# lumount s9BE /test
# flar create -n diff_pogoS9 -A /test \
     /export/home/archives/diff_s9fcs.flar
```

The following example shows the use of the `-R` option to specify a new system image other than the currently running system.

```
# flar create -n diff_pogoS9 -R /test \
   -A /images /export/home/archives/diff_s9fcs.flar
```

Note the caveat on the use of the `-R` option in the description of that option, above.

Exit Status The following exit values are returned for the `create`, `split`, and `combine` subcommands:

`0` Successful completion.

`>0` An error occurred.

The following exit values are returned for the `info` subcommand:

`0` Successful completion.

`1` Command failed. If the `-k` option is used and the requested keyword is not found, `flar` returns 2.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWinst
Stability Level	Evolving

See Also `cpio(1)`, `pax(1)`, `flarcreate(1M)`, `flash_archive(4)`, `attributes(5)`

Name flarcreate – create a flash archive from a master system

Synopsis

```
flarcreate -n name [-R root] [-A system_image] [-H] [-I]
     [-M] [-S] [-c] [-t [-p posn] [-b blocksize]]
     [-i date] [-u section]... [-m master]
     [-f [filelist | -] [-F]] [-a author]
     [-e descr | -E descr_file] [-L archiver] [-D dataset]
     [-T type] [-U key=value]... [-x exclude]...
     [-y include]... [-z filelist]... [-X filelist]... archive
```

Description The `flarcreate` command creates a flash archive from a master system. A master system is one that contains a reference configuration, which is a particular configuration of the Solaris operating environment, plus optional other software. A flash archive is an easily transportable version of the reference configuration.

In flash terminology, a system on which an archive is created is called a *master*. The system image stored in the archive is deployed to systems that are called *clones*.

A flash archive can be created on a system that is running a UFS root file system or a ZFS root file system. A flash archive of a ZFS root pool contains the entire pool hierarchy except for the swap and dump volumes and any excluded datasets. The swap and dump volumes are created when the flash archive is installed.

There are two types of flash archives: full and differential. A full archive contains all the files that are in a system image. A differential archive contains only differences between two system images. Installation of a differential archive is faster and consumes fewer resources than installation of a full archive.

In creating a differential archive, you compare two system images. A system image can be any of:

- a Live Upgrade boot environment, mounted on some directory using `lumount(1M)` (see `live_upgrade(5)`)
- a clone system mounted over NFS with root permissions
- a full flash archive expanded into some local directory

To explain the creation of a differential flash archive, the following terminology is used:

old
: The image prior to upgrade or other modification. This is likely the image as it was installed on clone systems.

new
: The old image, plus possible additions or changes and minus possible deletions. This is likely the image you want to duplicate on clone systems.

The `flarcreate` command compares *old* and *new*, creating a differential archive as follows:

- files on *new* that are not in *old* are added to the archive;

- files of the same name that are different between *old* and *new* are taken from *new* and added to the archive;
- files that are in *old* and not in *new* are put in list of files to be deleted when the differential archive is installed on clone systems.

When creating a differential flash archive, the currently running image is, by default, the new image and a second image, specified with the `-A` option, is the old image. You can use the `-R` option to designate an image other than the currently running system as the new image. These options are described below.

Note that differential flash archives are not supported for a ZFS root file system.

Following creation of a flash archive, you can use JumpStart to clone the archive on multiple systems. Using JumpStart to install one or more systems is the only installation method available for installing a flash archive of a ZFS root pool.

You can run `flarcreate` in multi- or single-user mode. You can also use the command when the master system is booted from the first Solaris software CD or from a Solaris net image.

Archive creation should be performed when the master system is in as stable a state as possible. Following archive creation, use the `flar(1M)` command to administer a flash archive.

See `flash_archive(4)` for a description of the flash archive.

The `flarcreate` command requires root privileges.

Options The `flarcreate` command has one required argument:

`-n` *name*
Specifies the name of the flash archive. *name* is supplied as the value of the `content_name` keyword. See `flash_archive(4)`.

The `flarcreate` command has the following general options:

`-A` *system_image*
Create a differential flash archive by comparing a new system image (see DESCRIPTION) with the image specified by the *system_image* argument. By default, the new system image is the currently running system. You can change the default with the `-R` option, described below. *system_image* is a directory containing an image. It can be accessible through UFS, NFS, or `lumount(1M)`.

The rules for inclusion and exclusion of files in a differential archive are described in DESCRIPTION. You can modify the effect of these rules with the use of the `-x`, `-X`, `-y`, and `-z` options, described below.

This option is not supported when creating a flash archive of a ZFS root pool.

`-c`
Compress the archive using `compress(1)`

`-D` *dataset*
Exclude specified datasets from the flash archive. This option can be used multiple times to exclude multiple datasets.

`-f` *filelist*
Use the contents of *filelist* as a list of files to include in the archive. The files are included in addition to the normal file list, unless `-F` is specified (see below). If *filelist* is `-`, the list is taken from standard input. *filelist* can include directories. If a directory is listed, all files and subdirectories under that directory are included.

This option is not supported when creating a flash archive of a ZFS root pool.

`-F`
Include only files in the list specified by `-f`. This option makes `-f` *filelist* an absolute list, rather than a list that is appended to the normal file list.

This option is not supported when creating a flash archive of a ZFS root pool.

`-H`
Do not generate hash identifier.

`-I`
Ignore integrity check. To prevent you from excluding important system files from an archive, `flarcreate` runs an integrity check. This check examines all files registered in a system package database and stops archive creation if any of them are excluded. Use this option to override this integrity check.

This option is not supported when creating a flash archive of a ZFS root pool.

`-L` *archiver*
By default, the value for the `files_archived_method` field in the identification section is `pax(1)`. If you specify `-L`, the archiver (`cpio(1)` and `pax`) is used instead.

`-M`
Used only when you are creating a differential flash archive. When creating a differential archive, `flarcreate` creates a long list of the files in the system that remain the same, are changed, and are to be deleted on clone systems. This list is stored in the manifest section of the archive (see `flash_archive(4)`). When the differential archive is deployed, the flash software uses this list to perform a file-by-file check, ensuring the integrity of the clone system. Use of this option to avoids such a check and saves the space used by the manifest section in a differential archive. However, you must weigh the savings in time and disk space against the loss of an integrity check upon deployment. Because of this loss, use of this option is not recommended.

This option is not supported when creating a flash archive of a ZFS root pool.

`-R` *root*
Create the archive from the file system tree rooted at *root*. If you do not specify this option, `flarcreate` creates an archive from a file system rooted at `/`.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-S`
Skip the disk space check and do not write archive size data to the archive. Without `-S`, `flarcreate` builds a compressed archive in memory before writing the archive to disk, to determine the size of the archive. This size information is written to the header of the archive in the `files_archived_size field` and is used during archive deployment on the client to ensure enough disk space is available on the client. Use `-S` to skip this step. The result of the use of `-S` is a significant decrease in the time it takes to create an archive.

`-U` *key=value...*
Include the user-defined keyword(s) and values in the archive identification section.

`-x` *exclude...*
Exclude the file or directory *exclude* from the archive. Note that the *exclude* file or directory is assumed to be relative to the alternate root specified using `-R`. If the parent directory of the file *exclude* is included with the `-y` option (see `-y` *include*), then only the specific file or directory specified by *exclude* is excluded. Conversely, if the parent directory of an included file is specified for exclusion, then only the file *include* is included. For example, if you specify:

```
-x /a -y /a/b
```

all of `/a` except for `/a/b` is excluded. If you specify:

```
-y /a -x /a/b
```

all of `/a` except for `/a/b` is included.

This option is not supported when creating a flash archive of a ZFS root pool.

`-X` *filelist...*
Use the contents of *filelist* as a list of files to exclude from the archive. If *filelist* is –, the list is taken from standard input.

This option is not supported when creating a flash archive of a ZFS root pool.

`-y` *include...*
Include the file or directory *include* in the archive. Note that the *include* file or directory is assumed to be relative to the alternate root specified using `-R`. See the description of the `-x` option, above, for a description of the interaction of the `-x` and `-y` options.

This option is not supported when creating a flash archive of a ZFS root pool.

`-z` *filelist...*
filelist is a list of files prefixed with a plus (`+`) or minus (`-`). A plus indicates that a file should be included in the archive; the minus indicates exclusion. If *filelist* is –, the list is taken from standard input.

This option is not supported when creating a flash archive of a ZFS root pool.

Use the following option with user-defined sections.

`-u` *section...*
Include the user-defined section located in the file *section* in the archive. *section* must be a blank-separated list of section names as described in `flash_archive(4)`.

Use the following options with tape archives.

`-b` *blocksize*
The block size to be used when creating the archive. If not specified, a default block size of 64K is used.

`-p` *posn*
Used only with `-t`. Specifies the position on the tape device where the archive should be created. If not specified, the current position of the tape device is used.

`-t`
Create an archive on a tape device. The *archive* operand (see OPERANDS) is assumed to be the name of the tape device.

The following options are used for archive identification.

`-a` *author*
author is used to provide an author name for the archive identification section. If you do not specify `-a`, no author name is included in the identification section.

`-e` *descr*
The description to be included in the archive as the value of the `content_description` archive identification key. This option is incompatible with `-E`.

`-E` *descr_file*
The description to be used as the value of the archive identification `content_description` key is retrieved from the file *descr_file*. This option is incompatible with `-e`.

`-i` *date*
By default, the value for the `creation_date` field in the identification section is generated automatically, based on the current system time and date. If you specify the `-i` option, *date* is used instead. *date* is in the format *YYYYMMDDhhmmss*, so that, for example:

20060905031000

...stands for 3:10 AM on September 5, 2006.

`-m` *master*
By default, the value for the `creation_master` field in the identification section is the name of the system on which you run `flarcreate`, as reported by `uname -n`. If you specify `-m`, *master* is used instead.

`-T` *type*
Content type included in the archive as the value of the `content_type` archive identification key. If you do not specify `-T`, the `content_type` keyword is not included.

Operands The following operand is supported:

archive
Path to tape device if the `-t` option was used. Otherwise, the complete path name of a flash archive. By convention, a file containing a flash archive has a file extension of `.flar`.

Exit Status The following exit values are returned:

`0`
Successful completion.

`>0`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWinst

See Also `cpio(1)`, `pax(1)`, `flar(1M)`, `flash_archive(4)`, `attributes(5)`

Name fmadm – fault management configuration tool

Synopsis `fmadm [-q] [`*subcommand* `[`*arguments*`]]`

Description The `fmadm` utility can be used by administrators and service personnel to view and modify system configuration parameters maintained by the Solaris Fault Manager, `fmd`(1M). `fmd` receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components.

`fmadm` can be used to:

- view the set of diagnosis engines and agents that are currently participating in fault management,
- view the list of system components that have been diagnosed as faulty, and
- perform administrative tasks related to these entities.

The Fault Manager attempts to automate as many activities as possible, so use of `fmadm` is typically not required. When the Fault Manager needs help from a human administrator, service repair technician, or Sun, it produces a message indicating its needs. It also refers you to a knowledge article on Sun's web site, http://www.sun.com/msg/. The web site might ask you to use `fmadm` or one of the other fault management utilities to gather more information or perform additional tasks. The documentation for `fmd`(1M), `fmdump`(1M), and `fmstat`(1M) describe more about tools to observe fault management activities.

The `fmadm` utility requires the user to possess the `SYS_CONFIG` privilege. Refer to the *System Administration Guide: Security Services* for more information about how to configure Solaris privileges. The `fmadm load` subcommand requires that the user possess all privileges.

SUBCOMMANDS `fmadm` accepts the following subcommands. Some of the subcommands accept or require additional options and operands:

`fmadm acquit` *fmri* | *label* [*uuid*]
Notify the Fault Manager that the specified resource is not to be considered to be a suspect in the fault event identified by *uuid*, or if no UUID is specified, then in any fault or faults that have been detected. The `fmadm acquit` subcommand should be used only at the direction of a documented Sun repair procedure. Administrators might need to apply additional commands to re-enable a previously faulted resource.

`fmadm acquit` *uuid*
Notify the Fault Manager that the fault event identified by *uuid* can be safely ignored. The `fmadm acquit` subcommand should be used only at the direction of a documented Sun repair procedure. Administrators might need to apply additional commands to re-enable any previously faulted resources.

`fmadm config`
Display the configuration of the Fault Manager itself, including the module name, version, and description of each component module. Fault Manager modules provide services such as automated diagnosis, self-healing, and messaging for hardware and software present on the system.

`fmadm faulty` [`-afgiprsv`] [`-n` *max*] [`-u` *uid*]
Display status information for resources that the Fault Manager currently believes to be faulty.

The following options are supported:

`-a` Display all faults. By default, the `fmadm faulty` command only lists output for resources that are currently present and faulty. If you specify the `-a` option, all resource information cached by the Fault Manager is listed, including faults which have been automatically corrected or where no recovery action is needed. The listing includes information for resources that might no longer be present in the system.

`-f` Display faulty `fru's` (Field replaceable units).

`-g` Group together faults which have the same fru, class and fault message.

`-i` Display persistent cache identifier for each resource in the Fault Manager.

`-n` *max* If faults or resources are grouped together with the `-a` or `-g` options, limit the output to *max* entries.

`-p` Pipe output through pager with form feed between each fault.

`-r` Display Fault Management Resource with their Identifier (FMRI) and their fault management state.

`-s` Display 1 line fault summary for each fault event.

`-u` *uid* Only display fault with given uid.

`-v` Display full output.

The percentage certainty is displayed if a fault has multiple suspects, either of different classes or on different `fru`'s. If more than one resource is on the same `fru` and it is not 100% certain that the fault is associated with the `fru`, the maximum percentage certainty of the possible suspects on the `fru` is displayed.

The Fault Manager associates the following states with every resource for which telemetry information has been received:

`ok` The resource is present and in use and has no known problems so far as the Fault Manager is concerned.

`unknown`	The resource is not present or not usable but has no known problems. This might indicate the resource has been disabled or deconfigured by an administrator. Consult appropriate management tools for more information.
`faulted`	The resource is present but is not usable because one or more problems have been diagnosed by the Fault Manager. The resource has been disabled to prevent further damage to the system.
`degraded`	The resource is present and usable, but one or more problems have been diagnosed in the resource by the Fault Manager. If all affected resources are in the same state, this is reflected in the message at the end of the list. Otherwise the state is given after each affected resource.
`fmadm flush` *fmri*	Flush the information cached by the Fault Manager for the specified resource, named by its FMRI. This subcommand should only be used when indicated by a documented Sun repair procedure. Typically, the use of this command is not necessary as the Fault Manager keeps its cache up-to-date automatically. If a faulty resource is flushed from the cache, administrators might need to apply additional commands to enable the specified resource.
`fmadm load` *path*	Load the specified Fault Manager module. *path* must be an absolute path and must refer to a module present in one of the defined directories for modules. Typically, the use of this command is not necessary as the Fault Manager loads modules automatically when Solaris initially boots or as needed.
`fmadm unload` *module*	Unload the specified Fault Manager module. Specify *module* using the basename listed in the `fmadm config` output. Typically, the use of this command is not necessary as the Fault Manager loads and unloads modules automatically based on the system configuration
`fmadm repaired` *fmri* | *label*	Notify the Fault Manager that a repair procedure has been carried out on the specified resource. The `fmadm repaired` subcommand should be used only at the direction of a documented Sun repair procedure. Administrators might need to apply additional commands to re-enable a previously faulted resource.

`fmadm replaced` *fmri* | *label*	Notify the Fault Manager that the specified resource has been replaced. This command should be used in those cases where the Fault Manager is unable to automatically detect the replacement. The `fmadm replaced` subcommand should be used only at the direction of a documented Sun repair procedure. Administrators might need to apply additional commands to re-enable a previously faulted resource.
`fmadm reset` [`-s` *serd*] *module*	Reset the specified Fault Manager module or module subcomponent. If the `-s` option is present, the specified Soft Error Rate Discrimination (SERD) engine is reset within the module. If the `-s` option is not present, the entire module is reset and all persistent state associated with the module is deleted. The `fmadm reset` subcommand should only be used at the direction of a documented Sun repair procedure. The use of this command is typically not necessary as the Fault Manager manages its modules automatically.

`fmadm rotate errlog | fltlog`	The `rotate` subcommand is a helper command for `logadm(1M)`, so that `logadm` can rotate live log files correctly. It is not intended to be invoked directly. Use one of the following commands to cause the appropriate logfile to be rotated, if the current one is not zero in size:
	`# logadm -p now -s 1b /var/fm/fmd/errlog`
	`# logadm -p now -s 1b /var/fm/fmd/fltlog`

Options The following options are supported:

`-q` Set quiet mode. `fmadm` does not produce messages indicating the result of successful operations to standard output.

Operands The following operands are supported:

cmd The name of a subcommand listed in `SUBCOMMANDS`.

args One or more options or arguments appropriate for the selected *subcommand*, as described in `SUBCOMMANDS`.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred. Errors include a failure to communicate with `fmd` or insufficient privileges to perform the requested operation.

`2` Invalid command-line options were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfmd
Interface Stability	See below.

The command-line options are Committed. The human-readable output is not-an-interface.

See Also `fmd(1M)`, `fmdump(1M)`, `fmstat(1M)`, `logadm(1M)`, `syslogd(1M)`, `attributes(5)`

System Administration Guide: Security Services

http://www.sun.com/msg/

Name fmd – fault manager daemon

Synopsis `/usr/lib/fm/fmd/fmd [-V] [-f` *file*`] [-o` *opt*`=`*val*`] [-R` *dir*`]`

Description `fmd` is a daemon that runs in the background on each Solaris system. `fmd` receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components. When appropriate, the fault manager also sends a message to the `syslogd(1M)` service to notify an administrator that a problem has been detected. The message directs administrators to a knowledge article on Sun's web site, http://www.sun.com/msg/, which explains more about the problem impact and appropriate responses.

Each problem diagnosed by the fault manager is assigned a Universal Unique Identifier (UUID). The UUID uniquely identifes this particular problem across any set of systems. The `fmdump(1M)` utility can be used to view the list of problems diagnosed by the fault manager, along with their UUIDs and knowledge article message identifiers. The `fmadm(1M)` utility can be used to view the resources on the system believed to be faulty. The `fmstat(1M)` utility can be used to report statistics kept by the fault manager. The fault manager is started automatically when Solaris boots, so it is not necessary to use the `fmd` command directly. Sun's web site explains more about what capabilities are currently available for the fault manager on Solaris.

Options The following options are supported

`-f` *file*	Read the specified configuration *file* prior to searching for any of the default fault manager configuration files.
`-o` *opt*`=`*value*	Set the specified fault manager option to the specified value. Fault manager options are currently a Private interface; see `attributes(5)` for information about Private interfaces.
`-R` *dir*	Use the specified root directory for all pathnames evaluated by the fault manager, instead of the default root (`/`).
`-V`	Print the fault manager's version to stdout and exit.

Exit Status The following exit values are returned:

`0`	Successful completion
`1`	An error occurred which prevented the fault manager from initializing, such as failure to open the telemetry transport.
`2`	Invalid command-line options were specified.

Files

`/etc/fm/fmd`	Fault manager configuration directory
`/usr/lib/fm/fmd`	Fault manager library directory
`/var/fm/fmd`	Fault manager log directory

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfmd
Interface Stability	Evolving

See Also svcs(1), fmadm(1M), fmdump(1M), fmstat(1M), syslogd(1M), attributes(5), smf(5)

http://www.sun.com/msg/

Notes The Fault Manager is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/fmd:default
```

The service's status can be queried using the svcs(1) command. Administrators should not disable the Fault Manager service.

Name fmdump – fault management log viewer

Synopsis

```
fmdump [-efmvV] [-c class] [-R dir] [-t time] [-T time]
     [-u uid] [-n name[.name]*[=value]] [file]
```

Description The `fmdump` utility can be used to display the contents of any of the log files associated with the Solaris Fault Manager, `fmd(1M)`. The Fault Manager runs in the background on each Solaris system. It receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components.

The Fault Manager maintains two sets of log files for use by administrators and service personnel:

error log — A log which records error telemetry, the symptoms of problems detected by the system.

fault log — A log which records fault diagnosis information, the problems believed to explain these symptoms.

By default, `fmdump` displays the contents of the fault log, which records the result of each diagnosis made by the fault manager or one of its component modules.

An example of a default `fmdump` display follows:

```
# fmdump
TIME                 UUID                                 SUNW-MSG-ID
Dec 28 13:01:27.3919 bf36f0ea-9e47-42b5-fc6f-c0d979c4c8f4 FMD-8000-11
Dec 28 13:01:49.3765 3a186292-3402-40ff-b5ae-810601be337d FMD-8000-11
Dec 28 13:02:59.4448 58107381-1985-48a4-b56f-91d8a617ad83 FMD-8000-OW
...
```

Each problem recorded in the fault log is identified by:

- The time of its diagnosis
- A Universal Unique Identifier (UUID) that can be used to uniquely identify this particular problem across any set of systems
- A message identifier that can be used to access a corresponding knowledge article located at Sun's web site, http://www.sun.com/msg/

If a problem requires action by a human administrator or service technician or affects system behavior, the Fault Manager also issues a human-readable message to `syslogd(1M)`. This message provides a summary of the problem and a reference to the knowledge article on the Sun web site, http://www.sun.com/msg/.

You can use the -v and -V options to expand the display from a single-line summary to increased levels of detail for each event recorded in the log. The -c, -t, -T, and -u options can be used to filter the output by selecting only those events that match the specified *class*, range of times, or *uuid*.

If more than one filter option is present on the command-line, the options combine to display only those events that are selected by the logical AND of the options. If more than one instance of the same filter option is present on the command-line, the like options combine to display any events selected by the logical OR of the options. For example, the command:

```
# fmdump -u uuid1 -u uuid2 -t 02Dec03
```

selects events whose attributes are (uuid1 OR uuid2) AND (time on or after 02Dec03).

Options The following options are supported:

-c *class*
: Select events that match the specified class. The class argument can use the glob pattern matching syntax described in sh(1). The class represents a hierarchical classification string indicating the type of telemetry event. More information about Sun's telemetry protocol is available at Sun's web site, http://www.sun.com/msg/.

-e
: Display events from the fault management error log instead of the fault log. This option is shorthand for specifying the pathname of the error log file.

  The error log file contains Private telemetry information used by Sun's automated diagnosis software. This information is recorded to facilitate post-mortem analysis of problems and event replay, and should not be parsed or relied upon for the development of scripts and other tools. See attributes(5) for information about Sun's rules for Private interfaces.

-f
: Follow the growth of the log file by waiting for additional data. fmdump enters an infinite loop where it will sleep for a second, attempt to read and format new data from the log file, and then go back to sleep. This loop can be terminated at any time by sending an interrupt (Control-C).

-m
: Print the localized diagnosis message associated with each entry in the fault log.

-n *name*[.*name*]*[=*value*]
: Select fault log or error log events, depending on the -e option, that have properties with a matching name (and optionally a matching value). For string properties the value can be a regular expression match. Regular expression syntax is described in the EXTENDED REGULAR EXPRESSIONS section of the regex(5) manual page. Be careful when using the characters:

  ```
  $ * { ^ | ( ) \
  ```

...or a regular expression, because these are meaningful to the shell. It is safest to enclose any of these in single quotes. For numeric properties, the value can be octal, hex, or decimal.

`-R` *dir*
Use the specified root directory for the log files accessed by `fmdump`, instead of the default root (`/`).

`-t` *time*
Select events that occurred at or after the specified time. The time can be specified using any of the following forms:

mm/*dd*/*yy hh*:*mm*:*ss*
Month, day, year, hour in 24-hour format, minute, and second. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.

mm/*dd*/*yy hh*:*mm*
Month, day, year, hour in 24-hour format, and minute. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.

mm/*dd*/*yy*
12:00:00AM on the specified month, day, and year.

ddMonyy hh:*mm*:*ss*
Day, month name, year, hour in 24-hour format, minute, and second. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.

ddMonyy hh:*mm*
Day, month name, year, hour in 24-hour format, and minute. Any amount of whitespace can separate the date and time. The argument should be quoted so that the shell interprets the two strings as a single argument.

Mon dd hh:*mm*:*ss*
Month, day, hour in 24-hour format, minute, and second of the current year.

yyyy-*mm*-*dd* `[T` *hh*:*mm*`[`:*ss*`]]`
Year, month, day, and optional hour in 24-hour format, minute, and second. The second, or hour, minute, and second, can be optionally omitted.

ddMonyy
12:00:00AM on the specified day, month name, and year.

hh:*mm*:*ss*
Hour in 24-hour format, minute, and second of the current day.

hh:*mm*
Hour in 24-hour format and minute of the current day.

T`ns` | *T*`nsec`
T nanoseconds ago where T is an integer value specified in base 10.

T`us` | *T*`usec`
T microseconds ago where T is an integer value specified in base 10.

T`ms` | *T*`msec`
T milliseconds ago where T is an integer value specified in base 10.

T`s` | *T*`sec`
T seconds ago where *T* is an integer value specified in base 10.

T`m` | *T*`min`
T minutes ago where *T* is an integer value specified in base 10.

T`h` | *T*`hour`
T hours ago where *T* is an integer value specified in base 10.

T`d` | *T*`day`
T days ago where *T* is an integer value specified in base 10.

You can append a decimal fraction of the form `.`*n* to any `-t` option argument to indicate a fractional number of seconds beyond the specified time.

`-T` *time*
Select events that occurred at or before the specified time. *time* can be specified using any of the time formats described for the `-t` option.

`-u` *uuid*
Select fault diagnosis events that exactly match the specified *uuid*. Each diagnosis is associated with a Universal Unique Identifier (UUID) for identification purposes. The `-u` option can be combined with other options such as `-v` to show all of the details associated with a particular diagnosis.

If the `-e` option and `-u` option are both present, the error events that are cross-referenced by the specified diagnosis are displayed.

`-v`
Display verbose event detail. The event display is enlarged to show additional common members of the selected events.

`-V`
Display very verbose event detail. The event display is enlarged to show every member of the name-value pair list associated with each event. In addition, for fault logs, the event display includes a list of cross-references to the corresponding errors that were associated with the diagnosis.

Operands The following operands are supported:

file Specifies an alternate log file to display instead of the system fault log. The `fmdump` utility determines the type of the specified log automatically and produces appropriate

output for the selected log.

Examples

EXAMPLE 1 Retrieving Given Class from `fmd` Log

Use any of the following commands to retrieve information about a specified class from the `fmd` log. The complete class name is `ereport.io.ddi.context`.

```
# fmdump -Ve -c 'ereport.io.ddi.context'
# fmdump -Ve -c 'ereport.*.context'
# fmdump -Ve -n 'class=ereport.io.ddi.context'
# fmdump -Ve -n 'class=ereport.*.context'
```

Any of the preceding commands produces the following output:

```
Oct 06 2007 11:53:20.975021712 ereport.io.ddi.context
        nvlist version: 0
                class = ereport.io.ddi.context
                ena = 0x1b03a15ecf00001
                detector = (embedded nvlist)
                nvlist version: 0
                        version = 0x0
                        scheme = dev
                        device-path = /
                (end detector)

                __ttl = 0x1
                __tod = 0x470706b0 0x3a1da690
```

EXAMPLE 2 Retrieving Specific Detector Device Path from `fmd` Log

The following command retrieves a detector device path from the `fmd` log.

```
# fmdump -Ve -n 'detector.device-path=.*/disk@1,0$'
Oct 06 2007 12:04:28.065660760 ereport.io.scsi.disk.rqs
nvlist version: 0
       class = ereport.io.scsi.disk.rqs
       ena = 0x453ff3732400401
       detector = (embedded nvlist)
                nvlist version: 0
                        version = 0x0
                        scheme = dev
                        device-path = /pci@0,0/pci1000,3060@3/disk@1,0
                (end detector)

                __ttl = 0x1
                __tod = 0x4707094c 0x3e9e758
```

Exit Status

The following exit values are returned:

`0` Successful completion. All records in the log file were examined successfully.

1 A fatal error occurred. This prevented any log file data from being examined, such as failure to open the specified file.

2 Invalid command-line options were specified.

3 The log file was opened successfully, but one or more log file records were not displayed, either due to an I/O error or because the records themselves were malformed. `fmdump` issues a warning message for each record that could not be displayed, and then continues on and attempts to display other records.

Files

`/var/fm/fmd`	Fault management log directory
`/var/fm/fmd/errlog`	Fault management error log
`/var/fm/fmd/fltlog`	Fault management fault log

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/fault-management
Interface Stability	See below.

The command-line options and the human-readable fault log output are Committed. The human-readable error log output is Private.

See Also `sh(1)`, `fmadm(1M)`, `fmd(1M)`, `fmstat(1M)`, `syslogd(1M)`, `libexacct(3LIB)`, `attributes(5)`, `regex(5)`

System Administration Guide: Security Services

http://www.sun.com/msg/

Notes Fault logs contain references to records stored in error logs that can be displayed using `fmdump -V` to understand the errors that were used in the diagnosis of a particular fault. These links are preserved if an error log is renamed as part of log rotation. They can be broken by removing an error log file, or by moving it to another filesystem directory. `fmdump` can not display error information for such broken links. It continues to display any and all information present in the fault log.

Name fmstat – report fault management module statistics

Synopsis

```
fmstat  [-astTz] [-m module] [interval [count]]
```

Description The `fmstat` utility can be used by administrators and service personnel to report statistics associated with the Solaris Fault Manager, `fmd(1M)` and its associated set of modules. The Fault Manager runs in the background on each Solaris system. It receives telemetry information relating to problems detected by the system software, diagnoses these problems, and initiates proactive self-healing activities such as disabling faulty components.

You can use `fmstat` to view statistics for diagnosis engines and agents that are currently participating in fault management. The documentation for `fmd(1M)`, `fmadm(1M)`, and `fmdump(1M)` describes more about tools to observe fault management activities.

If the `-m` option is present or the `-t` option is present, `fmstat` reports any statistics kept by the specified fault management module. The module list can be obtained using `fmadm config`.

If the `-m` option is not present, `fmstat` reports the following statistics for each of its client modules:

`module`	The name of the fault management module, as reported by `fmadm config`.
`ev_recv`	The number of telemetry events received by the module.
`ev_acpt`	The number of events accepted by the module as relevant to a diagnosis.
`wait`	The average number of telemetry events waiting to be examined by the module.
`svc_t`	The average service time for telemetry events received by the module, in milliseconds.
`%w`	The percentage of time that there were telemetry events waiting to be examined by the module.
`%b`	The percentage of time that the module was busy processing telemetry events.
`open`	The number of active cases (open problem investigations) owned by the module.
`solve`	The total number of cases solved by this module since it was loaded.
`memsz`	The amount of dynamic memory currently allocated by this module.
`bufsz`	The amount of persistent buffer space currently allocated by this module.

The `fmstat` utility requires the user to posses the `SYS_CONFIG` privilege. Refer to the *System Administration Guide: Security Services* for more information about how to configure Solaris privileges.

Options The following options are supported:

`-a` Print all statistics for a module, including those kept on its behalf by `fmd`. If the `-a` option is not present, only those statistics kept by the module are reported. If the `-a` option is used without the `-m` *module*, a set of global statistics associated with `fmd` are displayed.

`-m` *module* Print a report on the statistics associated with the specified fault management module, instead of the default statistics report. Modules can publish an arbitrary set of statistics to help Sun service the fault management software itself. The module statistics constitute a Private interface. See `attributes(5)` for information on Sun's rules for Private interfaces. Scripts should not be written that depend upon the values of fault management module statistics as they can change without notice.

`-s` Print a report on Soft Error Rate Discrimination (SERD) engines associated with the module instead of the default module statistics report. A SERD engine is a construct used by fault management software to determine if a statistical threshold measured as *N* events in some time *T* has been exceeded. The `-s` option can only be used in combination with the `-m` option.

`-t` Print a report on the statistics associated with each fault management event transport. Each fault management module can provide the implementation of one or more event transports.

`-T` Print a table of the authority information associated with each fault management event transport. If the `-m` option is present, only transports associated with the specified module are displayed.

`-z` Omit statistics with a zero value from the report associated with the specified fault management module. The `-z` option can only be used in combination with the `-m` option.

Operands The following operands are supported:

count Print only count reports, and then exit.

interval Print a new report every *interval* seconds.

If no *interval* and no *count* are specified, a single report is printed and `fmstat` exits. If an *interval* is specified but no *count* is specified, `fmstat` prints reports every *interval* seconds indefinitely until the command is interrupted.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` A fatal error occurred. A fatal error could be the failure to communicate with `fmd(1M)`. It could also be that insufficient privileges were available to perform the requested operation.

2 Invalid command-line options were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfmd
Interface Stability	See below.

The command-line options are Evolving. The human-readable default report is Unstable. The human-readable module report is Private.

See Also `fmadm(1M)`, `fmd(1M)`, `fmdump(1M)`, `attributes(5)`

System Administration Guide: Security Services

Name fmthard – populate label on hard disks

Synopsis

SPARC
```
fmthard -d data | -n volume_name | -s datafile [-i] /dev/rdsk/c?
     [t?] d?s2
```

x86
```
fmthard -d data | -n volume_name | -s datafile [-i] /dev/rdsk/c?
     [t?] d?s2
```

Description The `fmthard` command updates the VTOC (Volume Table of Contents) on hard disks and, on x86 systems, adds boot information to the Solaris `fdisk` partition. One or more of the options `-s` *datafile*, `-d` *data*, or `-n` *volume_name* must be used to request modifications to the disk label. To print disk label contents, see `prtvtoc(1M)`. The `/dev/rdsk/c?[t?]d?s2` file must be the character special file of the device where the new label is to be installed. On x86 systems, `fdisk(1M)` must be run on the drive before `fmthard`.

If you are using an x86 system, note that the term "partition" in this page refers to *slices* within the x86 `fdisk` partition on x86 machines. Do not confuse the partitions created by `fmthard` with the partitions created by `fdisk`.

Options The following options are supported:

`-d` *data*
: The *data* argument of this option is a string representing the information for a particular partition in the current VTOC. The string must be of the format *part:tag:flag:start:size* where *part* is the partition number, *tag* is the ID TAG of the partition, *flag* is the set of permission flags, *start* is the starting sector number of the partition, and *size* is the number of sectors in the partition. See the description of the *datafile* below for more information on these fields.

`-i`
: This option allows the command to create the desired VTOC table, but prints the information to standard output instead of modifying the VTOC on the disk.

`-n` *volume_name*
: This option is used to give the disk a *volume_name* up to 8 characters long.

`-s` *datafile*
: This option is used to populate the VTOC according to a *datafile* created by the user. If the *datafile* is – (a hyphen), `fmthard` reads from standard input. The *datafile* format is described below. This option causes all of the disk partition timestamp fields to be set to zero.

  Every VTOC generated by `fmthard` will also have partition 2, by convention, that corresponds to the whole disk. If the input in *datafile* does not specify an entry for partition 2, a default partition 2 entry will be created automatically in VTOC with the tag `V_BACKUP` and size equal to the full size of the disk.

The *datafile* contains one specification line for each partition, starting with partition 0. Each line is delimited by a new-line character (`\n`). If the first character of a line is an asterisk (*), the line is treated as a comment. Each line is composed of entries that are position-dependent, separated by white space and having the following format:

partition tag flag starting_sector size_in_sectors

where the entries have the following values:

partition
The partition number. Currently, for Solaris SPARC, a disk can have up to 8 partitions, `0`–`7`. Even though the *partition* field has 4 bits, only 3 bits are currently used. For x86, all 4 bits are used to allow slices `0`–`15`. Each Solaris `fdisk` partition can have up to 16 slices.

tag
The partition tag: a decimal number. The following are reserved codes: `0` (`V_UNASSIGNED`), `1` (`V_BOOT`), `2` (`V_ROOT`), `3` (`V_SWAP`), `4` (`V_USR`), `5` (`V_BACKUP`), `6` (`V_STAND`), `7` (`V_VAR`), and `8` (`V_HOME`).

flag
The flag allows a partition to be flagged as unmountable or read only, the masks being: `V_UNMNT 0x01`, and `V_RONLY 0x10`. For mountable partitions use `0x00`.

starting_sector
The sector number (decimal) on which the partition starts.

size_in_sectors
The number (decimal) of sectors occupied by the partition.

You can save the output of a `prtvtoc` command to a file, edit the file, and use it as the *datafile* argument to the `-s` option.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`uname(1)`, `format(1M)`, `prtvtoc(1M)`, `attributes(5)`

x86 Only `fdisk(1M)`, `installgrub(1M)`

Notes

Special care should be exercised when overwriting an existing VTOC, as incorrect entries could result in current data being inaccessible. As a precaution, save the old VTOC.

For disks under two terabytes, `fmthard` cannot write a VTOC on an unlabeled disk. Use `format(1M)` for this purpose.

Name format – disk partitioning and maintenance utility

Synopsis

```
format [-f command-file] [-l log-file] [-x data-file]
     [-d disk-name] [-t disk-type] [-p partition-name]
     [-s] [-m] [-M] [-e] [disk-list]
```

Description `format` enables you to format, label, repair, and analyze disks on your system. Unlike previous disk maintenance programs, `format` runs under SunOS. Because there are limitations to what can be done to the system disk while the system is running, `format` is also supported within the memory-resident system environment. For most applications, however, running `format` under SunOS is the more convenient approach.

`format` first uses the disk list defined in *data-file* if the `-x` option is used. `format` then checks for the `FORMAT_PATH` environment variable, a colon-separated list of filenames and/or directories. In the case of a directory, `format` searches for a file named `format.dat` in that directory; a filename should be an absolute pathname, and is used without change. `format` adds all disk and partition definitions in each specified file to the working set. Multiple identical definitions are silently ignored. If `FORMAT_PATH` is not set, the path defaults to `/etc/format.dat`.

disk-list is a list of disks in the form `c?t?d?` or `/dev/rdsk/c?t?d?s?`. With the latter form, shell wildcard specifications are supported. For example, specifying `/dev/rdsk/c2*` causes `format` to work on all drives connected to controller `c2` only. If no *disk-list* is specified, `format` lists all the disks present in the system that can be administered by `format`.

Removable media devices are listed only when users execute `format` in expert mode (option `-e`). This feature is provided for backward compatibility. Use `rmformat(1)` for rewritable removable media devices.

Options The following options are supported:

`-d` *disk-name* — Specify which disk should be made current upon entry into the program. The disk is specified by its logical name (for instance, `-d c0t1d0`). This can also be accomplished by specifying a single disk in the disk list.

`-e` — Enable SCSI expert menu. Note this option is not recommended for casual use.

`-f` *command-file* — Take command input from *command-file* rather than the standard input. The file must contain commands that appear just as they would if they had been entered from the keyboard. With this option, `format` does not issue `continue?` prompts; there is no need to specify y(es) or n(o) answers in the *command-file*. In non-interactive mode, `format` does not initially expect the input of a disk selection number. The user must specify the current working disk with the `-d` *disk-name* option when format is invoked, or specify `disk` and the disk selection number in the *command-file*.

`-l` *log-file*	Log a transcript of the `format` session to the indicated *log-file*, including the standard input, the standard output and the standard error.
`-m`	Enable extended messages. Provides more detailed information in the event of an error.
`-M`	Enable extended and diagnostic messages. Provides extensive information on the state of a SCSI device's mode pages, during formatting.
`-p` *partition-name*	Specify the partition table for the disk which is current upon entry into the program. The table is specified by its name as defined in the data file. This option can be used only if a disk is being made current, and its type is either specified or available from the disk label.
`-s`	Silent. Suppress all of the standard output. Error messages are still displayed. This is generally used in conjunction with the `-f` option.
`-t` *disk-type*	Specify the type of disk which is current upon entry into the program. A disk's type is specified by name in the data file. This option can only be used if a disk is being made current as described above.
`-x` *data-file*	Use the list of disks contained in *data-file*.

Usage When you invoke format with no options or with the `-e`, `-l`, `-m`, `-M`, or `-s` options, the program displays a numbered list of available disks and prompts you to specify a disk by list number. If the machine has more than 10 disks, press SPACE to see the next screenful of disks.

You can specify a disk by list number even if the disk is not displayed in the current screenful. For example, if the current screen shows disks 11-20, you can enter `25` to specify the twenty-fifth disk on the list. If you enter a number for a disk that is not currently displayed, `format` prompts you to verify your selection. If you enter a number from the displayed list, `format` silently accepts your selection.

After you specify a disk, `format` displays its main menu. This menu enables you to perform the following tasks:

analyze	Run read, write, compare tests, and data purge. The data purge function implements the National Computer Security Center Guide to Understanding Data Remnance (NCSC-TG-025 version 2) Overwriting Algorithm. See NOTES.
backup	Search for backup labels.
cache	Enable, disable, and query the state of the write cache and read cache. This menu item only appears when `format` is invoked with the `-e` option, and is only supported on SCSI devices..
current	Display the device name, the disk geometry, and the pathname to the disk device.

defect	Retrieve and print defect lists. This option is supported only on SCSI devices. IDE disks perform automatic defect management. Upon using the `defect` option on an IDE disk, you receive the message: `Controller does not support defect management or disk supports automatic defect management.`
disk	Choose the disk that will be used in subsequent operations (known as the current disk.)
fdisk	Run the `fdisk(1M)` program to create a `fdisk` partition for Solaris software (x86 based systems only).
format	Format and verify the current disk. This option is supported only on SCSI devices. IDE disks are pre-formatted by the manufacturer. Upon using the `format` option on an IDE disk, you receive the message: `Cannot format this drive. Please use your manufacturer-supplied formatting utility.`
inquiry	Display the vendor, product name, and revision level of the current drive.
label	Write a new label to the current disk.
partition	Create and modify slices.
quit	Exit the format menu.
repair	Repair a specific block on the disk.
save	Save new disk and slice information.
type	Select (define) a disk type.
verify	Read and display labels. Print information such as the number of cylinders, alternate cylinders, heads, sectors, and the partition table.
volname	Label the disk with a new eight character volume name.

Environment Variables

`FORMAT_PATH`	a colon-separated list of filenames and/or directories of disk and partition definitions. If a directory is specified, `format` searches for the file `format.dat` in that directory.

Files

`/etc/format.dat`	default data file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `fmthard(1M)`, `prtvtoc(1M)`, `rmformat(1)`, `format.dat(4)`, `attributes(5)`, `sd(7D)`

System Administration Guide: Basic Administration

x86 Only `fdisk(1M)`

Warnings When the `format` function is selected to format the Maxtor 207MB disk, the following message displays:

```
Mode sense page(4) reports rpm value as 0, adjusting it to 3600
```

This is a drive bug that may also occur with older third party drives. The above message is not an error; the drive will still function correctly.

Cylinder 0 contains the partition table (disk label), which can be overwritten if used in a raw disk partition by third party software.

`format` supports writing EFI-compliant disk labels in order to support disks or LUNs with capacities greater than one terabyte. However, care should be exercised since many software components, such as filesystems and volume managers, are still restricted to capacities of one terabyte or less. See the *System Administration Guide: Basic Administration* for additional information.

By default, on an unlabeled disk, EFI labels will be written on disks larger than 2 TB. When `format` is invoked with the `-e` option, on writing the label, the label type can be chosen. Booting is not currently supported on a disk with an EFI label.

Notes `format` provides a help facility you can use whenever format is expecting input. You can request help about what information is expected by simply entering a question mark (`?`) and `format` prints a brief description of what type of input is needed. If you enter a `?` at the menu prompt, a list of available commands is displayed.

For SCSI disks, formatting is done with both Primary and Grown defects list by default. However, if only Primary list is extracted in defect menu before formatting, formatting will be done with Primary list only.

Changing the state of the caches is only supported on SCSI devices, and not all SCSI devices support changing or saving the state of the caches.

The NCSC-TG-025 algorithm for overwriting meets the DoD 5200.28-M (ADP Security Manual) Eraser Procedures specification. The NIST Guidelines for Media Sanitization (NIST SP 800-88) also reference this algorithm.

Name fpsd – Fp-scrubber daemon

Synopsis

```
/usr/lib/fps/fpsd
```

Description `fpsd` is a user-level daemon that periodically runs non-intrusive tests to validate proper functioning of FPU (Floating Point Unit) hardware in the system. A fault management action is initiated by means of `fmd(1M)`, in case an error is detected by the test.

Exit Status The following exit values are returned:

`0`
Successful completion.

`non-zero`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfsu
Interface Stability	Uncommitted

See Also `svcs(1)`, `fmd(1M)`, `svcadm(1M)`, `svccfg(1M)`, `attributes(5)`, `smf(5)`

Notes The `fpsd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/fpsd:default
```

You can use `svccfg(1M)` to change the default `fpsd` behavior:

```
Property Name            Type     Description
-------------            ----     -----------
config/exclude_cpus      astring  comma delimited list of
                                   CPU IDs to be excluded
                                   from proactive testing.
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Disabling the service can increase system's vulnerability to silent data corruption, if a fault were to develop in an FPU within the system.

Name fruadm – prints and updates customer data associated with FRUs

Synopsis

```
/usr/platform/sun4u/sbin/fruadm

/usr/platform/sun4u/sbin/fruadm -l

/usr/platform/sun4u/sbin/fruadm [-r] path [text]
```

Description `fruadm` prints or sets the customer data for Field-Replaceable Units (FRUs).

Without arguments, `fruadm` prints the paths of all FRU ID-capable FRUs (containers) in the system, along with the contents of the customer data record, if present, for each such FRU; for FRUs without customer data, `fruadm` prints only the container's path.

Only a privileged user can create or update data in containers. The privileges required to perform these write operations are hardware dependent. Typically, a default system configuration restricts write operations to the superuser or to the platform-administrator user.

Options The following options are supported:

`-l` List the system's frutree paths.

`-r` Recursively display or update the data for all containers rooted at the argument *path*.

Operands The following operands are supported:

path A full or partial system frutree path for or under which to print or set the customer data. The first field of each line of output of `fruadm -l` gives the valid full `frutree` paths for the system.

Paths can include shell meta-characters; such paths should be quoted appropriately for the user's shell. For partial paths, the first matching full path is selected for display or update. Without the `-r` option, the path must be that of a container; with the `-r` option, all containers (if any) under *path* will be selected.

text Up to 80 characters of text set as the customer data. If the text contains white space or shell metacharacters, it should be quoted appropriately for the user's shell.

Examples **EXAMPLE 1** Displaying All Customer Data

The following example prints all customer data available from FRUs on the system. For containers with no customer data, only the containers' paths will be listed.

```
example% fruadm
```

EXAMPLE 2 Displaying Customer Data For a Single FRU

The following command prints the customer data, if present, for the specified FRU:

```
example% fruadm /frutree/chassis/system-board
```

EXAMPLE 3 Displaying Customer Data For a Single FRU

The following command prints the customer data, if present, for the first `mem-module` found:

```
example% fruadm mem-module
```

EXAMPLE 4 Setting Customer Data

The following example sets the customer data for a FRU:

```
example# fruadm system-board 'Asset Tag 123456'
```

EXAMPLE 5 Setting Customer Data

The following command sets the customer data for all FRUs under `chassis`:

```
example# fruadm -r /frutree/chassis "Property of XYZ, Inc."
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfruip.u
Interface Stability	Unstable

See Also

`prtfru(1M)`, `attributes(5)`

Name fsck – check and repair file systems

Synopsis

```
fsck  [-F FSType] [-m] [-V] [special]...

fsck  [-F FSType] [-n |  N |  y |  Y] [-V] [-o FSType-specific-options]
 [special]...
```

Description `fsck` audits and interactively repairs inconsistent file system conditions. If the file system is inconsistent the default action for each correction is to wait for the user to respond `yes` or `no`. If the user does not have write permission `fsck` defaults to a `no` action. Some corrective actions will result in loss of data. The amount and severity of data loss can be determined from the diagnostic output.

FSType-specific-options are options specified in a comma-separated (with no intervening spaces) list of options or keyword-attribute pairs for interpretation by the *FSType*-specific module of the command.

special represents the character special device on which the file system resides, for example, `/dev/rdsk/c1t0d0s7`. Note: the character special device, not the block special device, should be used. `fsck` will not work if the block device is mounted.

If no *special* device is specified `fsck` checks the file systems listed in `/etc/vfstab`. Those entries in `/etc/vfstab` which have a character special device entry in the `fsckdev` field and have a non-zero numeric entry in the `fsckpass` field will be checked. Specifying `-F` *FSType* limits the file systems to be checked to those of the type indicated.

If *special* is specified, but `-F` is not, the file system type will be determined by looking for a matching entry in `/etc/vfstab`. If no entry is found, the default local file system type specified in `/etc/default/fs` will be used.

If a file system type supports parallel checking, for example, `ufs`, some file systems eligible for checking may be checked in parallel. Consult the file system-specific man page (for example, `fsck_ufs`(1M)) for more information.

Options The following generic options are supported:

`-F` *FSType*
Specify the file system type on which to operate.

`-m`
Check but do not repair. This option checks that the file system is suitable for mounting, returning the appropriate exit status. If the file system is ready for mounting, `fsck` displays a message such as:

```
ufs fsck: sanity check: /dev/rdsk/c0t3d0s1 okay
```

`-n` | `-N`
Assume a `no` response to all questions asked by `fsck`; do not open the file system for writing.

`-V`
Echo the expanded command line but do not execute the command. This option may be used to verify and to validate the command line.

`-y|Y`
Assume a `yes` response to all questions asked by `fsck`.

`-o` *specific-options*
These *specific-options* can be any combination of the following separated by commas (with no intervening spaces).

`b=`*n*
Use block *n* as the super block for the file system. Block 32 is always one of the alternate super blocks. Determine the location of other super blocks by running `newfs(1M)` with the `-Nv` options specified.

`c`
If the file system is in the old (static table) format, convert it to the new (dynamic table) format. If the file system is in the new format, convert it to the old format provided the old format can support the file system configuration. In interactive mode, `fsck` will list the direction the conversion is to be made and ask whether the conversion should be done. If a negative answer is given, no further operations are done on the file system. In preen mode, the direction of the conversion is listed and done if possible without user interaction. Conversion in preen mode is best used when all the file systems are being converted at once. The format of a file system can be determined from the first line of output from `fstyp(1M)`. Note: the `c` option is seldom used and is included only for compatibility with pre-4.1 releases. There is no guarantee that this option will be included in future releases.

`f`
Force checking of file systems regardless of the state of their super block clean flag.

`p`
Check and fix the file system non-interactively ("preen"). Exit immediately if there is a problem requiring intervention. This option is required to enable parallel file system checking.

`w`
Check writable file systems only.

Exit Status

`0`	file system is okay and does not need checking
`1`	erroneous parameters are specified
`32`	file system is unmounted and needs checking (`fsck -m` only)
`33`	file system is already mounted
`34`	cannot stat device
`36`	uncorrectable errors detected - terminate normally

37	a signal was caught during processing
39	uncorrectable errors detected - terminate immediately
40	for root, same as 0.

Usage The fsck command is *large file aware* for UFS file systems, per the largefile(5) man page.

Files /etc/default/fs
default local file system type. Default values can be set for the following flags in /etc/default/fs. For example: LOCAL=ufs.

LOCAL
The default partition for a command if no FSType is specified.

/etc/vfstab
list of default parameters for each file system

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also clri(1M), fsck_cachefs(1M), fsck_ufs(1M), fsdb_ufs(1M), fsirand(1M), fstyp(1M), mkfs(1M), mkfs_ufs(1M), mountall(1M), newfs(1M), reboot(1M), vfstab(4), attributes(5), largefile(5), ufs(7FS)

Warnings The operating system buffers file system data. Running fsck on a mounted file system can cause the operating system's buffers to become out of date with respect to the disk. For this reason, the file system should be *unmounted* when fsck is used. If this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately after fsck is run. Quite often, however, this will not be sufficient. A panic will probably occur if running fsck on a file system modifies the file system.

Notes This command may not be supported for all *FSTypes*.

Starting with Solaris 9, fsck manages extended attribute data on the disk. (See fsattr(5) for a description of extended file attributes.) A file system with extended attributes can be mounted on versions of Solaris that are not attribute-aware (versions prior to Solaris 9), but the attributes will not be accessible and fsck will strip them from the files and place them in lost+found. Once the attributes have been stripped, the file system is completely stable on versions of Solaris that are not attribute-aware, but would be considered corrupted on attribute-aware versions. In the latter circumstance, run the attribute-aware fsck to stabilize the file system before using it in an attribute-aware environment.

Name fsck_cachefs – check integrity of data cached with CacheFS

Synopsis

```
fsck -F cachefs [-m] [-o noclean] cache_directory
```

Description The CacheFS version of the `fsck` command checks the integrity of a cache directory.This utility corrects any CacheFS problems it finds by default. There is no interactive mode. The most likely invocation of `fsck` for CacheFS file systems is at boot time from an entry in the `/etc/vfstab` file. See `vfstab(4)`.

Options The following options are supported:

`-m` Check, but do not repair.

`-o noclean` Force a check on the cache even if there is no reason to suspect there is a problem.

Examples **EXAMPLE 1** Using `fsck_cachefs` to Force a Check on the Cache Directory

The following example forces a check on the cache directory `/cache3`:

```
example% fsck -F cachefs -o noclean /cache3
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `cfsadmin(1M)`, `fsck(1M)`, `mount_cachefs(1M)`, `vfstab(4)`, `attributes(5)`

Name fsck_pcfs – file system consistency check and interactive repair

Synopsis

```
fsck -F pcfs [generic_options] special

fsck -F pcfs [generic_options] [-o specific_options] special
```

Description The `fsck` utility audits and interactively repairs inconsistent conditions on file systems. *special* represents the character special device on which the file system resides, for example `/dev/rdiskette`. The character special device, not the block special device, should be used.

In the case of correcting serious inconsistencies, by default, `fsck` asks for confirmation before making a repair and waits for the operator to respond either `yes` or `no`. If the operator does not have write permission on the file system, `fsck` defaults to a `-n` (no corrections) action. See `fsck(1M)`.

Repairing some file system inconsistencies may result in loss of data. The amount and severity of data loss may be determined from the diagnostic output.

When executed with the verify option (`-o v`), `fsck_pcfs` automatically scans the entire file system to verify that all of its allocation units are accessible. If it finds any units inaccessible, it updates the file allocation table (FAT) appropriately. It also updates any effected directory entries to reflect the problem. This directory update includes truncating the file at the point in its allocation chain where the file data is no longer accessible. Any remaining accessible allocation units become orphaned.

Orphaned chains of accessible allocation units are, with the operator's concurrence, linked back into the file system as files in the root directory. These files are assigned names of the form file*NNNN*.chk, where the *N*s are digits in the integral range from `0` through `9`.

After successfully scanning and correcting any errors in the file system, `fsck` displays a summary of information about the file system. This summary includes the size of the file system in bytes, the number of bytes used in directories and individual files, and the number of available allocation units remaining in the file system.

Options

generic_options The following generic options are supported:

`-m` Check but do not repair. This option checks that the file system is suitable for mounting, returning the appropriate exit status. If the file system is ready for mounting, `fsck` displays a message such as:

```
pcfs fsck: sanity check:
/dev/rdiskette okay
```

`-n` | `-N` Assume a no response to all questions asked by `fsck`; do not open the file system for writing.

`-V` Echo the expanded command line, but do not execute the command. This option may be used to verify and to validate the command line.

`-y` | `-Y` Assume a `yes` response to all questions asked by `fsck`.

`-o` *specific_options* Specify `pcfs` file system specific options in a comma-separated list, in any combination, with no intervening spaces.

- v Verify all allocation units are accessible prior to correcting inconsistencies in the metadata.
- p Check and fix the file system non-interactively (`preen`). Exit immediately if there is a problem requiring intervention.
- w Check writable file systems only.

Files *special* The device which contains the pcfs. The device name for a diskette is specified as `/dev/rdiskette0` for the first diskette drive, or `/dev/rdiskette1` for a second diskette drive. A hard disk device or high-capacity removable device name much be qualified with a suffix to indicate the proper `FDISK` partition.

For example, in the names: `/dev/rdsk/c0t0d0p0:c` and `/dev/rdsk/c0t4d0s2:c`, the `:c` suffix indicates the first partition on the disk contains the `pcfs`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu
Interface Stability	Stable

See Also `fsck(1M)`, `fstyp(1M)`, `fdisk(1M)`, `mkfs(1M)`, `mkfs_pcfs(1M)`, `mountall(1M)`, `attributes(5)`, `pcfs(7FS)`,

Warnings The operating system buffers file system data. Running `fsck` on a mounted file system can cause the operating system's buffers to become out of date with respect to the disk. For this reason, the file system should be unmounted when `fsck` is used. If this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately after `fsck` is run. Quite often, however, this is not sufficient. A panic will probably occur if running `fsck` on a file system modifies the file system.

Name fsck_udfs – file system consistency check and interactive repair

Synopsis

```
fsck -F udfs [generic_options] [special . . .]

fsck -F udfs [generic_options] [-o specific_options]
     [special . . .]
```

Description `fsck` audits and interactively repairs inconsistent conditions on file systems. A file system to be checked can be specified by giving the name of the block or character special device or by giving the name of its mount point if a matching entry exists in `/etc/vfstab`.

special represents the character special device, for example, `/dev/rdsk/c0t2d0s0`, on which the file system resides. The character special device, not the block special device should be used. `fsck` does not work on a mounted block device.

If no special device is specified, all `udfs` file systems specified in the `vfstab` file with a `fsckdev` entry are checked. If the `-p` (`preen`) option is specified, `udfs` file systems with an `fsckpass` number greater than `1` are checked in parallel. See `fsck`(1M).

In the case of correcting serious inconsistencies, by default, `fsck` asks for confirmation before making a repair and waits for the operator to respond with either `yes` or `no`. If the operator does not have write permission on the file system, `fsck` defaults to the `-n` (no corrections) option. See `fsck`(1M).

Repairing some file system inconsistencies can result in loss of data. The amount and severity of data loss can be determined from the diagnostic output.

`fsck` automatically corrects innocuous inconsistencies. It displays a message for each corrected inconsistency that identifies the nature of the correction which took place on the file system. After successfully correcting a file system, `fsck` prints the number of files on that file system and the number of used and free blocks.

Inconsistencies checked are as follows:

- Blocks claimed by more than one file or the free list
- Blocks claimed by a file or the free list outside the range of the file system
- Incorrect link counts in file entries
- Incorrect directory sizes
- Bad file entry format
- Blocks not accounted for anywhere
- Directory checks, file pointing to unallocated file entry and absence of a parent directory entry
- Descriptor checks, more blocks for files than there are in the file system
- Bad free block list format
- Total free block count incorrect

Options The following options are supported:

generic_options The following *generic_options* are supported:

`-m` Check but do not repair. This option checks to be sure that the file system is suitable for mounting, and returns the appropriate exit status. If the file system is ready for mounting, `fsck` displays a message such as:

```
udfs fsck: sanity check: /dev/rdsk/c0t2d0s0 okay
```

`-n` | `-N` Assume a `no` response to all questions asked by `fsck`; do not open the file system for writing.

`-V` Echo the expanded command line, but do not execute the command. This option can be used to verify and to validate the command line.

`-y` | `-Y` Assume a `yes` response to all questions asked by `fsck`.

`-o` *specific_options* Specify `udfs` file system specific options in a comma-separated list with no intervening spaces. The following *specific_options* are available:

`f` Force checking of file systems regardless of the state of their logical volume integrity state.

`p` Check and fix the file system non-interactively (`preen`). Exit immediately if there is a problem that requires intervention. This option is required to enable parallel file system checking.

`w` Check writable file systems only.

Files `/etc/vtstab` List of default parameters for each file system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWudf

See Also `fsck(1M)`, `fsdb_udfs(1M)`, `fstyp(1M)`, `mkfs(1M)`, `mkfs_udfs(1M)`, `mountall(1M)`, `reboot(1M)`, `vfstab(4)`, `attributes(5)`

Warnings The operating system buffers file system data. Running `fsck` on a mounted file system can cause the operating system's buffers to become out of date with respect to the disk. For this reason, use `fsck` only when the file system is unmounted. If this is not possible, take care that the system is quiescent and that it is rebooted immediately after running `fsck`. A panic will probably occur if running `fsck` on a file system that modifies the file system while it is mounted.

If an unmount of the file system is not done before the system is shut down, the file system might become corrupted. In this case, a file system check needs to be completed before the next mount operation.

Diagnostics

not writable
: You cannot write to the device.

Currently Mounted on
: The device is already mounted and cannot run `fsck`.

FILE SYSTEM WAS MODIFIED
: File system has been modified to bring it to a consistent state.

Can't read allocation extent
: Cannot read the block containing allocation extent.

Bad tag on alloc extent
: Invalid tag detected when expecting an allocation extent.

Volume sequence tag error
: Invalid tag detected in the volume sequence.

Space bitmap tag error
: Invalid tag detected in the space bitmap.

UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY
: Use `fsck` in interactive mode.

Name fsck_ufs – file system consistency check and interactive repair

Synopsis

```
fsck -F ufs [generic-options] [special]...

fsck -F ufs [generic-options] [-o specific-options]
     [special]...
```

Description The `fsck` utility audits and interactively repairs inconsistent conditions on file systems. A file system to be checked may be specified by giving the name of the block or character *special* device or by giving the name of its mount point if a matching entry exists in `/etc/vfstab`.

The *special* parameter represents the character special device, for example, `/dev/rdsk/c1t0d0s7`, on which the file system resides. The character special device, not the block special device should be used. The `fsck` utility will not work if the block device is mounted, unless the file system is error-locked.

If no *special* device is specified, all `ufs` file systems specified in the `vfstab` with a `fsckdev` entry will be checked. If the `-p` ("preen") option is specified, `ufs` file systems with an `fsckpass` number greater than `1` are checked in parallel. See `fsck`(1M).

In the case of correcting serious inconsistencies, by default, `fsck` asks for confirmation before making a repair and waits for the operator to respond either `yes` or `no`. If the operator does not have write permission on the file system, `fsck` will default to a `-n` (no corrections) action. See `fsck`(1M).

Repairing some file system inconsistencies can result in loss of data. The amount and severity of data loss can be determined from the diagnostic output.

The `fsck` utility automatically corrects innocuous inconsistencies such as unreferenced inodes, too-large link counts in inodes, missing blocks in the free list, blocks appearing in the free list and also in files, or incorrect counts in the super block. It displays a message for each inconsistency corrected that identifies the nature of the correction on the file system which took place. After successfully correcting a file system, `fsck` prints the number of files on that file system, the number of used and free blocks, and the percentage of fragmentation.

Inconsistencies checked include:

- Blocks claimed by more than one inode or the free list.
- Blocks claimed by an inode or the free list outside the range of the file system.
- Incorrect link counts.
- Incorrect directory sizes.
- Bad inode format.
- Blocks not accounted for anywhere.
- Directory checks, file pointing to unallocated inode, inode number out of range, and absence of '.' and '. .' as the first two entries in each directory.
- Super Block checks: more blocks for inodes than there are in the file system.

- Bad free block list format.
- Total free block and/or free inode count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the `lost+found` directory. The name assigned is the inode number. If the `lost+found` directory does not exist, it is created. If there is insufficient space in the `lost+found` directory, its size is increased.

An attempt to mount a `ufs` file system with the `-o nolargefiles` option will fail if the file system has ever contained a large file (a file whose size is greater than or equal to 2 Gbyte). Invoking `fsck` resets the file system state if no large files are present in the file system. A successful mount of the file system after invoking `fsck` indicates the absence of large files in the file system. An unsuccessful mount attempt indicates the presence of at least one large file. See `mount_ufs(1M)`.

Options The *generic-options* consist of the following options:

`-m` Check but do not repair. This option checks that the file system is suitable for mounting, returning the appropriate exit status. If the file system is ready for mounting, `fsck` displays a message such as:

```
ufs fsck: sanity check: /dev/rdsk/c0t3d0s1 okay
```

`-n` | `N` Assume a `no` response to all questions asked by `fsck`; do not open the file system for writing.

`-V` Echo the expanded command line, but do not execute the command. This option may be used to verify and to validate the command line.

`-v` Enables verbose output. Might not be supported by all filesystem-specific `fsck` implementations.

`-y` | `Y` Assume a `yes` response to all questions asked by `fsck`.

See generic `fsck(1M)` for the details for specifying *special*.

`-o` *specific-options* Specify `ufs` file system specific options. These options can be any combination of the following separated by commas (with no intervening spaces).

`b=`*n* Use block *n* as the super block for the file system. Block 32 is always one of the alternate super blocks. Determine the location of other super blocks by running `newfs(1M)` with the `-Nv` options specified.

`f` Force checking of file systems regardless of the state of their super block clean flag.

p Check and fix the file system non-interactively ("preen"). Exit immediately if there is a problem requiring intervention. This option is required to enable parallel file system checking.

w Check writable file systems only.

Files `/etc/vfstab` list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `clri(1M)`, `fsck(1M)`, `fsdb_ufs(1M)`, `fsirand(1M)`, `fstyp(1M)`, `mkfs(1M)`, `mkfs_ufs(1M)`, `mount_ufs(1M)`, `mountall(1M)`, `newfs(1M)`, `reboot(1M)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`, `ufs(7FS)`

Warnings The operating system buffers file system data. Running `fsck` on a mounted file system can cause the operating system's buffers to become out of date with respect to the disk. For this reason, the file system should be *unmounted* when `fsck` is used. If this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately after `fsck` is run. Quite often, however, this will not be sufficient. A panic will probably occur if running `fsck` on a file system modifies the file system.

Notes It is usually faster to check the character special device than the block special device.

Running `fsck` on file systems larger than 2 Gb fails if the user chooses to use the block interface to the device:

```
fsck /dev/dsk/c?t?d?s?
```

rather than the raw (character special) device:

```
fsck /dev/rdsk/c?t?d?s?
```

Name fsdb – file system debugger

Synopsis `fsdb [-F` *FSType*`] [-V] [-o` *FSType-specific_options*`]` *special*

Description `fsdb` is a file system debugger that allows for the manual repair of a file system after a crash. *special* is a special device used to indicate the file system to be debugged. `fsdb` is intended for experienced users only. *FSType* is the file system type to be debugged. Since different *FSTypes* have different structures and hence different debugging capabilities, the manual pages for the *FSType*-specific `fsdb` should be consulted for a more detailed description of the debugging capabilities.

Options

`-F` Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table, or by consulting `/etc/default/fs`.

`-V` Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option may be used to verify and validate the command line.

`-o` Specify *FSType*-specific options.

Usage See `largefile(5)` for the description of the behavior of `fsdb` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/default/fs` default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: LOCAL=ufs

`LOCAL:` The default partition for a command if no FSType is specified.

`/etc/vfstab` list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `vfstab(4)`, `attributes(5)`, `largefile(5)` Manual pages for the *FSType*-specific modules of `fsdb`.

Notes This command may not be supported for all *FSTypes*.

Name fsdb_udfs – udfs file system debugger

Synopsis `fsdb [-F] udfs [`*generic_option*`] [-o` *specific_option*`]` *special*

Description The `fsdb_udfs` command is an interactive tool that can be used to patch up a damaged `udfs` file system. `fsdb_udfs` has conversions to translate block and i-numbers into their corresponding disk addresses. Mnemonic offsets to access different parts of an inode are also included. Mnemonic offsets greatly simplify the process of correcting control block entries or descending the file system tree.

`fsdb` contains several error-checking routines to verify inode and block addresses. These can be disabled if necessary by invoking `fsdb` with the `-o` option or by using the `o` command.

`fsdb` reads one block at a time, and therefore works with raw as well as block I/O devices. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block. In order to modify any portion of the disk, `fsdb` must be invoked with the `-w` option.

Wherever possible, `adb`-like syntax has been adopted to promote the use of `fsdb` through familiarity.

Options The following options are supported:

`-o` *specific_option* Specify `udfs` file system specific options in a comma-separated list with no intervening spaces. The following specific options are supported:

o	Override some error conditions.
p=*string*	Set prompt to *string*.
w	Open for write.
?	Display usage.

Usage Numbers are considered hexadecimal by default. The user has control over how data is to be displayed or accepted. The `base` command displays or sets the input and output base. Once set, all input defaults to this base and all output displays in this base. The base can be overriden temporarily for input by preceding hexadecimal numbers by `0x`, preceding decimal numbers with a `0t`, or octal numbers with a `0`. Hexadecimal numbers beginning with `a-f` or `A-F` must be preceded with a `0x` to distinguish them from commands.

Disk addressing by `fsdb` is at the byte level. However, `fsdb` offers many commands to convert a desired inode, directory entry, block, and so forth, to a byte address. After the address has been calculated, `fsdb` records the result in the current address (`dot`).

Several global values are maintained by `fsdb`:

- Current base (referred to as `base`)
- Current address (referred to as `dot`)

- Current inode (referred to as `inode`)
- Current count (referred to as `count`)
- Current type (referred to as `type`)

Most commands use the preset value of `dot` in their execution. For example,

```
 > 2:inode
```

first sets the value of dot (`.`) to `2`, colon (`:`), signifies the start of a command, and the `inode` command sets `inode` to `2`. A count is specified after a comma (`,`). Once set, count remains at this value until a new command is encountered that resets the value back to `1` (the default).

So, if

```
> 2000,400/X
```

is entered, `400` hex longs are listed from `2000`, and when completed, the value of dot is `2000 + 400 * sizeof` (long). If a RETURN is then entered, the output routine uses the current values of `dot`, `count`, and `type` and displays `400` more hex longs. An asterisk (`*`) causes the entire block to be displayed. An example showing several commands and the use of RETURN would be:

```
> 2:ino; 0:dir?d
```

or

```
> 2:ino; 0:db:block?d
```

The two examples are synonymous for getting to the first directory entry of the root of the file system. Once there, subsequently entering a RETURN, plus (`+`), or minus (`-`) advances to subsequent entries. Notice that

```
> 2:inode; :ls
```

or

```
> :ls /
```

is again synonymous.

Expressions The following symbols are recognized by `fsdb`:

RETURN Update the value of dot by the current value of *type* and *display* using the current value of *count*.

\# Update the value of dot by specifying a numeric expression. Specify numeric expressions using addition, subtraction, mulitiplication, and division operators (`+`, `-`, `*`, and `%`). Numeric expressions are evaluated from left to right and can use parentheses. After evaluation, the value of dot is updated.

, *count*	Update the count indicator. The global value of *count* is updated to *count*. The value of *count* remains until a new command is run. A *count* specifier of * attempts to show a blocks's worth of information. The default for *count* is `1`.
?*f*	Display in structured style with format specifier `f`. See `Formatted Output`.
/*f*	Display in unstructured style with format specifier `f`. See `Formatted Output`.
.	Display the value of dot.
+*e*	Increment the value of dot by the expression *e*. The amount actually incremented is dependent on the size of type: `dot = dot +` *e* `* sizeof` (*type*) The default for *e* is `1`.
−*e*	Decrement the value of dot by the expression *e* . See `+`.
**e*	Multiply the value of dot by the expression *e*. Multiplication and division don't use *type*. In the above calculation of dot, consider the `sizeof` (*type*) to be `1`.
%*e*	Divide the value of dot by the expression *e*. See *.
< *name*	Restore an address saved in register *name*. *name* must be a single letter or digit.
> *name*	Save an address in register *name*. *name* must be a single letter or digit.
=*f*	Display indicator. If *f* is a legitimate format specifier (see `Formatted Output`), then the value of dot is displayed using format specifier *f*. Otherwise, assignment is assumed. See `= [s] [e]`.
= [*s*] [*e*]	Change the value of dot using an assignment indicator. The address pointed to by dot has its contents changed to the value of the expression *e* or to the ASCII representation of the quoted (") string *s*. This can be useful for changing directory names or ASCII file information.
=+ *e*	Change the value of dot using an incremental assignment. The address pointed to by dot has its contents incremented by expression *e*.
=- e	Change the value of dot using a decremental assignment. Decrement the contents of the address pointed to by dot by expression *e*.

Commands

A command must be prefixed by a colon (`:`). Only enough letters of the command to uniquely distinguish it are needed. Multiple commands can be entered on one line by separating them by a SPACE, TAB, or semicolon (`;`).

To view a potentially unmounted disk in a reasonable manner, `fsdb` supports the `cd`, `pwd`, `ls`, and `find` commands. The functionality of each of these commands basically matches that of its UNIX counterpart. See `cd(1)`, `pwd(1)`,`ls(1)`, and`find(1)` for details. The *, ,, ?, and - wildcard characters are also supported.

The following commands are supported:

base[=*b*]	Display or set the base. All input and output is governed by the current base. Without the = *b*, displays the current base. Otherwise, sets the current base to *b*. Base is interpreted using the old value of base, so to ensure correctness use the `0`, `0t`, or `0x` prefix when changing the base. The default for base is hexadecimal.
block	Convert the value of dot to a block address.
cd [*dir*]	Change the current directory to directory *dir*. The current values of inode and dot are also updated. If `dir` is not specified, changes directories to inode 2, root (`/`).
directory	If the current inode is a directory, converts the value of dot to a directory slot offset in that directory, and dot now points to this entry.
file	Set the value of dot as a relative block count from the beginning of the file. The value of dot is updated to the first byte of this block.
find *dir* [`-name` *n*] | [`-inum` *i*]	Find files by name or i-number. Recursively searches directory *dir* and below for file names whose i-number matches `i` or whose name matches pattern *n*. Only one of the two options (`-name` or `-inum`) can be used at one time. The find `-print` is not necessary or accepted.
fill=*p*	Fill an area of disk with pattern *p*. The area of disk is delimited by dot and count.
inode	Convert the value of dot to an inode address. If successful, the current value of inode is updated as well as the value of dot. As a convenient shorthand, if `:inode` appears at the beginning of the line, the value of dot is set to the current inode and that inode is displayed in inode format.
ls [`-R`] [`-l`] *pat1 pat2...*	List directories or files. If no file is specified, the current directory is assumed. Either or both of the options can be used (but, if used, must be specified before the filename specifiers). Wild card characters are available and multiple arguments are acceptable. The long listing shows only the i-number and the name; use the inode command with `?i` to get more information.
override	Toggle the value of override. Some error conditions might be overridden if override is toggled to `on`.
prompt "*p*"	Change the `fsdb` prompt to *p*. *p* must be enclosed in quotes.

pwd	Display the current working directory.
quit	Quit `fsdb`.
tag	Convert the value of dot and if this is a valid tag, print the volume structure according to the tag.
!	Escape to the shell.

Inode Commands

In addition to the above commands, several other commands deal with inode fields and operate directly on the current inode (they still require the colon (`:`). They can be used to more easily display or change the particular fields. The value of dot is only used by the `:db` and `:ib` commands. Upon completion of the command, the value of dot is changed so that it points to that particular field. For example,

```
> :ln=+1
```

increments the link count of the current inode and sets the value of dot to the address of the link count field.

The following inode commands are supported:

at	Access time
bs	Block size
ct	Creation time
gid	Group id
ln	Link number
mt	Modification time
md	Mode
maj	Major device number
min	Minor device number
nm	This command actually operates on the directory name field. Once poised at the desired directory entry (using the `directory` command), this command allows you to change or display the directory name. For example, `> 7:dir:nm="foo"` gets the 7th directory entry of the current inode and changes its name to `foo`. Directory names cannot be made larger than the field allows. If an attempt is made to make a directory name larger than the field allows,, the string is truncated to fit and a warning message is displayed.
sz	File size

uid User ID

uniq Unique ID

Formatted Output

Formatted output comes in two styles and many format types. The two styles of formatted output are: structured and unstructured. Structured output is used to display inodes, directories, and so forth. Unstructured output displays raw data.

Format specifiers are preceded by the slash (`/`) or question mark (`?`) character. *type* is updated as necessary upon completion.

The following format specifiers are preceded by the `?` character:

i Display as inodes in the current base.

d Display as directories in the current base.

The following format specifiers are preceded by the `/` character:

b Display as bytes in the current base.

c Display as characters.

o | O Display as octal shorts or longs.

d | D Display as decimal shorts or longs.

x | X Display as hexadecimal shorts or longs.

Examples

EXAMPLE 1 Using fsdb as a calculator for complex arithmetic

The following command displays `2010` in decimal format, and is an example of using `fsdb` as a calculator for complex arithmetic.

```
> 2000+400%(20+20)=D
```

EXAMPLE 2 Using fsdb to display an i-number in idode fomat

The following command displays the i-number `386` in inode format.`386` becomes the current inode.

```
> 386:ino?i
```

EXAMPLE 3 Using fsdb to change the link count

The following command changes the link count for the current inode to `4`.

```
> :ln=4
```

EXAMPLE 4 Using fsdb to increment the link count

The following command increments the link count by 1.

```
> :ln=+1
```

EXAMPLE 5 Using fsdb to display the creation time as a hexadecimal long

The following command displays the creation time as a hexadecimal long.

```
> :ct=X
```

EXAMPLE 6 Using fsdb to display the modification time in time format

The following command displays the modification time in time format.

```
> :mt=t
```

EXAMPLE 7 Using fsdb to display in ASCII

The following command displays, in ASCII, block 0 of the file associated with the current inode.

```
> 0:file/c
```

EXAMPLE 8 Using fsdb to display the directory enteries for the root inode

The following command displays the first block's directory entries for the root inode of this file system. This command stops prematurely if the EOF is reached.

```
> 2:ino,*?d
```

EXAMPLE 9 Using fsdb to change the current inode

The following command changes the current inode to that associated with the 5th directory entry (numbered from 0) of the current inode. The first logical block of the file is then displayed in ASCII.

```
> 5:dir:inode; 0:file,*/c
```

EXAMPLE 10 Using fsdb to change the i-number

The following command changes the i-number for the 7th directory slot in the root directory to 3.

```
> 2:inode; 7:dir=3
```

EXAMPLE 11 Using fsdb to change the name field

The following command changes the *name* field in the directory slot to name.

```
 > 7:dir:nm="name"
```

EXAMPLE 12 Using fsdb to display the a block

The following command displays the 3rd block of the current inode as directory entries.

EXAMPLE 13 Using fsdb to set the contents of address

The following command sets the contents of address `2050` to `0xffffffff`. `0xffffffff` can be truncated, depending on the current type.

```
> 2050=0xffff
```

EXAMPLE 14 Using fsdb to place an ASCII string at an address

The following command places the ASCII string `this is some text` at address `1c92434`.

```
> 1c92434="this is some text"
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWudf

See Also

`clri(1M)`, `fsck_udfs(1M)`, `dir(4)`, `attributes(5)`

Name fsdb_ufs – ufs file system debugger

Synopsis `fsdb -F ufs [`*generic_options*`] [`*specific_options*`] `*special*

Description The `fsdb_ufs` command is an interactive tool that can be used to patch up a damaged UFS file system. It has conversions to translate block and i-numbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an inode. These greatly simplify the process of correcting control block entries or descending the file system tree.

`fsdb` contains several error-checking routines to verify inode and block addresses. These can be disabled if necessary by invoking `fsdb` with the `-o` option or by the use of the `o` command.

`fsdb` reads a block at a time and will therefore work with raw as well as block I/O devices. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block. Note that in order to modify any portion of the disk, `fsdb` must be invoked with the `w` option.

Wherever possible, `adb`-like syntax was adopted to promote the use of `fsdb` through familiarity.

Options The following option is supported:

`-o` Specify UFS file system specific options. These options can be any combination of the following separated by commas (with no intervening spaces). The options available are:

`?`	Display usage
`o`	Override some error conditions
`p='string'`	set prompt to string
`w`	open for write

Usage Numbers are considered hexadecimal by default. However, the user has control over how data is to be displayed or accepted. The `base` command will display or set the input/output base. Once set, all input will default to this base and all output will be shown in this base. The base can be overridden temporarily for input by preceding hexadecimal numbers with '`0x`', preceding decimal numbers with '`0t`', or octal numbers with '`0`'. Hexadecimal numbers beginning with `a-f` or `A-F` must be preceded with '`0x`' to distinguish them from commands.

Disk addressing by `fsdb` is at the byte level. However, `fsdb` offers many commands to convert a desired inode, directory entry, block, superblock and so forth to a byte address. Once the address has been calculated, `fsdb` will record the result in dot (`.`).

Several global values are maintained by `fsdb`:

- the current base (referred to as `base`),

- the current address (referred to as `dot`),
- the current inode (referred to as `inode`),
- the current count (referred to as `count`),
- and the current type (referred to as `type`).

Most commands use the preset value of `dot` in their execution. For example,

```
> 2:inode
```

will first set the value of `dot` to 2, ':', will alert the start of a command, and the `inode` command will set `inode` to 2. A count is specified after a ','. Once set, `count` will remain at this value until a new command is encountered which will then reset the value back to 1 (the default). So, if

```
> 2000,400/X
```

is typed, 400 hex longs are listed from 2000, and when completed, the value of `dot` will be `2000 + 400 * sizeof (long)`. If a RETURN is then typed, the output routine will use the current values of `dot`, `count`, and `type` and display 400 more hex longs. A '*' will cause the entire block to be displayed.

End of fragment, block and file are maintained by `fsdb`. When displaying data as fragments or blocks, an error message will be displayed when the end of fragment or block is reached. When displaying data using the `db`, `ib`, `directory`, or `file` commands an error message is displayed if the end of file is reached. This is mainly needed to avoid passing the end of a directory or file and getting unknown and unwanted results.

An example showing several commands and the use of RETURN would be:

```
> 2:ino; 0:dir?d
      or
> 2:ino; 0:db:block?d
```

The two examples are synonymous for getting to the first directory entry of the root of the file system. Once there, any subsequent RETURN (or +, -) will advance to subsequent entries. Note that

```
> 2:inode; :ls
      or
> :ls /
```

is again synonymous.

Expressions The symbols recognized by `fsdb` are:

RETURN update the value of `dot` by the current value of `type` and display using the current value of `count`.

`#`	numeric expressions may be composed of +, -, *, and % operators (evaluated left to right) and may use parentheses. Once evaluated, the value of `dot` is updated.
`,` *count*	count indicator. The global value of `count` will be updated to `count`. The value of `count` will remain until a new command is run. A count specifier of '*' will attempt to show a *blocks's* worth of information. The default for `count` is 1.
`?`*f*	display in structured style with format specifier *f*. See `FormattedOutput`.
`/`*f*	display in unstructured style with format specifier *f* See `FormattedOutput`.
`.`	the value of `dot`.
`+`*e*	increment the value of `dot` by the expression *e*. The amount actually incremented is dependent on the size of `type`: `dot = dot + e * sizeof (type)` The default for *e* is `1`.
`-`*e*	decrement the value of `dot` by the expression *e*. See `+`.
`*`*e*	multiply the value of `dot` by the expression *e*. Multiplication and division don't use `type`. In the above calculation of `dot`, consider the `sizeof(type)` to be `1`.
`%`*e*	divide the value of `dot` by the expression *e*. See `*`.
`<` *name*	restore an address saved in register *name*. *name* must be a single letter or digit.
`>` *name*	save an address in register *name*. *name* must be a single letter or digit.
`=`*f*	display indicator. If *f* is a legitimate format specifier. then the value of `dot` is displayed using the format specifier *f*. See `FormattedOutput`. Otherwise, assignment is assumed See `=`.
`= [`*s*`] [`*e*`]`	assignment indicator. The address pointed to by `dot` has its contents changed to the value of the expression *e* or to the ASCII representation of the quoted ('') string *s*. This may be useful for changing directory names or ASCII file information.
`=+` *e*	incremental assignment. The address pointed to by `dot` has its contents incremented by expression *e*.
`=-` *e*	decremental assignment. The address pointed to by `dot` has its contents decremented by expression *e*.

Commands A command must be prefixed by a ':' character. Only enough letters of the command to uniquely distinguish it are needed. Multiple commands may be entered on one line by separating them by a SPACE, TAB or ';'.

In order to view a potentially unmounted disk in a reasonable manner, `fsdb` offers the `cd`, `pwd`, `ls` and `find` commands. The functionality of these commands substantially matches those of its UNIX counterparts. See individual commands for details. The '*', '?', and '[-]' wild card characters are available.

`base=b`	display or set base. As stated above, all input and output is governed by the current `base`. If the `=b` is omitted, the current `base` is displayed. Otherwise, the current `base` is set to *b*. Note that this is interpreted using the old value of `base`, so to ensure correctness use the '0', '0t', or '0x' prefix when changing the `base`. The default for `base` is hexadecimal.
`block`	convert the value of `dot` to a block address.
`cd` *dir*	change the current directory to directory *dir*. The current values of `inode` and `dot` are also updated. If no *dir* is specified, then change directories to inode `2` ("/").
`cg`	convert the value of `dot` to a cylinder group.
`directory`	If the current `inode` is a directory, then the value of `dot` is converted to a directory slot offset in that directory and `dot` now points to this entry.
`file`	the value of `dot` is taken as a relative block count from the beginning of the file. The value of `dot` is updated to the first byte of this block.
`find` *dir* [`-name` *n*] [`-inum` *i*]	find files by name or i-number. `find` recursively searches directory `dir` and below for filenames whose i-number matches *i* or whose name matches pattern *n*. Note that only one of the two options (-name or -inum) may be used at one time. Also, the -print is not needed or accepted.
`fill=`*p*	fill an area of disk with pattern *p*. The area of disk is delimited by `dot` and `count`.
`fragment`	convert the value of *dot* to a fragment address. The only difference between the `fragment` command and the `block` command is the amount that is able to be displayed.
`inode`	convert the value of *dot* to an inode address. If successful, the current value of `inode` will be updated as well as the value of *dot*. As a convenient shorthand, if ':inode' appears at the beginning of the line, the value of *dot* is set to the current `inode` and that inode is displayed in inode format.
`log_chk`	run through the valid log entries without printing any information and verify the layout.

`log_delta`	count the number of deltas into the log, using the value of dot as an offset into the log. No checking is done to make sure that offset is within the head/tail offsets.
`log_head`	display the header information about the file system logging. This shows the block allocation for the log and the data structures on the disk.
`log_otodb`	return the physical disk block number, using the value of dot as an offset into the log.
`log_show`	display all deltas between the beginning of the log (BOL) and the end of the log (EOL).
`ls`	[`-R`] [`-l`] *pat1 pat2* . . . list directories or files. If no file is specified, the current directory is assumed. Either or both of the options may be used (but, if used, *must* be specified before the filename specifiers). Also, as stated above, wild card characters are available and multiple arguments may be given. The long listing shows only the i-number and the name; use the `inode` command with '?i' to get more information.
`override`	toggle the value of override. Some error conditions may be overriden if override is toggled on.
`prompt` *p*	change the `fsdb` prompt to *p*. *p* must be surrounded by (")s.
`pwd`	display the current working directory.
`quit`	quit `fsdb`.
`sb`	the value of *dot* is taken as a cylinder group number and then converted to the address of the superblock in that cylinder group. As a shorthand, ':sb' at the beginning of a line will set the value of *dot* to *the* superblock and display it in superblock format.
`shadow`	if the current inode is a shadow inode, then the value of *dot* is set to the beginning of the shadow inode data.
`!`	escape to shell

Inode Commands

In addition to the above commands, there are several commands that deal with inode fields and operate directly on the current `inode` (they still require the ':'). They may be used to more easily display or change the particular fields. The value of *dot* is only used by the '`:db`' and '`:ib`' commands. Upon completion of the command, the value of *dot* is changed to point to that particular field. For example,

```
> :ln=+1
```

would increment the link count of the current `inode` and set the value of *dot* to the address of the link count field.

`at` access time.

`bs` block size.

`ct` creation time.

`db` use the current value of *dot* as a direct block index, where direct blocks number from 0 - 11. In order to display the block itself, you need to 'pipe' this result into the `block` or `fragment` command. For example,

```
> 1:db:block,20/X
```

would get the contents of data block field 1 from the inode and convert it to a block address. 20 longs are then displayed in hexadecimal. See `Formatted`Output.

`gid` group id.

`ib` use the current value of *dot* as an indirect block index where indirect blocks number from 0 - 2. This will only get the indirect block itself (the block containing the pointers to the actual blocks). Use the `file` command and start at block 12 to get to the actual blocks.

`ln` link count.

`mt` modification time.

`md` mode.

`maj` major device number.

`min` minor device number.

`nm` although listed here, this command actually operates on the directory name field. Once poised at the desired directory entry (using the *directory* command), this command will allow you to change or display the directory name. For example,

```
> 7:dir:nm="foo"
```

will get the 7th directory entry of the current `inode` and change its name to foo. Note that names cannot be made larger than the field is set up for. If an attempt is made, the string is truncated to fit and a warning message to this effect is displayed.

`si` shadow inode.

`sz` file size.

`uid` user id.

Formatted Output There are two styles and many format types. The two styles are structured and unstructured. Structured output is used to display inodes, directories, superblocks and the like. Unstructured displays raw data. The following shows the different ways of displaying:

`?`

- `c` display as cylinder groups
- `i` display as inodes
- `d` display as directories
- `s` display as superblocks
- `S` display as shadow inode data

`/`

- `b` display as bytes
- `c` display as characters
- `o O` display as octal shorts or longs
- `d D` display as decimal shorts or longs
- `x X` display as hexadecimal shorts or longs

The format specifier immediately follows the '/' or '?' character. The values displayed by '/b' and all '?' formats are displayed in the current `base`. Also, `type` is appropriately updated upon completion.

Examples

EXAMPLE 1 Displaying in Decimal

The following command displays `2010` in decimal (use of `fsdb` as a calculator for complex arithmetic):

```
> 2000+400%(20+20)=D
```

EXAMPLE 2 Displaying an i-number in Inode Format

The following command displays i-number `386` in an inode format. This now becomes the current `inode`:

```
> 386:ino?i
```

EXAMPLE 3 Changing the Link Count

The following command changes the link count for the current `inode` to `4`:

```
> :ln=4
```

EXAMPLE 4 Incrementing the Link Count

The following command increments the link count by 1:

```
> :ln=+1
```

EXAMPLE 5 Displaying the Creation Time

The following command displays the creation time as a hexadecimal long:

```
> :ct=X
```

EXAMPLE 6 Displaying the Modification Time

The following command displays the modification time in time format:

```
> :mt=t
```

EXAMPLE 7 Displaying in ASCII

The following command displays in ASCII, block zero of the file associated with the current `inode`:

```
> 0:file/c
```

EXAMPLE 8 Displaying the First Block's Worth of Directorty Entries

The following command displays the first block's worth of directory entries for the root inode of this file system. It will stop prematurely if the EOF is reached:

```
> 2:ino,*?d
```

EXAMPLE 9 Displaying Changes to the Current Inode

The following command displays changes the current inode to that associated with the 5th directory entry (numbered from zero) of the current `inode`. The first logical block of the file is then displayed in ASCII:

```
> 5:dir:inode; 0:file,*/c
```

EXAMPLE 10 Displaying the Superblock

The following command displays the superblock of this file system:

```
> :sb
```

EXAMPLE 11 Displaying the Cylinder Group

The following command displays cylinder group information and summary for cylinder group 1:

```
> 1:cg?c
```

EXAMPLE 12 Changing the i-number

The following command changes the i-number for the seventh directory slot in the root directory to 3:

```
> 2:inode; 7:dir=3
```

EXAMPLE 13 Displaying as Directory Entries

The following command displays the third block of the current `inode` as directory entries:

```
> 2:db:block,*?d
```

EXAMPLE 14 Changing the Name Field

The following command changes the name field in the directory slot to *name*:

```
> 7:dir:nm="name"
```

EXAMPLE 15 Getting and Filling Elements

The following command gets fragment `3c3` and fill `20` `type` elements with `0x20`:

```
> 3c3:fragment,20:fill=0x20
```

EXAMPLE 16 Setting the Contents of an Address

The following command sets the contents of address `2050` to `0xffffffff`. `0xffffffff` may be truncated depending on the current `type`:

```
> 2050=0xffff
```

EXAMPLE 17 Placing ASCII

The following command places the ASCII for the string at `1c92434`:

```
> 1c92434="this is some text"
```

EXAMPLE 18 Displaying Shadow Inode Data

The following command displays all of the shadow inode data in the shadow inode associated with the root inode of this file system:

```
> 2:ino:si:ino;0:shadow,*?S
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also clri(1M), fsck_ufs(1M), dir_ufs(4), attributes(5), ufs(7FS)

Warnings Since `fsdb` reads the disk raw, extreme caution is advised in determining its availability of `fsdb` on the system. Suggested permissions are 600 and owned by bin.

Notes The old command line syntax for clearing i-nodes using the ufs-specific `'-z i-number'` option is still supported by the new debugger, though it is obsolete and will be removed in a future release. Use of this flag will result in correct operation, but an error message will be printed warning of the impending obsolesence of this option to the command. The equivalent functionality is available using the more flexible clri(1M) command.

Name fsirand – install random inode generation numbers

Synopsis

```
fsirand [-p] special
```

Description `fsirand` installs random inode generation numbers on all the inodes on device *special*, and also installs a file system ID in the superblock. This helps increase the security of file systems exported by NFS.

`fsirand` must be used only on an unmounted file system that has been checked with `fsck(1M)` The only exception is that it can be used on the root file system in single-user mode, if the system is immediately re-booted afterwards.

Options

`-p` Print out the generation numbers for all the inodes, but do not change the generation numbers.

Usage See `largefile(5)` for the description of the behavior of `fsirand` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `fsck(1M)`, `attributes(5)`, `largefile(5)`

Name fssnap – create temporary snapshots of a file system

Synopsis

```
fssnap [-F FSType] [-V] -o special_options /mount/point

fssnap  -d [-F  FSType] [-V] /mount/point | dev

fssnap -i [-F FSType] [-V] [-o special_options]
     [/mount/point | dev]
```

Description The `fssnap` command creates a stable, read-only snapshot of a file system when given either an active mount point or a special device containing a mounted file system, as in the first form of the synopsis. A snapshot is a temporary image of a file system intended for backup operations.

While the snapshot file system is stable and consistent, an application updating files when the snapshot is created might leave these files in an internally inconsistent, truncated, or otherwise unusable state. In such a case, the snapshot will contain these partially written or corrupted files. It is a good idea to ensure active applications are suspended or checkpointed and their associated files are also consistent during snapshot creation.

File access times are not updated while the snapshot is being created.

A path to the virtual device that contains this snapshot is printed to standard output when a snapshot is created.

Options The following options are supported:

`-d`	Deletes the snapshot associated with the given file system.
`-F` *FSType*	Specifies the file system type to be used. The `FSType` should either be specified here or be determined by matching the block special device with an entry in the `/etc/vfstab` table, or by consulting `/etc/default/fs`.
`-i`	Displays the state of any given `FSType` snapshot. If a mount-point or device is not given, a list of all snapshots on the system is displayed. When a mount-point or device is specified, detailed information is provided for the specified file system snapshot by default. The format and meaning of this information is file-system dependent. See the `FSType`-specific `fssnap` man page for details.
`-o` *special_options*	See the `FSType`-specific man page for `fssnap`.
`-V`	Echoes the complete command line, but does not execute the command.

Operands The following operands are supported:

/mount/point	The directory where the file system resides.

Examples See FSType-specific man pages for examples.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/vfstab`	Specifies file system type.
`/etc/default/fs`	Specifies the default local file system type.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `fssnap_ufs(1M)`, `attributes(5)`

Notes This command might not be supported for all FSTypes.

Name fssnap_ufs – create a temporary snapshot of a UFS file system

Synopsis

```
fssnap [-F ufs] [-V] -o backing-store=path,
     [specific-options] /mount/point

fssnap -d [-F ufs] [-V] /mount/point | dev

fssnap -i [-F ufs] [-V] [-o specific-options] /mount/point | dev
```

Description The `fssnap` command queries, creates, or deletes a temporary snapshot of a UFS file system. A snapshot is a point-in-time image of a file system that provides a stable and unchanging device interface for backups.

When creating a file system snapshot, you must specify the file system to be captured and the backing-store file. The backing-store file(s) are where the snapshot subsystem saves old file system data before it is overwritten. Beyond the first backing-store file, `fssnap` automatically creates additional backing-store files on an as-needed basis.

The number and size of the backing store files varies with the amount of activity in the file system. The destination path must have enough free space to hold the backing-store file(s). This location must be different from the file system that is being captured in a snapshot. The backing-store file(s) can reside on any type of file system, including another UFS file system or an NFS–mounted file system.

Options The following options are supported:

`-d`
Deletes the snapshot associated with the given file system.

`-i`
Displays the state of one or all UFS snapshots. If a mount-point or device is not specified, a list of all snapshots on the system is displayed. When a mount-point or device is specified, detailed information is provided for the specified file system snapshot by default.

Use the `-o` options with the `-i` option to specify what snapshot information is displayed. Since this feature is provided primarily for use in scripts and on the command line, no labels are displayed for the data. Sizes are all in bytes, and the output is not internationalized or localized. The information is displayed on one line per option. Unrecognized options display a single `?` on the line. One line per option guarantees that there are the same number of lines as options specified and there is a one–to-one correspondence between an output line and an option.

The following `-o` options display specific information for a given snapshot. See the EXAMPLES section for examples of how to use these options.

`snapnumber`
Display the snapshot number.

`blockdevname`
Display the block device path.

`rawdevname`
Display the raw device path.

`mountpoint`
Display the mount point of the master file system.

`state`
Display the state of the snapshot device.

`backing-store`
Display the location of the first backing-store file for this snapshot. If there are multiple backing-store files, subsequent files have the same name as the first file, with the suffixes `.2`, `.3`, and so forth.

`backing-store-len`
Display the sum of the sizes of the backing-store files.

`maxsize`
Display the `maxsize` value specified for the backing-store file(s).

`createtime`
Display the time that the snapshot was created.

`chunksize`
Display the copy-on-write granularity.

`-o` *specific-options*
Without `-d` or `-i`, the default action is to create a snapshot. Specify the following options when creating a snapshot. All of these options are discretionary, except for the backing-store file, which is required.

`backing-store=`*path*
Uses *path* in the creation of the backing-store file(s). *path* must not reside on the file system that is being captured in a snapshot and must not be the name of an existing file. If *path* is a directory, then a backing-store file is created within it using a name that is generated automatically. If *path* is not a directory and does not already exist, then a backing-store file with that name is created. If more than one backing-store file is required, `fssnap` creates subsequent files automatically. The second and subsequent files have the same name as the first file, with suffixes of `.2`, `.3`, and so forth.

This option can be abbreviated as `bf=`*path* or `bs=`*path*.

`unlink`
Unlinks the backing-store file after the snapshot is created. This option specifies that the backing-store file does not need to be removed manually when the snapshot is deleted. This might make administration more difficult since the file is not visible in the file system. If this option is not specified, the backing-store files should be removed manually after the snapshot is deleted.

`chunksize=`*n* [`k`,`m`,`g`]
Uses *n* for the chunk size. Chunk size is the granularity of the data that is sent to the backing store.

Specify `chunksize` in the following units: `k` for kilobytes, `m` for megabytes, or `g` for gigabytes. By default, chunk size is four times the block size of the file system (typically `32k`).

`maxsize=`*n*[`k`,`m`,`g`]
Does not allow the sum of the sizes of the backing-store file(s) to exceed *n*, where *n* is the unit specified. The snapshot is deleted automatically when the sum of the sizes of the backing-store file(s) exceeds `maxsize`.

Specify `maxsize` in the following units: `k` for kilobytes, `m` for megabytes, or `g` for gigabytes.

`raw`
Displays to standard output the name of the raw device instead of the block device when a snapshot is created. The block device is printed by default (when `raw` is not specified). This option makes it easier to embed `fssnap` commands in the command line for commands that require the raw device instead. Both devices are always created. This option affects only the output.

Operands The following operands are supported:

mount-point
The directory where the file system resides.

special
The physical device for the file system, such as `/dev/dsk/c0t0d0s7`.

Examples **EXAMPLE 1** Creating a Snapshot of a File System

The following example creates a snapshot of a file system. The block special device created for the snapshot is `/dev/fssnap/0`.

```
# fssnap -F ufs -o backing-store=/var/tmp /export/home
/dev/fssnap/0
```

EXAMPLE 2 Backing Up a File System Snapshot Without Having To Unmount the File System

The following example backs up a file system snapshot without having to unmount the file system. Since `ufsdump` requires the path to a raw device, the `raw` option is used. The `/export/home` file system snapshot is removed in the second command.

```
# ufsdump 0uf /dev/rmt/0 ‘fssnap -F ufs
      -o raw,bs=/export/snap /export/home‘
<output from ufsdump>
# fssnap -F ufs -d /export/home
```

EXAMPLE 3 Backing Up a File System

When backing up a file system, do not let the backing-store file(s) exceed `400` Mbytes. The second command removes the `/export/home` file system snapshot.

```
# ufsdump 0uf /dev/rmt/0 ‘fssnap -F ufs
      -o maxsize=400m,backing-store=/export/snap,raw
      /export/home‘
# fssnap -F ufs -d /export/home
```

EXAMPLE 4 Performing an Incremental Dump of a Snapshot

The following example uses `ufsdump` to back up a snapshot of `/var`. Note the use of the `N` option to `ufsdump`, which writes the name of the device being dumped, rather than the name of the snapshot device, to `/etc/dumpdates` file. See ufsdump(1M) for details on the `N` flag.

```
# ufsdump lfNu /dev/rmt/0 /dev/rdsk/c0t3d0s2 ‘fssnap -F ufs
-o raw,bs=/export/scratch,unlink /var‘
```

EXAMPLE 5 Finding Out What Snapshots Currently Exist

The following command displays the currently existing snapshots.

```
# fssnap -i
0  /src
1  /export/home
<output continues>
```

EXAMPLE 6 Mounting a File System Snapshot

The following example creates a file system snapshot. After you create a file system snapshot, mount it on `/tmp/mount` for temporary read-only access.

```
# fssnap -F ufs -o backing-store=/nfs/server/scratch /export/home
/dev/fssnap/1
# mkdir /tmp/mount
# mount -F ufs -o ro /dev/fssnap/1 /tmp/mount
```

EXAMPLE 7 Creating a File System Snapshot and Unlinking the Backing-store File

The following example creates a file system snapshot and unlinks the backing-store file. After creating a file system snapshot and unlinking the backing-store file, check the state of the snapshot.

```
# fssnap -o bs=/scratch,unlink /src
/dev/fssnap/0
# fssnap -i /src
Snapshot number              : 0
Block Device                 : /dev/fssnap/0
Raw Device                   : /dev/rfssnap/0
Mount point                  : /src
Device state                 : active
```

EXAMPLE 7 Creating a File System Snapshot and Unlinking the Backing-store File *(Continued)*

```
Backing store path            : /scratch/snapshot2 <UNLINKED>
Backing store size            : 192 KB
Maximum backing store size    : Unlimited
Snapshot create time          : Sat May 06 10:55:11 2000
Copy-on-write granularity     : 32 KB
```

EXAMPLE 8 Displaying the Size and Location of the Backing-store File(s) and the Creation Time for the Snapshot

The following example displays the size of the backing-store file(s) in bytes, the location of the backing store, and the creation time for the snapshot of the `/test` file system.

```
# fssnap -i -o backing-store-len,backing-store,createtime /test
196608
/snapshot2
Sat May 6 10:55:11 2000
```

Note that if there are multiple backing-store files stored in `/snapshot2`, they will have names of the form *file* (for the first file), *file*.1, *file*.2, and so forth.

Exit Status

The following exit values are returned:

`0`
Successful completion.

`>0`
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

The script-readable output mode is a stable interface that can be added to, but will not change. All other interfaces are subject to change.

See Also

`xntpd(1M)`, `mlock(3C)`, `attributes(5)`

Notes

The `fssnap` device files should be treated like a regular disk block or character device.

The association between a file system and the snapshot is lost when the snapshot is deleted or the system reboots. Snapshot persistence across reboots is not currently supported.

To avoid unnecessary performance impacts, perform the snapshot and system backup when the system is least active.

It is not possible to perform a snapshot of a file system if any of the following conditions are true:

- The file system is in use by system accounting
- The file system contains a local swap file
- The file system is used as backing store by an application that uses `mlock(3C)` to lock its pages. Typically, these are real time applications, such as `xntpd(1M)`.

These conditions result in `fssnap` being unable to write lock the file system prior to performing the snapshot.

Name fsstat – report file system statistics

Synopsis

```
fsstat [-a|f|i|n|v] [-T | u|d] {-F | {fstype|path}...}
     [interval [count]]
```

Description `fsstat` reports kernel file operation activity by the file system type (*fstype*) or by the path name, which is converted to a mount point. The first set of lines of output reports all activity since:

- The file system module was loaded (in the case of *fstype*)
- The file system was mounted (in the case of mount point)

Statistics are gathered at the file system independent layer at both the *fstype* and the mount point levels. However, not all file system types are represented in the gathering of statistics. (See the NOTES section of this man page.)

The output of `fsstat` is dependent on the mode (option) requested. All statistic fields are displayed using "smart numbers" which automatically scale the units in a human readable form that fits in a maximum of 5 characters. For example:

100	is displayed as 100
2048	is displayed as 2K
3000000	is displayed as 2.86M

The unit modifiers are: K (Kbyte), M (Mbyte), G (Gbyte), T (terabyte), P (petabyte), and E (exabyte).

During the execution of `fsstat`, the state of the system can change. If relevant, a state change message is included in the `fsstat` output in one of the following forms:

```
<<mount point no longer available: {path}>>
<<file system module no longer loaded: {fstype}>>
```

After the state change messages are displayed, `fsstat` continues to display the statistics as directed. If all of the *fstypes* and mount points that `fsstat` was reporting on are no longer available, then `fsstat` exits.

The user is required to specify the `-F` option (all available file system types) or a list of one or more *fstypes* and/or mount points.

The default report shows general file system activity. This display combines similar operations into general categories as follows:

new file	Number of creation operations for file system objects (for example, files, directories, symlinks, etc.)
name remov	Number of name removal operations
name chng	Number of name change operations

attr get	Number of object attribute retrieval operations
attr set	Number of object attribute change operations
lookup ops	Number of object lookup operations
rddir ops	Number of read directory operations
read ops	Number of data read operations
read bytes	Bytes transferred by data read operations
write ops	Number of data write operations
write bytes	Bytes transferred by data write operations

The entity being reported on (*fstype* or mount point) is displayed in the last column.

Options The following options are supported:

`-a` Report the activity for kernel attribute operations. The following statistics are reported:

getattr	Number of file attribute retrieval calls
setattr	Number of file attribute modification calls
getsec	Number of file security attribute retrieval calls
setsec	Number of file security attribute modification calls

The entity being reported on (*fstype* or mount point) is displayed in the last column.

`-f` Report the full activity for all kernel file operations. Each file operation is listed in the left column. The following statistics are reported for each operation:

#ops	Number of calls for this operation
bytes	Average transfer size in bytes (only applies to read, write, readdir)

The entity being reported on (*fstype* or mount point) is displayed in the first row.

`-i` Reports the activity for kernel I/O operations. The following statistics are reported:

read ops	Number of data read calls
read bytes	Number of bytes read
write ops	Number of data write calls
write bytes	Number of bytes written
rddir ops	Number of read directory calls

rddir bytes Number of bytes read by reading directories

rwlock ops Number of internal file system lock operations

rwulock ops Number of internal file system unlock operations

The entity being reported on (*fstype* or mount point) is displayed in the last column.

`-n` Reports the activity for kernel naming operations. The following statistics are reported:

lookup Number of file name retrieval calls

creat Number of file creation calls

remov Number of file remove calls

link Number of link calls

renam Number of file renaming calls

mkdir Number of directory creation calls

rmdir Number of directory removal calls

rddir Number of directory read calls

symlink Number of symlink creation calls

rdlink Number of symlink read calls

The entity being reported on (*fstype* or mount point) is displayed in the last column.

`-v` Reports the activity for calls to the virtual memory operations. The following statistics are reported.

map Number of calls mapping a file

addmap Number of calls setting additional mapping to a mapped file

delmap Number of calls deleting mapping to a file

getpag Number of calls retrieving a page of data from a file

putpag Number of calls writing a page of data to a file

pagio Number of calls to transfer pages in file system swap files

The entity being reported on (fstype or mount point) is displayed in the last column.

`-F` Report on all available file system types.

`-T` *u*|*d* Display a time stamp.

Specify *u* for a printed representation of the internal representation of time (see `time(2)`) Specify *d* for the standard date format. (See `date(1)`). The time stamp is only used when an interval is set.

Operands The following operands are supported:

count Display only *count* reports.

fstype Explicitly specify the file system type(s) to be reported. The file system module must be loaded.

interval Report once each *interval* seconds.

path Specify the path(s) of the mount point(s) to be reported. If path is not a mount point, the mount point containing path will be determined and displayed in the output.

If no *interval* and no *count* are specified, a single report is printed and `fsstat` exits. If an *interval* is specified but no *count* is specified, `fsstat` prints reports every *interval* seconds indefinitely until the command is interrupted.

Examples **EXAMPLE 1** Displaying General Activity

The following example shows general activity for all file system types.

```
$ fsstat -F
 new  name   name  attr   attr lookup rddir  read read  write write
 file remov  chng   get    set    ops   ops   ops bytes   ops bytes
  313K  214K 38.5K 2.16M 56.2K  8.36M 52.8K 19.7M 39.9G 18.8M 39.1G ufs
     0     0     0 2.95K     0  3.81K   282 2.52K  466K     0     0 proc
     0     0     0     0     0      0     0     0     0     0     0 nfs
    10     8     2    86     9     98    15   413  103M 8.43K 1.05G zfs
    13    14     4    98    16    125    10 1.01K  258M 15.9K  127M lofs
8.73K 3.29K 5.25K 55.3K    37  1.20M    44 37.9K 38.3M 47.2K 35.9M tmpfs
     0     0     0 4.93K     0      0     0 1.08K  913K     0     0 mntfs
     3     2     1   503     3    897    13   122 25.8K   128  272K nfs3
    10     8     0   615    10  10.1K    18    61 45.6K   292 2.26M nfs4
```

EXAMPLE 2 Displaying Naming Activity

The following example shows the naming activity for ufs, nfs, nfs3, nfs4, and tmpfs:

```
$ fsstat -n ufs nfs nfs3 nfs4 tmpfs
lookup creat remov  link renam mkdir rmdir rddir symlnk rdlnk
3.57M  3.10K   586     6    24   115   100 30.2K      5  330K ufs
    0      0     0     0     0     0     0     0      0     0 nfs
18.3K      3     5     0     0     0     0 1.03K      2   346 nfs3
  535      0     0     0     0     0     0    46      0     4 nfs4
```

EXAMPLE 2 Displaying Naming Activity *(Continued)*

```
  146     24    15     0     0     4     0     4      0     0 tmpfs
```

EXAMPLE 3 Displaying Attribute Activity

The following example shows the attribute activity for the FS type ufs and the mounted file systems "/" and "/export/home" every three seconds for every third iteration:

```
# fsstat -a ufs / /export/home 3 3
getattr setattr getsec setsec
  378K    91.9K  11.8K      0 ufs
  367K    82.3K  11.6K      0 /
 11.3K     9.6K    198      0 /export/home
 4.97K    2.27K    163      0 ufs
 3.94K    1.36K    162      0 /
 1.03K      927      1      0 /export/home
 2.30K    1.06K     73      0 ufs
 1.95K      766     71      0 /
   361      317      2      0 /export/home
 2.33K    1.06K     78      0 ufs
 1.64K      451     77      0 /
   711      631      1      0 /export/home
```

EXAMPLE 4 Displaying File Operation Statistics

The following example shows the statistics for each file operation for "/" (using the `-f` option):

```
$ fsstat -f /
Mountpoint: /
 operation  #ops  bytes
      open 8.54K
     close  9.8K
      read 43.6K  65.9M
     write 1.57K  2.99M
     ioctl 2.06K
     setfl     4
   getattr 40.3K
   setattr    38
    access 9.19K
    lookup  203K
    create   595
    remove    56
      link     0
    rename     9
     mkdir    19
     rmdir     0
   readdir 2.02K  2.27M
   symlink     4
```

EXAMPLE 4 Displaying File Operation Statistics *(Continued)*

```
  readlink 8.31K
     fsync   199
  inactive 2.96K
       fid     0
     rwlock 47.2K
   rwunlock 47.2K
      seek 29.1K
       cmp 42.9K
     frlock 4.45K
      space     8
     realvp 3.25K
    getpage  104K
    putpage 2.69K
        map 13.2K
     addmap 34.4K
     delmap 33.4K
       poll   287
       dump     0
  pathconf    54
    pageio     0
   dumpctl     0
   dispose 23.8K
getsecattr   697
setsecattr     0
   shrlock     0
   vnevent     0
```

Environment Variables See `environ(5)` for descriptions of the following environment variables that affect the execution of `fsstat`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, `LC_TIME`, and `NLSPATH`.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` A fatal error occurred. A fatal error could be a failed system call or another internal error.

`2` Invalid command-line options were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	Enabled

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	See below.

The command-line options are Unstable. The human-readable output is not considered an interface.

See Also `date(1)`, `time(2)`, `attributes(5)`

Notes All display options (`-a`, `-f`, `-i`, `-n`, `-v`) are mutually exclusive. Entering more than one of these options will result in an error.

The *fstype* and *path* operands must appear after the option, but before the *interval* or *count* on the command line. For example, "`fsstat -a` *fstype interval*". Preference is given to *fstype* so that if a user wishes to see the statistics for a directory that has the same name as an *fstype* (for example, ufs), then the path must be specified unambiguously (for example, ./ufs). Similarly, in order to define a file with a numeric name (for example, "10") from an interval or count operand, the name should be prefixed accordingly (for example, ./10).

When an interval is used, headers repeat after more than 12 lines of statistics have been displayed and the set of lines to be displayed in the current interval have completed.

Statistics are not displayed for all pseudo-filesystems. The output displayed with the `-F` option shows which of the loaded filesystem types are supported.

Unbundled file systems may not be recognized by `fsstat`.

The command-line options are classified as Unstable and could change. The output is not considered to be an interface. The construction of higher level software tools depend on either the command-line options or the output of `fsstat` is not recommended.

Name fstyp – determine file system type

Synopsis

```
fstyp [-v] special
```

Description `fstyp` allows the user to determine the file system type of unmounted file systems using heuristic programs.

An `fstyp` module for each file system type to be checked is executed; each of these modules applies an appropriate heuristic to determine whether the supplied *special* file is of the type for which it checks. If it is, the program prints on standard output the usual file system identifier for that type (for example, "ufs") and exits with a return code of `0`; if none of the modules succeed, the error message `unknown_fstyp (no matches)` is returned and the exit status is `1`. If more than one module succeeds, the error message `unknown_fstyp (multiple matches)` is returned and the exit status is `2`.

This command is unreliable and its results should not be used to make any decisions about subsequent use of a storage device or disk partition.

Options

`-v` Produce verbose output. This is usually information about the file systems superblock and varies across different *FSTypes*. See `ufs`(7FS), `mkfs_ufs`(1M), and `tunefs`(1M) for details.

Usage See `largefile`(5) for the description of the behavior of `fstyp` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `mkfs_ufs`(1M), `tunefs`(1M), `attributes`(5), `largefile`(5), `hsfs`(7FS), `ufs`(7FS), `pcfs`(7FS)

Notes The use of heuristics implies that the result of `fstyp` is not guaranteed to be accurate.

This command is unreliable and its results should not be used to make any decisions about subsequent use of a storage device or disk partition.

Name ftpaddhost – set up a virtual FTP host

Synopsis

```
ftpaddhost -c | -l [-b] [-x xferlog] hostname root_dir
```

Description The `ftpaddhost` script is executed by the super user to set up virtual FTP hosts. The `ftpaddhost` command configures the virtual host *hostname* under directory *root_dir*. The value of *hostname* can be an IP address or the name of a host.

Options The `ftpaddhost` script supports the following options:

`-b` Create a banner for the virtual host. This option is useful to confirm that the virtual host is working.

`-c` Configure complete virtual hosting. This option allows each virtual host to have its own version of the `ftpaccess`, `ftpconversions`, `ftpgroups`, `ftphosts`, and `ftpusers` files. The master version of each of these configuration files is copied from the `/etc/ftpd` directory and placed in the `/etc/ftpd/virtual-ftpd/`*hostname* directory. If the `/etc/ftpusers` file exists it is appended to the virtual `ftpusers` file. If a virtual host lacks its own version of a configuration file, the master version is used.

`-l` Configure limited virtual hosting. This option allows a small number of parameters to be configured differently for a virtual host. See the `virtual` keyword on the `ftpaccess(4)` manual page.

`-x` *xferlog* Create a logfile entry such that the transfer logs for the virtual host are written to the specified file. An absolute path must be specified for the *xferlog* file.

Operands The following operands are supported:

hostname The host name or IP address of the virtual server.

root_dir The absolute pathname of the directory under which the virtual server is set up.

Exit Status The following exit values are returned:

`0` Successful completion

`1` Improper usage of the command

`2` Command failed

Files `/etc/ftpd/virtual-ftpd/`*hostname* The configuration files directory for the virtual host *hostname*.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also ftpconfig(1M), in.ftpd(1M), ftpaccess(4), ftpconversions(4), ftpgroups(4), ftphosts(4), ftpusers(4), xferlog(4), attributes(5)

Name ftpconfig – set up anonymous FTP

Synopsis

```
ftpconfig [ftpdir]
ftpconfig -d ftpdir
```

Description The `ftpconfig` script is executed by the super user to set up anonymous FTP. Anonymous FTP allows users to remotely log on to the FTP server by specifying the user name `ftp` or `anonymous` and the user's email address as password. The anonymous users are logged on to the FTP Server and given access to a restricted file area with its own file system root. See `chroot(2)`. The FTP area has its own minimal system files.

This command will copy and set up all the components needed to operate an anonymous FTP server, including creating the *ftp* user account, creating device nodes, copying `/usr/lib` files, and copying timezone data. The `passwd` and `group` files set up have been stripped down to prevent malicious users from finding login names on the server. The anonymous file area will be placed under `ftpdir`. If the *ftp* user account already exists, then the current FTP area is used, and the system files in it are updated. All other files are left untouched. This command should be run to update the anonymous FTP area's configuration whenever a system patch is installed, or the system is upgraded.

Options

`-d` Create a new or update an existing *ftpdir* without creating or updating the *ftp* user account. Use this option when creating guest FTP user accounts.

Operands The following operands are supported:

ftpdir The absolute pathname of the directory under which the anonymous FTP area is set up.

Exit Status The following exit values are returned:

`0` Successful completion

`1` Improper usage of the command

`2` Command failed

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu
Interface Stability	Evolving

See Also `ftpaddhost(1M)`, `in.ftpd(1M)`, `useradd(1M)`, `chroot(2)`, `attributes(5)`

Name ftprestart – restart previously shutdown FTP Servers

Synopsis

```
ftprestart [-V]
```

Description Use the `ftprestart` command to restart an FTP Server previously shut down by means of `ftpshut(1M)`. The `ftprestart` command reads the `shutdown` capability from the `ftpaccess(4)` file to determine the path of the shutdown message files. It then reenables the FTP Server by removing any shutdown message files in the anonymous and virtual FTP Server area, as well as the system wide shutdown message file.

Options The `ftprestart` command supports the following options:

`-V` Display program copyright and version information, then terminate.

Examples **EXAMPLE 1** Sample Output from `ftprestart`

The following example shows sample output from the `ftprestart` command:

```
example% ftprestart
ftprestart: /export/home/ftp/etc/ftpd/shutdown.msg removed.
ftprestart: /export/home/virtual1/etc/ftpd/shutdown.msg removed.
ftprestart: /etc/ftpd/shutdown.msg removed.
```

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/ftpd/ftpaccess`

`/etc/ftpd/ftpservers`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu
Interface Stability	External

See Also `ftpshut(1M)`, `in.ftpd(1M)`, `ftpaccess(4)`, `ftpservers(4)`, `attributes(5)`

Name ftpshut – close down the FTP Servers at a given time

Synopsis

```
ftpshut [-V] [-v] [-l min] [-d min] time
     [warning-message]...
```

Description The `ftpshut` command provides an automated shutdown procedure that the superuser can use to notify FTP users when the FTP Server is shutting down.

Ten minutes before shutdown, or immediately if the value of *time* is less than ten minutes, any new FTP Server connections will be disabled. You may adjust the shutdown of new FTP Server connections by means of the `-l` option.

Five minutes before shutdown, or immediately if the value of *time* is less than five minutes, all current FTP connections will be disconnected. You may adjust the shutdown of current FTP connections by means of the `-d` option.

The `ftpshut` command creates shutdown message files that the FTP Server uses to determine when to shutdown. Separate shutdown message files are created in the anonymous and virtual host FTP Server areas, in addition to the system wide shutdown message file. Once the shutdown occurs, the server continues to refuse connections until the appropriate shutdown message file is removed. This normally is done by using the `ftprestart(1M)` command. The location of the shutdown message file is specified by the `shutdown` capability in the `ftpaccess` file.

The following magic cookies are available:

`%s` The time system is going to shut down.

`%r` The time new connections will be denied.

`%d` The time current connections will be dropped.

`%C` The current working directory.

`%E` The maintainer's email address as defined in the `ftpaccess` file.

`%F` The free space in the partition of CWD, in kilobytes.

`%L` The local host name.

`%M` The maximum allowed number of users in this class.

`%N` The current number of users in this class.

`%R` The remote host name.

`%T` The local time (form Thu Nov 15 17:12:42 1990).

`%U` The username given at login time.

Options The `ftpshut` command supports the following options:

`-V` Display program copyright and version information, then terminate.

`-d` *min* The time ahead of shutdown, in minutes, that existing connections will be disconnected upon completion of their current or next (if idle) FTP request.

`-l` *min* The time ahead of shutdown, in minutes, that new connections will be refused.

`-v` Verbose. Output the pathname of the shutdown message files created.

Operands The `ftpshut` command supports the following operands:

time The *time* at which `ftpshut` will bring the FTP Servers down. *time* can have a value of `now`, which indicates an immediate shutdown. Alternatively, *time* can specify a future time in one of two formats: +*number* or *HHMM*. The first form brings the FTP Server down in *number* minutes. The second brings the FTP Server down at the time of day indicated, using a 24-hour clock format. When using the absolute time format, you can only specify times between now and 23:59.

warning-message The message to display that warns of the imminent shutdown. The *warning-message* will be formatted at 70 characters wide. `ftpshut` knows the actual string length of the magic cookies. If no warning-message is supplied, the default message "`System shutdown at %s`" is used.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/etc/ftpd/ftpaccess`

`/etc/ftpd/ftpservers`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu
Interface Stability	External

See Also `in.ftpd(1M)`, `ftprestart(1M)`, `shutdown(1M)`, `ftpaccess(4)`, `ftpservers(4)`, `attributes(5)`

Name fuser – identify users of files and devices

Synopsis

```
/usr/sbin/fuser [-c | -d | -f] [-nu] [-k | -s sig] files
     [ [- ] [-c | -d | -f] [-nu] [-k | -s sig] files] ...
```

Description The `fuser` utility displays the process IDs of the processes that are using the *files* specified as arguments.

Each process ID is followed by a letter code. These letter codes are interpreted as follows. If the process is using the file as

`c` Indicates that the process is using the file as its current directory.

`m` Indicates that the process is using a file mapped with `mmap(2)`. See `mmap(2)` for details.

`n` Indicates that the process is holding a non-blocking mandatory lock on the file.

`o` Indicates that the process is using the file as an open file.

`r` Indicates that the process is using the file as its root directory.

`t` Indicates that the process is using the file as its text file.

`y` Indicates that the process is using the file as its controlling terminal.

For block special devices with mounted file systems, all processes using any file on that device are listed. For all types of files (text files, executables, directories, devices, and so forth), only the processes using that file are reported.

For all types of devices, `fuser` also displays any known kernel consumers that have the device open. Kernel consumers are displayed in one of the following formats:

[*module_name*]
[*module_name*,dev_path=*path*]
[*module_name*,dev=(*major*,*minor*)]
[*module_name*,dev=(*major*,*minor*),dev_path=*path*]

If more than one group of files are specified, the options may be respecified for each additional group of files. A lone dash cancels the options currently in force.

The process IDs are printed as a single line on the standard output, separated by spaces and terminated with a single new line. All other output is written on standard error.

Any user can run fuser, but only the superuser can terminate another user's process.

Options The following options are supported:

`-c` Reports on files that are mount points for file systems, and any files within that mounted file system.

`-d` Report device usage information for all minor nodes bound to the same device node as the specified minor node. This option does not report file usage for files within a mounted file system.

`-f` Prints a report for the named file, not for files within a mounted file system.

`-k` Sends the `SIGKILL` signal to each process. Since this option spawns kills for each process, the kill messages may not show up immediately (see `kill(2)`). No signals will be sent to kernel file consumers.

`-n` Lists only processes with non-blocking mandatory locks on a file.

`-s` *sig* Sends a signal to each process. The *sig* option argument specifies one of the symbolic names defined in the `<signal.h>` header, or a decimal integer signal number. If *sig* is a symbolic name, it is recognized in a case-independent fashion, without the `SIG` prefix. The `-k` option is equivalent to `-s KILL` or `-s 9`. No signals will be sent to kernel file consumers.

`-u` Displays the user login name in parentheses following the process ID.

Examples

EXAMPLE 1 Reporting on the Mount Point and Files

The following example reports on the mount point and files within the mounted file system.

```
example% fuser -c /export/foo
```

EXAMPLE 2 Restricting Output when Reporting on the Mount Point and Files

The following example reports on the mount point and files within the mounted file system, but the output is restricted to processes that hold non-blocking mandatory locks.

```
example% fuser -cn /export/foo
```

EXAMPLE 3 Sending SIGTERM to Processes Holding a Non-blocking Mandatory Lock

The following command sends `SIGTERM` to any processes that hold a non-blocking mandatory lock on file `/export/foo/my_file`.

```
example% fuser -fn -s term /export/foo/my_file
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `fuser`: `LANG`, `LC_ALL` `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Standard

See Also ps(1), mount(1M), kill(2), mmap(2), signal(3C), attributes(5), environ(5), standards(5)

Notes Because `fuser` works with a snapshot of the system image, it may miss processes that begin using a file while `fuser` is running. Also, processes reported as using a file may have stopped using it while `fuser` was running. These factors should discourage the use of the `-k` option.

Name fwflash – firmware query and update utility

Synopsis

```
/usr/sbin/fwflash  [-l [-c device_class | ALL ]]
     | [-v] | [-h]

fwflash  [-f file | -r file]
     [-y] [-d dev_spec]
```

Description The `fwflash` command writes a binary image file to flash devices of an HBA or HCA device. It also provides the ability to read firmware to a file if supported by the device. Because changing the firmware in a device can have significant impact on the stability of a system, only users with the privilege `All` are allowed to execute this command. Users authorized to run `fwflash` can be granted the "Firmware Flash Update" Rights Profile.

The first form of the command, above, provides information about devices. It lists all devices currently available on the system and supported by `fwflash` for firmware upgrade. You can filter the list operation, to display only specified classes of devices. The second form of the command provides the operations to read or write the firmware images to specific devices.

Options The following options are supported:

`-c` *device_class*
: An optional parameter, valid only when used with the `-l` option. This option causes the command to list only devices of a specific class type. Currently supported classes are `IB` or `ALL`. If `-c` is not specified for the `-l`option, the class defaults to `ALL`.

`-d` *dev_spec*
: The *dev_spec* is an identifier of the device that the user wants to modify with the `-f`or `-r` operation. *dev_spec* can be either the absolute path name as displayed in the `-l` listing or as the device number from the same listing. If the device cannot be found, the command fails.

`-f` *file*
: Specify the path to a binary firmware file you want to write to the device. `fwflash` will verify that the file is a valid firmware binary for the *dev_spec* specified. If it is not, the command fails with an appropriate error message.

`-h`
: List the command line help for `fwflash`.

`-l`
: List the devices on a system available for firmware upgrade and display information specific to each device or device class.

  For InfiniBand (IB) devices, the list operation displays the `guids` (Globally Unique Identifier) currently set for the HCA, as well as the current firmware revision installed. There are four separate `guids` on the HCA; two of them can be set with the same value.

`-r` *file*
: Specify the path to a file to create when reading the firmware from the device. The `-f` and `-r` options are mutually exclusive.

`-v`
Output `fwflash` version information and exit.

`-y`
Valid during an `-f` or `-r` operation, causes `fwflash` not to prompt for confirmation during operation and operate non-interactively.

Examples

EXAMPLE 1 Entering Command Without Arguments

The following command shows `fwflash` when the command is entered without arguments.

```
example# fwflash
Usage: fwflash [-l [-c <device_class> | ALL]] | [-v] | [-h]
               [-f <file> | -r <file>]  [-y] [-d <dev_spec>]
```

EXAMPLE 2 Listing Devices Available to Flash

The following command lists the devices available to be flashed.

```
example# fwflash -l
List of available devices:
Device[0],  /devices/pci@0,0/pci8086,3595@2/pci8086,32a@0,2/\
           pci15b3,5a46@c/pci15b3,5a44@0:devctl
    Class [IB]
        GUID: System Image - 0002c901081e33b3
              Node         - 0000000000003446
              Port 1       - 0002c901081e33b1
              Port 2       - 0002c901081e33b2
        Firmware revision: 3.3.0002
        No HW information available

Device[1],  /devices/pci@0,0/pci8086,3597@4/pci15b3,6278@0:devctl
    Class [IB]
        GUID: System Image - 0002c9010a99e3b3
              Node         - 0002c9010a99e3b0
              Port 1       - 0002c9010a99e3b1
              Port 2       - 0002c9010a99e3b2
        Firmware revision: 4.8.00c8
        Product           : MTLP25208-CF256T (Lion cub)
        PSID              : MT_00B0000001
```

EXAMPLE 3 Flash Upgrading an IB HCA Device

The following command flash upgrades an IB HCA device.

```
example# fwflash -f ./version.3.2.0000 -d 0
About to update firmware on:
  /devices/pci@1d,700000/pci@1/pci15b3,5a44@0:devctl
Continue (Y/N): Y

Updating . . . . . .
```

EXAMPLE 3 Flash Upgrading an IB HCA Device *(Continued)*

```
Done.  New image will be active after the system is rebooted.
```

Note that you are prompted before the upgrading proceeds.

The following command adds the `-y` option to the preceding command.

```
example# fwflash -y -f ./version.3.2.0000 -d 0
About to update firmware on:
  /devices/pci@1d,700000/pci@1/pci15b3,5a44@0:devctl

Updating . . . . . .
Done.  New image will be active after the system is rebooted.
```

EXAMPLE 4 Reading Device Firmware to File

The command shown below reads the device firmware to a file. The command uses the `-y` option so that read occurs without prompting.

```
example# fwflash -y -r /firmware.bin -d 1
About to read firmware on:
         /devices/pci@0,0/pci8086,3596@3/pci15b3,6278@0:devctl
to filename: /firmware.bin

Reading . . .
Done.
```

Return Values

The `fwflash` command returns the following values:

`0`
: Success

`1`
: Failure

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfwflash

See Also

`attributes(5)`, `tavor(7D)`

Notes

The `fwflash` command supports IB class HCA cards containing either the AMD or the Intel parallel flash parts. The `tavor(7D)` HCA driver is required.

Name fwtmp, wtmpfix – manipulate connect accounting records

Synopsis

```
/usr/lib/acct/fwtmp [-ic]

/usr/lib/acct/wtmpfix [file]...
```

Description `fwtmp` reads from the standard input and writes to the standard output, converting binary records of the type found in `/var/adm/wtmpx` to formatted ASCII records. The ASCII version is useful when it is necessary to edit bad records.

`wtmpfix` examines the standard input or named files in `utmpx` format, corrects the time/date stamps to make the entries consistent, and writes to the standard output. A hyphen (–) can be used in place of *file* to indicate the standard input. If time/date corrections are not performed, `acctcon(1M)` will fault when it encounters certain date-change records.

Each time the date is set, a pair of date change records are written to `/var/adm/wtmpx`. The first record is the old date denoted by the string "`old time`" placed in the `line` field and the flag `OLD_TIME` placed in the `type` field of the `utmpx` structure. The second record specifies the new date and is denoted by the string `new time` placed in the `line` field and the flag `NEW_TIME` placed in the `type` field. `wtmpfix` uses these records to synchronize all time stamps in the file.

In addition to correcting time/date stamps, `wtmpfix` will check the validity of the `name` field to ensure that it consists solely of alphanumeric characters or spaces. If it encounters a name that is considered invalid, it will change the login name to `INVALID` and write a diagnostic to the standard error. In this way, `wtmpfix` reduces the chance that `acctcon` will fail when processing connect accounting records.

Options `-ic` Denotes that input is in ASCII form, and output is to be written in binary form.

Files `/var/adm/wtmpx` history of user access and administration information

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

See Also `acctcom(1)`, `ed(1)`, `acct(1M)`, `acctcms(1M)`, `acctcon(1M)`, `acctmerg(1M)`, `acctprc(1M)`, `acctsh(1M)`, `runacct(1M)`, `acct(2)`, `acct.h(3HEAD)`, `utmpx(4)`, `attributes(5)`

System Administration Guide: Basic Administration

Name getdev – lists devices based on criteria

Synopsis

```
getdev [-ae] [criteria]... [device]...
```

Description `getdev` generates a list of devices that match certain criteria. The criteria includes a list of attributes (given in expressions) and a list of devices. If no criteria are given, all devices are included in the list.

Devices must satisfy at least one of the criteria in the list unless the `-a` option is used. Then, only those devices which match all of the criteria in a list will be included.

Devices which are defined on the command line and which match the criteria are included in the generated list. However, if the `-e` option is used, the list becomes a set of devices to be *excluded* from the list. See `OPTIONS` and `OPERANDS`.

Options The following options are supported:

`-a` Specifies that a device must match all criteria to be included in the list generated by this command. The option has no effect if no criteria are defined.

`-e` Specifies that the list of devices which follows on the command line should be *excluded* from the list generated by this command. Without the `-e` the named devices are *included* in the generated list. The flag has no effect if no devices are defined.

Operands The following operands are supported:

criteria Defines the criteria that a device must match to be included in the generated list. *criteria* is specified by expressions.

There are four possible expression types which the criteria specified in the *criteria* argument may follow:

attribute`=`*value* Selects all devices whose attribute *attribute* is defined and is equal to *value*.

attribute`!=`*value* Selects all devices whose attribute *attribute* is defined and does not equal *value*.

attribute`:*` Selects all devices which have the attribute *attribute* defined.

attribute`!:*` Selects all devices which do not have the attribute *attribute* defined.

See the `putdev(1M)` manual page for a complete listing and description of available attributes.

device Defines the devices which should be included in the generated list. This can be the pathname of the device or the device alias.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Command syntax was incorrect, invalid option was used, or an internal error occurred.

`2` Device table could not be opened for reading.

Files `/etc/device.tab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `devattr(1M)`, `getdgrp(1M)`, `putdev(1M)`, `putdgrp(1M)`, `attributes(5)`

Name getdevpolicy – inspect the system's device policy

Synopsis

```
/usr/sbin/getdevpolicy [device...]
```

Description Without arguments, `getdevpolicy` outputs the device policy in effect to standard output.

With arguments, each argument is treated as a pathname to a device and the device policy in effect for that specific device is printed preceeded by the supplied pathname.

Usage The device policy adds access restrictions over and above the file permissions.

Exit Status The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	See below

The invocation is evolving. The output is unstable.

See Also `add_drv(1M)`, `rem_drv(1M)`, `update_drv(1M)`, `attributes(5)`, `privileges(5)`, `devfs(7FS)`

Name getdgrp – lists device groups which contain devices that match criteria

Synopsis

```
/usr/sbin/getdgrp [-ael] [criteria]... [dgroup]...
```

Description `getdgrp` generates a list of device groups that contain devices matching the given criteria. The criteria is given in the form of expressions.

Options The following options are supported:

`-a` Specifies that a device must match all criteria to be included in the list generated by this command. The option has no effect if no criteria are defined.

`-e` Specifies that the list of device groups on the command line should be *excluded* from the list generated by this command. Without the `-e` option the named device groups are *included* in the generated list. The flag has no effect if no devices are defined.

`-l` Specifies that all device groups (subject to the `-e` option and the *dgroup* list) should be listed even if they contain no valid device members. This option has no affect if *criteria* is specified on the command line.

Operands The following operands are supported:

criteria Defines criteria that a device must match before a device group to which it belongs can be included in the generated list. Specify *criteria* as an expression or a list of expressions which a device must meet for its group to be included in the list generated by `getdgrp`. If no criteria are given, all device groups are included in the list.

Devices must satisfy at least one of the criteria in the list. However, the `-a` option can be used to define that a "logical and" operation should be performed. Then, only those groups containing devices which match all of the criteria in a list will be included.

There are four possible expressions types which the criteria specified in the *criteria* argument may follow:

attribute`=`*value* Selects all device groups with a member whose attribute *attribute* is defined and is equal to *value*.

attribute`!=`*value* Selects all device groups with a member whose attribute *attribute* is defined and does not equal *value*.

attribute`:*` Selects all device groups with a member which has the attribute *attribute* defined.

attribute`!:*` Selects all device groups with a member which does not have the attribute *attribute* defined.

See `putdev(1M)` for a complete listing and description of available attributes.

dgroup Defines a set of device groups which should be included in or excluded from the generated list. Device groups that are defined and which contain devices matching the criteria are included.

If the `-e` option is used, this list defines a set of device groups to be excluded. When the `-e` option is used and *criteria* is also defined, the generated list will include device groups containing devices which match the criteria and are not in the command line list.

Exit Status The following exit values are returned:

`0` Successful completion of the task.

`1` Command syntax was incorrect, invalid option was used, or an internal error occurred.

`2` Device table or device group table could not be opened for reading.

Files `/etc/device.tab`

`/etc/dgroup.tab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `devattr(1M)`, `getdev(1M)`, `putdev(1M)`, `putdgrp(1M)`, `attributes(5)`

Name getent – get entries from administrative database

Synopsis `getent` *database* [*key*]...

Description `getent` gets a list of entries from the administrative database specified by *database*. The information generally comes from one or more of the sources that are specified for the *database* in `/etc/nsswitch.conf`.

database is the name of the database to be examined. This can be `passwd`, `group`, `hosts`, `ipnodes`, `services`, `protocols`, `ethers`, `project`, `networks`, or `netmasks`. For each of these databases, `getent` uses the appropriate library routines described in `getpwnam`(3C), `getgrnam`(3C), `gethostbyaddr`(3NSL), `gethostbyname`(3NSL), `getipnodebyaddr`(3SOCKET), `getipnodebyname`(3SOCKET), `getservbyname`(3SOCKET), `getprotobyname`(3SOCKET), `ethers`(3SOCKET), `getprojbyname`(3PROJECT) and `getnetbyname`(3SOCKET), respectively.

Each *key* must be in a format appropriate for searching on the respective database. For example, it can be a *username* or *numeric-uid* for `passwd`; *hostname* or *IP address* for `hosts`; or *service*, *service/protocol*, *port*, or *port/proto* for `services`.

`getent` prints out the database entries that match each of the supplied keys, one per line, in the format of the matching administrative file: `passwd`(4), `group`(4), `project`(4), `hosts`(4), `services`(4), `protocols`(4), `ethers`(3SOCKET), `networks`(4), or `netmasks`(4). If no key is given, all entries returned by the corresponding enumeration library routine, for example, `getpwent()` or `gethostent()`, are printed. Enumeration is not supported on `ipnodes`.

Key Interpretation for `passwd` and `group` Databases

When `getent` is invoked with database set to `passwd`, each key value is processed as follows:

- If the key value consists only of numeric characters, `getent` assumes that the key value is a numeric user ID and searches the user database for a matching user ID.
- If the user ID is not found in the user database or if the key value contains any non-numeric characters, `getent` assumes the key value is a user name and searches the user database for a matching user name.

Similarly, when `getent` is invoked with database set to `group`, each key value is processed as follows:

- If the key value consists only of numeric characters, `getent` assumes that the key value is a numeric group ID and searches the group database for a matching group ID.
- If the group ID is not found in the `group` database or if the key value contains any non-numeric characters, `getent` assumes the key value is a group name and searches the `group` database for a matching group name.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Command syntax was incorrect, an invalid option was used, or an internal error occurred.

2 At least one of the specified entry names was not found in the database.

3 There is no support for enumeration on this database.

Files

`/etc/nsswitch.conf`	name service switch configuration file
`/etc/passwd`	password file
`/etc/group`	group file
`/etc/inet/hosts`	IPv4 and IPv6 host name database
`/etc/services`	Internet services and aliases
`/etc/project`	project file
`/etc/protocols`	protocol name database
`/etc/ethers`	Ethernet address to hostname database or domain
`/etc/networks`	network name database
`/etc/netmasks`	network mask database

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `ethers(3SOCKET)`, `getgrnam(3C)`, `gethostbyaddr(3NSL)`, `gethostbyname(3NSL)`, `gethostent(3NSL)`, `getipnodebyaddr(3SOCKET)`, `getipnodebyname(3SOCKET)`, `getnetbyname(3SOCKET)`, `getprojbyname(3PROJECT)`, `getprotobyname(3SOCKET)`, `getpwnam(3C)`, `getservbyname(3SOCKET)`, `group(4)`, `hosts(4)`, `netmasks(4)`, `networks(4)`, `nsswitch.conf(4)`, `passwd(4)`, `project(4)`, `protocols(4)`, `services(4)`, `attributes(5)`

Name gettable – get DoD Internet format host table from a host

Synopsis `/usr/sbin/gettable` *host*

Description `gettable` is a simple program used to obtain the DoD Internet host table from a "hostname" server. The specified *host* is queried for the table. The table is placed in the file `hosts.txt`.

`gettable` operates by opening a TCP connection to the port indicated in the service specification for "hostname". A request is then made for all names and the resultant information is placed in the output file.

`gettable` is best used in conjunction with the `htable(1M)` program which converts the DoD Internet host table format to that used by the network library lookup routines.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `htable(1M)`, `attributes(5)` Harrenstien, Ken, Mary Stahl, and Elizabeth Feinler, HOSTNAME *Server*, RFC 953, Network Information Center, SRI International, Menlo Park, California, October 1985.

Notes Should allow requests for only part of the database.

Name getty – set terminal type, modes, speed, and line discipline

Synopsis

```
/usr/lib/saf/ttymon [-h] [-t timeout] line
     [speed [type [linedisc]]]

/usr/lib/saf/ttymon -c file
```

Description `getty` sets terminal type, modes, speed, and line discipline. `getty` is a symbolic link to `/usr/lib/saf/ttymon`. It is included for compatibility with previous releases for the few applications that still call `getty` directly.

`getty` can only be executed by the super-user, (a process with the user ID `root`). Initially `getty` prints the login prompt, waits for the user's login name, and then invokes the `login` command. `getty` attempts to adapt the system to the terminal speed by using the options and arguments specified on the command line.

Without optional arguments, `getty` specifies the following: The *speed* of the interface is set to 300 baud, either parity is allowed, NEWLINE characters are converted to carriage return-line feed, and tab expansion is performed on the standard output. `getty` types the login prompt before reading the user's name a character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user pressing the BREAK key. This will cause `getty` to attempt the next *speed* in the series. The series that `getty` tries is determined by what it finds in `/etc/ttydefs` .

Options The following options are supported:

`-h` If the `-h` flag is not set, a hangup will be forced by setting the speed to zero before setting the speed to the default or a specified speed.

`-t` *timeout* Specifies that `getty` should exit if the open on the line succeeds and no one types anything in *timeout* seconds.

`-c` *file* The `-c` option is no longer supported. Instead use `/usr/sbin/sttydefs -l` to list the contents of the `/etc/ttydefs` file and perform a validity check on the file.

Operands The following operands are supported:

line The name of a TTY line in `/dev` to which `getty` is to attach itself. `getty` uses this string as the name of a file in the `/dev` directory to open for reading and writing.

speed The *speed* argument is a label to a speed and TTY definition in the file `/etc/ttydefs`. This definition tells `getty` at what speed to run initially, what the initial TTY settings are, and what speed to try next, (should the user press the BREAK key to indicate that the speed is inappropriate). The default *speed* is 300 baud.

type and *linedisc* These options are obsolete and will be ignored.

Files `/etc/ttydefs`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

See Also `ct(1C)`, `login(1)`, `sttydefs(1M)`, `ttymon(1M)`, `ioctl(2)`, `attributes(5)`, `tty(7D)`

Name getvol – verifies device accessibility

Synopsis

```
/usr/bin/getvol -n [-l label] device

/usr/bin/getvol [-f | -F] [-ow] [-l label | -x label] device
```

Description `getvol` verifies that the specified device is accessible and that a volume of the appropriate medium has been inserted. The command is interactive and displays instructional prompts, describes errors, and shows required label information.

Options The following options are supported:

`-n` Runs the command in non-interactive mode. The volume is assumed to be inserted upon command invocation.

`-l` *label* Specifies that the label *label* must exist on the inserted volume (can be overridden by the `-o` option).

`-f` Formats the volume after insertion, using the format command defined for this device in the device table.

`-F` Formats the volume after insertion and places a file system on the device. Also uses the format command defined for this device in the device table.

`-o` Allows the administrator to override a label check.

`-w` Allows administrator to write a new label on the device. User is prompted to supply the label text. This option is ineffective if the `-n` option is enabled.

`-x` *label* Specifies that the label *label* must exist on the device. This option should be used in place of the `-l` option when the label can only be verified by visual means. Use of the option causes a message to be displayed asking the administrator to visually verify that the label is indeed *label*.

Operands The following operands are supported:

device Specifies the device to be verified for accessibility.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Command syntax was incorrect, invalid option was used, or an internal error occurred.

`3` Device table could not be opened for reading.

Files `/etc/device.tab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also attributes(5)

Notes This command uses the device table to determine the characteristics of the device when performing the volume label checking.

Name gkadmin – Kerberos principals and policies administration GUI

Synopsis `/usr/sbin/gkadmin`

Description `gkadmin` is an interactive graphical user interface (GUI) that enables you to maintain Kerberos principals and policies. `gkadmin` provides much the same functionality as the `kadmin(1M)` command.

`gkadmin` does not support the management of keytabs. You must use `kadmin` for keytabs management. `gkadmin` uses Kerberos authentication and an encrypted RPC to operate securely from anywhere on the network.

When `gkadmin` is invoked, the login window is populated with default values. For the principal name, `gkadmin` determines your user name from the `USER` environment variable. It appends `/admin` to the name (*username*`/admin`) to create a default user instance in the same manner as `kadmin`. It also selects appropriate defaults for realm and master KDC (*admin_server*) from the `/etc/krb5/krb5.conf` file.

You can change these defaults on the login window. When you enter your password, a session is started with `kadmind`. Operations performed are subject to permissions that are granted or denied to the chosen user instance by the Kerberos ACL file. See `kadm5.acl(4)`.

After the session is started, a tabbed folder is displayed that contains a principal list and a policy list. The functionality is mainly the same as `kadmin`, with addition, deletion, and modification of principal and policy data available.

`gkadmin` also includes an interface to specify principal key encryption types when modifying or creating principal records. The default set of encryption types is used if they are not selected through this interface. The default set of encryption types can be found in `krb5.conf(4)` under the `default_tkt_enctypes` section.

In addition, `gkadmin` provides the following features:

- New principal or policy records can be added either from default values or from the settings of an existing principal.
- A comment field is available for principals.
- Default values are saved in `$HOME/.gkadmin`.
- A logout option permits you to log back in as another user instance without exiting the tool.
- Principal and policy lists and attributes can be printed or saved to a file.
- Online context-sensitive help and general help is available in the `Help` menu.

Files

`/etc/krb5/krb5.conf`	Kerberos configuration information on a Kerberos client. Used to search for default realm and master KDC (`admin_server`), including a port number for the master KDC.

`$HOME/.gkadmin`	Default parameters used to initialize new principals created during the session.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkdcu
Interface Stability	Evolving

See Also `kpasswd(1)`, `kadmin(1M)`, `kadmind(1M)`, `kadmin.local(1M)`, `kdb5_util(1M)`, `kadm5.acl(4)`, `kdc.conf(4)`, `krb5.conf(4)`, `attributes(5)`, `kerberos(5)`

Diagnostics The `gkadmin` interface is currently incompatible with the MIT `kadmind` daemon interface, so you cannot use this interface to administer an MIT-based Kerberos database. However, clients running the Solaris implementation of Kerberos can still use an MIT-based KDC.

Name groupadd – add (create) a new group definition on the system

Synopsis

```
/usr/sbin/groupadd [-g gid [-o]] group
```

Description The `groupadd` command creates a new group definition on the system by adding the appropriate entry to the `/etc/group` file.

Options The following options are supported:

`-g` *gid* Assigns the group id *gid* for the new group. This group id must be a non-negative decimal integer below `MAXUID` as defined in `/usr/include/sys/param.h`. The group ID defaults to the next available (unique) number above the highest number currently assigned. For example, if groups `100`, `105`, and `200` are assigned as groups, the next default group number is `201`. (Group IDs from `0`–`99` are reserved by SunOS for future applications.)

`-o` Allows the *gid* to be duplicated (non-unique).

Operands The following operands are supported:

group A string consisting of characters from the set of lower case alphabetic characters and numeric characters. A warning message is written if the string exceeds `MAXGLEN-1`, which is usually eight characters. The *group* field must contain at least one character; it accepts lower case or numeric characters or a combination of both, and must not contain a colon (`:`) or NEWLINE.

Exit Status The following exit values are returned:

`0` Successful completion.

`2` Invalid command syntax. A usage message for the `groupadd` command is displayed.

`3` An invalid argument was provided to an option.

`4` The *gid* is not unique (when `-o` option is not used).

`9` The *group* is not unique.

`10` The `/etc/group` file cannot be updated.

Files

```
/etc/group

/usr/include/userdefs.h
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also users(1B), groupdel(1M), groupmod(1M), grpck(1M), logins(1M), pwck(1M), useradd(1M), userdel(1M), usermod(1M), group(4), attributes(5)

Notes groupadd only adds a group definition to the local system. If a network name service such as NIS or NIS+ is being used to supplement the local /etc/group file with additional entries, groupadd cannot change information supplied by the network name service. However, groupadd verifies the uniqueness of group name and group ID against the external name service.

Name groupdel – delete a group definition from the system

Synopsis

```
/usr/sbin/groupdel group
```

Description The `groupdel` utility deletes a group definition from the system. It deletes the appropriate entry from the `/etc/group` file.

Operands *group* An existing group name to be deleted.

Exit Status The following exit values are returned:

`0` Success.

`2` Invalid command syntax. A usage message for the `groupdel` command is displayed.

`6` *group* does not exist.

`10` Cannot update the `/etc/group` file.

Files `/etc/group` system file containing group definitions

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `users(1B)`, `groupadd(1M)`, `groupmod(1M)`, `logins(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `attributes(5)`

Notes The `groupdel` utility only deletes a group definition that is in the local `/etc/group` file. If a network nameservice such as NIS or NIS+ is being used to supplement the local `/etc/group` file with additional entries, `groupdel` cannot change information supplied by the network nameservice.

Name groupmod – modify a group definition on the system

Synopsis `/usr/sbin/groupmod [-g` *gid* `[-o]] [-n` *name*`]` *group*

Description The `groupmod` command modifies the definition of the specified group by modifying the appropriate entry in the `/etc/group` file.

Options The following options are supported:

`-g` *gid* Specify the new group ID for the group. This group ID must be a non-negative decimal integer less than `MAXUID`, as defined in `<param.h>`. The group ID defaults to the next available (unique) number above 99. (Group IDs from 0-99 are reserved by SunOS for future applications.)

`-n` *name* Specify the new name for the group. The *name* argument is a string of no more than eight bytes consisting of characters from the set of lower case alphabetic characters and numeric characters. A warning message will be written if these restrictions are not met. A future Solaris release may refuse to accept group fields that do not meet these requirements. The *name* argument must contain at least one character and must not include a colon (`:`) or NEWLINE (`\n`).

`-o` Allow the *gid* to be duplicated (non-unique).

Operands The following operands are supported:

group An existing group name to be modified.

Exit Status The `groupmod` utility exits with one of the following values:

`0` Success.

`2` Invalid command syntax. A usage message for the `groupmod` command is displayed.

`3` An invalid argument was provided to an option.

`4` *gid* is not unique (when the `-o` option is not used).

`6` *group* does not exist.

`9` *name* already exists as a group name.

`10` Cannot update the `/etc/group` file.

Files `/etc/group` group file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also users(1B), groupadd(1M), groupdel(1M), logins(1M), useradd(1M), userdel(1M), usermod(1M), group(4), attributes(5)

Notes The groupmod utility only modifies group definitions in the /etc/group file. If a network name service such as NIS or NIS+ is being used to supplement the local /etc/group file with additional entries, groupmod cannot change information supplied by the network name service. The groupmod utility will, however, verify the uniqueness of group name and group ID against the external name service.

Name growfs – non-destructively expand a UFS file system

Synopsis

```
/usr/sbin/growfs [-M mount-point] [newfs-options]
     [raw-device]
```

Description `growfs` non-destructively expands a mounted or unmounted UNIX file system (UFS) to the size of the file system's slice(s).

Typically, disk space is expanded by first adding a slice to a metadevice, then running the `growfs` command. When adding space to a mirror, you expand each submirror before expanding the file system.

`growfs` will "write-lock" (see `lockfs(1M)`) a mounted file system when expanding. The length of time the file system is write-locked can be shortened by expanding the file system in stages. For instance, to expand a 1 Gbyte file system to 2 Gbytes, the file system can be grown in 16 Mbyte stages using the `-s` option to specify the total size of the new file system at each stage. The argument for `-s` is the number of sectors, and must be a multiple of the cylinder size. Note: The file system cannot be grown if a cylinder size of less than 2 is specified. Refer to the `newfs(1M)` man page for information on the options available when growing a file system.

`growfs` displays the same information as `mkfs` during the expansion of the file system.

If `growfs` is aborted, recover any lost free space by unmounting the file system and running the `fsck` command, or run the `growfs` command again.

Note: If `growfs` is aborted and the file system is used before `fsck` is run on it, UFS metadata might be left in an incomplete state, with the result that the file system would be corrupted. In such a circumstance, you would have to restore the file system from backups.

Options Root privileges are required for all of the following options.

`-M` *mount-point*	The file system to be expanded is mounted on *mount-point*. File system locking (`lockfs`) will be used.
newfs-options	The options are documented in the `newfs` man page.
raw-device	Specifies the name of a raw metadevice or raw special device, residing in `/dev/md/rdsk`, or `/dev/rdsk`, respectively, including the disk slice, where you want the file system to be grown.

Examples **EXAMPLE 1** Expanding nonmetadevice slice for `/export` file system

The following example expands a nonmetadevice slice for the `/export` file system. In this example, the existing slice, `/dev/dsk/c1t0d0s3`, is converted to a metadevice so additional slices can be concatenated.

```
# metainit -f d8 2 1 c1t0d0s3 1 c2t0d0s3
# umount /export
```

EXAMPLE 2 Associate `/export` with new metadevice

Edit the `/etc/vfstab` file to change the entry for `/export` to the newly defined metadevice, `d8`.

```
# mount /export
# growfs -M /export /dev/md/rdsk/d8
```

The first example starts by running the `metainit` command with the `-f` option to force the creation of a new concatenated metadevice `d8`, which consists of the existing slice `/dev/dsk/c1t0d0s3` and a new slice `/dev/dsk/c2t0d0s3`. Next, the file system on `/export` must be unmounted. The `/etc/vfstab` file is edited to change the entry for `/export` to the newly defined metadevice name, rather than the slice name. After the file system is remounted, the `growfs` command is run to expand the file system. The file system will span the entire metadevice when `growfs` completes. The `-M` option enables the `growfs` command to expand a mounted file system. During the expansion, write access for `/export` is suspended until `growfs` unlocks the file system. Read access is not affected, though access times are not kept when the lock is in effect.

EXAMPLE 3 Dynamic Expansion of `/export` file system

The following example picks up from the previous one. Here, the `/export` file system mounted on metadevice `d8` is dynamically expanded.

```
# metattach d8 c0t1d0s2
# growfs -M /export /dev/md/rdsk/d8
```

This example begins by using the `metattach` command to dynamically concatenate a new slice, `/dev/dsk/c0t1d0s2`, to the end of an existing metadevice, `d8`. Next, the `growfs` command specifies that the mount-point is `/export` and that it is to be expanded onto the raw metadevice `/dev/md/rdsk/d8`. The file system will span the entire metadevice when `growfs` completes. During the expansion, write access for `/export` is suspended until `growfs` unlocks the file system. Read access is not affected, though access times are not kept when the lock is in effect.

EXAMPLE 4 Expanding mounted file system to existing mirror

The following example expands a mounted file system `/files`, to an existing mirror, `d80`, which contains two submirrors, `d9` and `d10`.

```
# metattach d9 c0t2d0s5
# metattach d10 c0t3d0s5
# growfs -M /files /dev/md/rdsk/d80
```

In this example, the `metattach` command dynamically concatenates the new slices to each submirror. The `metattach` command must be run for each submirror. The mirror will automatically grow when the last submirror is dynamically concatenated. The mirror will grow to the size of the smallest submirror. The `growfs` command then expands the file system. The `growfs` command specifies that the mount-point is `/files` and that it is to be expanded

EXAMPLE 4 Expanding mounted file system to existing mirror *(Continued)*

onto the raw metadevice `/dev/md/rdsk/d80`. The file system will span the entire mirror when the `growfs` command completes. During the expansion, write access for the file system is suspended until `growfs` unlocks the file system. Read access is not affected, though access times are not kept when the lock is in effect.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also `fsck(1M)`, `lockfs(1M)`, `mkfs(1M)`, `metattach(1M)`, `newfs(1M)`, `attributes(5)`

Solaris Volume Manager Administration Guide

Limitations Only UFS file systems (either mounted or unmounted) can be expanded using the `growfs` command. Once a file system is expanded, it cannot be decreased in size. The following conditions prevent you from expanding file systems: When `acct` is activated and the accounting file is on the target device. When C2 security is activated and the logging file is on the target file system. When there is a local `swap` file in the target file system. When the file system is root (`/`), `/usr`, or `swap`.

Name gsscred – add, remove, and list gsscred table entries

Synopsis

```
gsscred [-n user [-o oid] [-u uid]] [-c comment] -m mech -a

gsscred [-n user [-o oid]] [-u uid] [-m mech] -r

gsscred [-n user [-o oid]] [-u uid] [-m mech] -l
```

Description The `gsscred` utility is used to create and maintain a mapping between a security principal name and a local UNIX *uid*. The format of the user name is assumed to be `GSS_C_NT_USER_NAME`. You can use the `-o` option to specify the object identifier of the *name* type. The `OID` must be specified in dot-separated notation, for example: `1.2.3.45464.3.1`

The `gsscred` table is used on server machines to lookup the *uid* of incoming clients connected using `RPCSEC_GSS`.

When adding users, if no *user* name is specified, an entry is created in the table for each user from the `passwd` table. If no *comment* is specified, the `gsscred` utility inserts a comment that specifies the user name as an ASCII string and the GSS-APIsecurity mechanism that applies to it. The security mechanism will be in string representation as defined in the `/etc/gss/mech` file.

The parameters are interpreted the same way by the `gsscred` utility to delete users as they are to create users. At least one of the following options must be specified: `-n`, `-u`, or `-m`. If no security mechanism is specified, then all entries will be deleted for the user identified by either the *uid* or *user* name. If only the security mechanism is specified, then all *user* entries for that security mechanism will be deleted.

Again, the parameters are interpreted the same way by the `gsscred` utility to search for users as they are to create users. If no options are specified, then the entire table is returned. If the *user* name or *uid* is specified, then all entries for that *user* are returned. If a security mechanism is specified, then all *user* entries for that security mechanism are returned.

Options

`-a`	Add a table entry.
`-c` *comment*	Insert comment about this table entry.
`-l`	Search table for entry.
`-m` *mech*	Specify the mechanism for which this name is to be translated.
`-n` *user*	Specify the optional principal name.
`-o` *oid*	Specify the OID indicating the name type of the user.
`-r`	Remove the entry from the table.
`-u` *uid*	Specify the *uid* for the *user* if the *user* is not local.

Examples

EXAMPLE 1 Creating a `gsscred` Table for the Kerberos v5 Security Mechanism

The following shows how to create a `gsscred` table for the kerberos v5 security mechanism. `gsscred` obtains *user* names and *uid*'s from the `passwd` table to populate the table.

```
example% gsscred -m kerberos_v5 -a
```

EXAMPLE 2 Adding an Entry for `root/host1` for the Kerberos v5 Security Mechanism

The following shows how to add an entry for `root/host1` with a specified *uid* of `0` for the kerberos v5 security mechanism.

```
example% gsscred -m kerberos_v5 -n root/host1 -u 0 -a
```

EXAMPLE 3 Listing All User Mappings for the Kerberos v5 Security Mechanism

The following lists all user mappings for the kerberos v5 security mechanism.

```
example% gsscred -m kerberos_v5 -l
```

EXAMPLE 4 Listing All Mappings for All Security Mechanism for a Specified User

The following lists all mappings for all security mechanisms for the user `bsimpson`.

```
example% gsscred -n bsimpson -l
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWgss
Interface Stability	Evolving

See Also

`gssd(1M)`, `gsscred.conf(4)`, `attributes(5)`

Notes

Some GSS mechanisms, such as `kerberos_v5`, provide their own authenticated-name-to-local-name (uid) mapping and thus do not usually have to be mapped using `gsscred`. See `gsscred.conf(4)` for more information.

Name gssd – generates and validates GSS-API tokens for kernel RPC

Synopsis

```
/usr/lib/gss/gssd
```

Description `gssd` is the user mode daemon that operates between the kernel rpc and the Generic Security Service Application Program Interface (GSS-API) to generate and validate GSS-API security tokens. In addition, `gssd` maps the GSS-API principal names to the local user and group ids. By default, all groups that the requested user belongs to will be included in the grouplist credential. `gssd` is invoked by the Internet daemon `inetd(1M)` the first time that the kernel RPC requests GSS-API services.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWgssk
Interface Stability	Evolving

See Also `kill(1)`, `pkill(1)`, `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `gsscred(1M)`, `svcadm(1M)`, `gsscred.conf(4)`, `resolv.conf(4)`, `attributes(5)`, `smf(5)`

RFC 2078

Notes The following signal has the specified effect when sent to the server process using the `kill(1)` command:

`SIGHUP` `gssd` rereads the `gsscred.conf(4)` options.

When one of the mechanisms being used is Kerberos, then the `gssd` process must be restarted after adding or changing the `resolv.conf(4)` file.

The `gssd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/gss:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name halt, poweroff – stop the processor

Synopsis

```
/usr/sbin/halt [-dlnqy]

/usr/sbin/poweroff [-dlnqy]
```

Description The `halt` and `poweroff` utilities write any pending information to the disks and then stop the processor. The `poweroff` utility has the machine remove power, if possible.

The `halt` and `poweroff` utilities normally log the system shutdown to the system log daemon, `syslogd(1M)`, and place a shutdown record in the login accounting file `/var/adm/wtmpx`. These actions are inhibited if the `-n` or `-q` options are present.

Options The following options are supported:

`-d` Force a system crash dump before rebooting. See `dumpadm(1M)` for information on configuring system crash dumps.

`-l` Suppress sending a message to the system log daemon, `syslogd(1M)`, about who executed `halt`.

`-n` Prevent the `sync(1M)` before stopping.

`-q` Quick halt. No graceful shutdown is attempted.

`-y` Halt the system, even from a dialup terminal.

Files `/var/adm/wtmpx` History of user access and administration information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `dumpadm(1M)`, `init(1M)`, `reboot(1M)`, `shutdown(1M)`, `sync(1M)`, `syslogd(1M)`, `inittab(4)`, `attributes(5)`, `smf(5)`

Notes The `halt` and `poweroff` utilities do not cleanly shutdown `smf(5)` services. Execute the scripts in `/etc/rcnum.d` or execute shutdown actions in `inittab(4)`. To ensure a complete shutdown of system services, use `shutdown(1M)` or `init(1M)` to reboot a Solaris system.

Name hextoalabel – convert an internal text label to its human readable equivalent

Synopsis

```
/usr/sbin/hextoalabel [internal-text-sensitivity-label]
/usr/sbin/hextoalabel -c [internal-text-clearance]
```

Description `hextoalabel` converts an internal text label into its human readable equivalent and writes the result to the standard output file. This internal form is often hexadecimal. If no option is supplied, the label is assumed to be a sensitivity label.

If no internal text label is specified, the label is read from the standard input file. The expected use of this command is emergency repair of labels that are stored in internal databases.

Options

`-c` Identifies the internal text label as a clearance.

Exit Status The following exit values are returned:

`0` On success.

`1` On failure, and writes diagnostics to the standard error file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	See below.

The command output is Committed for systems with the same `label_encodings` file. The command invocation is Committed for systems that implement the DIA MAC policy.

Files

`/etc/security/tsol/label_encodings`
The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system.

See Also `atohexlabel(1M)`, `label_to_str(3TSOL)`, `str_to_label(3TSOL)`, `label_encodings(4)`, `attributes(5)`

"How to Obtain a Readable Label From Its Hexadecimal Form" in *Oracle Solaris Trusted Extensions Administrator's Procedures*

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

This file is part of the Defense Intelligence Agency (DIA) Mandatory Access Control (MAC) policy. This file might not be applicable to other MAC policies that might be developed for future releases of Solaris Trusted Extensions software.

Name host – DNS lookup utility

Synopsis

```
host [-aCdilmrsTvw] [-c class] [-N ndots] [-R number]
     [-t type] [-W wait]  [-4 | -6] name [server]
```

Description The `host` utility performs simple DNS lookups. It is normally used to convert names to IP addresses and IP addresses to names. When no arguments or options are given, `host` prints a short summary of its command line arguments and options.

The *name* argument is the domain name that is to be looked up. It can also be a dotted-decimal IPv4 address or a colon-delimited IPv6 address, in which case `host` by default performs a reverse lookup for that address. The optional *server* argument is either the name or IP address of the name server that `host` should query instead of the server or servers listed in `/etc/resolv.conf`.

Options The following options are supported:

`-4`
Use only IPv4 transport. By default, both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-6`
Use only IPv6 transport. By default, both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-a`
Equivalent to setting the `-v` option and asking `host` to make a query of type `ANY`.

`-c` *class*
Make a DNS query of class *class*. This can be used to lookup Hesiod or Chaosnet class resource records. The default class is IN (Internet).

`-C`
Attempt to display the SOA records for zone *name* from all the listed authoritative name servers for that zone. The list of name servers is defined by the NS records that are found for the zone.

`-d`
Generate verbose output. This option is equivalent to `-v`. These two options are provided for backward compatibility. In previous versions, the `-d` option switched on debugging traces and `-v` enabled verbose output.

`-i`
Specifies that reverse lookups of IPv6 addresses should use the IP6.INT domain as defined in RFC 1886. The default is to use RFC 3152 domain IP6.ARPA.

`-l`
List mode. This option makes `host` perform a zone transfer for zone *name*, displaying the NS, PTR and address records (A/AAAA). If combined with `-a`, all records will be displayed. The argument is provided for compatibility with previous implementations. Options `-la` is equivalent to making a query of type `AXFR`.

`-m`
Sets the memory usage debugging flags: record, usage, and trace.

`-N` *ndots*
Set the number of dots that have to be in *name* for it to be considered absolute. The default value is that defined using the ndots statement in `/etc/resolv.conf`, or 1 if no ndots statement is present. Names with fewer dots are interpreted as relative names and will be searched for in the domains listed in the `search` or `domain` directive in `/etc/resolv.conf`.

`-r`
Make a non-recursive query. Setting this option clears the `RD` (recursion desired) bit in the query made by `host`. The name server receiving the query does not attempt to resolve *name*. The `-r` option enables `host` to mimic the behaviour of a name server by making non-recursive queries and expecting to receive answers to those queries that are usually referrals to other name servers.

`-R` *number*
Change the number of UDP retries for a lookup. The *number* argument indicates how many times `host` will repeat a query that does not get answered. The default number of retries is 1. If *number* is negative or zero, the number of retries will default to 1.

`-s`
Specifies that the host not send the query to the next name server if any server responds with a `SERVFAIL` response, which is the reverse of normal stub resolver behavior.

`-t` *type*
Select the query type. The *type* argument can be any recognised query type: `CNAME`, `NS`, `SOA`, `SIG`, `KEY`, and `AXFR`, among others. When no query type is specified, `host` automatically selects an appropriate query type. By default it looks for A, AAAA, and MX records, but if the `-C` option is specified, queries are made for SOA records. If *name* is a dotted-decimal IPv4 address or colon-delimited IPv6 address, `host` queries for PTR records.

If a query type of IXFR is chosen the starting serial number can be specified by appending an equal followed by the starting serial number (for example: `-t IXFR=12345678`).

`-T`
Use a TCP connection when querying the name server. TCP is automatically selected for queries that require it, such as zone transfer (`AXFR`) requests. By default `host` uses UDP when making queries.

`-v`
Generate verbose output. This option is equivalent to `-d`.

`-w`
Wait forever for a reply. The time to wait for a response will be set to the number of seconds given by the hardware's maximum value for an integer quantity.

`-W` *wait*
Wait for *wait* seconds for a reply. If *wait* is less than one, the wait interval is set to one second.

Files `/etc/resolv.conf`
Resolver configuration file

Attributes See for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dns/bind
Interface Stability	Volatile

See Also `dig(1M)`, `named(1M)`, `attributes(5)`

RFC 1035, *RFC 1886*, *RFC 3152*

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name hostconfig – configure a system's host parameters

Synopsis

```
/usr/sbin/hostconfig -p protocol [-d] [ -h] [-n] [-v]
     [-i interface] [-f hostname]
```

Description The `hostconfig` program uses a network protocol to acquire a machine's *host parameters* and set these parameters on the system.

The program selects which protocol to use based on the argument to the required `-p` flag. Different protocols may set different host parameters. Currently, only one protocol (`bootparams`) is defined.

Options The following options are supported:

`-d` Enable `debug` output.

`-f` *hostname* Run the protocol as if this machine were named `hostname`.

`-h` Echo the received `hostname` to `stdout`, rather than setting `hostname` using the system name directly.

`-i` *interface* Use only the named network interface to run the protocol.

`-n` Run the network protocol, but do not set the acquired parameters into the system.

`-p` *protocol* Run `hostconfig` using *protocol*. Currently, only one protocol (`bootparams`) is available. This option is required.

Specifying the `-p bootparams` option uses the `whoami` call of the RPC `bootparams` protocol. This sets the system's `hostname`, `domainname`, and default IP router parameters.

`-v` Enable verbose output.

Examples **EXAMPLE 1** Configuring Host Parameters with Verbose Output

The following command configures a machine's host parameters using the `whoami` call of the RPC `bootparams` protocol with a verbose output.

```
example# hostconfig -p bootparams -v
```

EXAMPLE 2 Displaying Host Parameters

The following command displays the parameters that would be set using the `whoami` call of the RPC `bootparams` protocol.

```
example# hostconfig -p bootparams -n -v
```

EXAMPLE 3 Configuring Host Parameters Less the System Name

The following command configures a machine's host parameters, less the system name, using the `whoami` call of the RPC `bootparams` protocol.

```
example# hostconfig='hostconfig -p bootparams -h'
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`hostname(1)`, `domainname(1M)`, `route(1M)`, `attributes(5)`

Name hotplug – configure hotplug connectors and ports

Synopsis

```
hotplug list [-l] [-v] [path [connection]]

hotplug online path port

hotplug offline [-f] [-q] path port

hotplug enable path connector

hotplug disable [-f] [-q] path connector

hotplug poweron path connector

hotplug poweroff [-f] [-q] path connector

hotplug set -o options path connector

hotplug get -o options path connector

hotplug -?
```

Description The `hotplug` command is used to manage hotplug connections. A connection can be a connector or port. A hotplug connector is a representation of a physical point in the system where components can be inserted or removed. A hotplug port is a representation of a logical point in the system device tree where the connection of a device to the system is managed.

The `hotplug` command only supports hotplug operations on hotplug connectors for PCI Express buses and PCI buses that implement the Standard PCI Hotplug feature. Hotplug ports on PCI Expresss and PCI buses in systems with PCI Express fabrics are also supported. Additional buses may be supported in the future.

The `hotplug` command operates on the following kinds of objects:

`path`
: Hotplug connectors and ports are integrated into the Solaris device tree. The names of connectors and ports are unique relative only to their bus controller. A device path is required to uniquely reference a connector or port.

`connector`
: If a hardware component supports being physically inserted or removed, then a hotplug connector represents the location where this action may occur. When a connector exists, it has a hierarchy of ports and device nodes that depend upon it.

`port`
: All device nodes can be virtually hotplugged, even if their hardware does not support physical hotplugging. A hotplug port exists between a device node and its parent node in the system device tree. It represents the location where the device node and its dependents can be managed.

`connection`
: A hotplug connection is a generic term to refer to either a hotplug connector or a hotplug port.

Hotplug connectors and ports are managed according to a state model. The `hotplug` command can list information about the hotplug connections in a system, or it can initiate change of state operations on specific hotplug connections.

Hotplug connectors can be in the following states:

`empty`
A component is not physically inserted in the connector.

`present`
A component is physically inserted in the connector, but the component is powered off. The component is not in use.

`powered`
A component is physically inserted in the connector, and the component is powered on. The component is disabled and is not in use.

`enabled`
A component is physically inserted in the connector. The component is powered on and has been probed and tested. The component is enabled and devices that represent its functions can be used.

Hotplug ports can be in the following states:

`port-empty`
No device exists for the hotplug port.

`port-present`
A device exists for the hotplug port, but the device has not been probed and it has no attached device driver. The device is not in use.

`offline`
A device exists for the hotplug port, and the device has been probed. A device driver is not attached, and the device is not in use.

`online`
A device exists for the hotplug port, and its device driver is fully attached. The device is in use.

`maintenance`
A device exists for the hotplug port, and its device driver is fully attached. The device is in use, but not fully operational. A maintenance or fault management operation is affecting the device.

The `hotplug` command can also access bus private properties for each hotplug connector. The current values of bus private properties can be displayed. New values for each bus private property can be set directly.

Sub-commands The following subcommands are supported:

`list`
Show information for hotplug connectors, ports, and their associated devices. Hotplug connectors and hotplug ports are integrated into the Solaris device tree hierarchy. The `list` subcommand therefore displays the hierarchy of device nodes with additional information included to show the locations of hotplug connectors and hotplug ports. The names of hotplug connectors are enclosed in square brackets, and the names of hotplug ports are enclosed in angled brackets. The current state of each hotplug connection is displayed next to its name.

`online`
Change the state of a hotplug port to the `online` state.

`offline`
Change the state of a hotplug port to the `offline` state.

`enable`
Change the state of a hotplug connector to the `enabled` state. All of the hotplug connector's dependent ports will be automatically probed and initialized into the `online` state.

`disable`
Change the state of a hotplug connector from the `enabled` state to the `powered` state. All dependent ports that are in the `online` state will first be transitioned to the `port-present` state.

`poweron`
Change the state of a hotplug connector from the `present` state to the `powered` state.

`poweroff`
Change the state of a hotplug connector from the `powered` or `enabled` state to the `present` state. All dependent ports that are in the `online` state will first be transitioned to the `port-present` state, and will then be removed.

`set`
Set bus-specific properties for a hotplug connector. The specified option string is a bus specific string of name and value pairs, as could be parsed by `getsubopt(3C)`. The names and values will be passed directly to the bus controller that manages the specified hotplug connector to perform a bus-specific function.

`get`
Display the current values of bus specific properties for a hotplug connector. The specified option string is a bus specific string of named properties, as could be parsed by `getsubopt(3C)`. The names will be passed directly to the bus controller to specify which properties should be returned. The current values of each named property will then be displayed.

Options The following options are supported:

`-l`, `--list-path`
Show full paths to connections and device nodes. By default, the `list` subcommand shows hotplug connectors, ports, and devices in the format of a tree. This option enables the display of full paths to each connection and device node.

`-v`, `--verbose`
Show verbose usage details. By default, the `list` subcommand shows only hotplug connectors, ports, and devices. This option enables the display of more detailed information about how the devices are currently consumed. Examples include mounted filesystems or plumbed network interfaces associated with individual devices.

`-f`, `--force`
Force the operation. Some change state operations that impact resources currently in use will fail with a warning. A forced operation will attempt to ignore these warnings and proceed.

This option should be used with extreme caution.

`-q`, `--query`
Query the operation. Instead of actually performing a change state operation, perform a test to predict if the operation would succeed or fail. If it would fail, show the error messages that would be expected if the operation had really been attempted.

It is not possible to predict every failure. An operation that succeeds during a query could still fail for another reason when actually attempted.

This option will not actually change the state of the system.

`-o` *options*, `--=`*options*
Specify bus-specific properties for a `set` or `get` command. The options string conforms to the `getsubopt(3C)` syntax convention.

For the `get` subcommand, there are two special options that can be used. The special options value of `help` will display all supported properties and their possible values. The special options value of `all` will display the current value of all supported properties.

For the `set` subcommand, there is one special option that can be used. The special options value of `help` will display all supported properties which can be set and their possible values.

See "Notes" section for the properties supported by bus controllers.

`-?`, `--help`
Display a brief help message on proper use of the command.

Examples

EXAMPLE 1 Showing All Hotplug Connections

The following command shows all hotplug connections:

```
# hotplug list -v
pci@0,0
        <pci.2,1> (ONLINE)
         pci108e,534a@2,1
                 [pci30] (EMPTY)
        <pci.e,0> (ONLINE)
         pci10de,5d@e
                 <pci.b,0> (ONLINE)
                 display@b
                 [NEM0] (ENABLED)
                 <pci.a,0> (ONLINE)
                 pci108e,534a@a,0
                         { Network interface nge0 }
                         { nge0: hosts IP addresses: 10.0.0.1 }
                 <pci.a,1> (MAINTENANCE)
                 pci108e,534a@a,1
                 [NEM1] (EMPTY)
         <pci.c,0> (OFFLINE)
                pci108e,534a@4
```

To show the full paths of hotplug connections and devices, enter the following command:

```
# hotplug list -l
/pci@0,0 <pci.2,1> (ONLINE)
/pci@0,0/pci108e,534a@2,1 [pci30] (EMPTY)
/pci@0,0 pci.e,0> (ONLINE)
/pci@0,0/pci10de,5d@e <pci.b,0> (ONLINE)
/pci@0,0/pci10de,5d@e/display@b
/pci@0,0/pci10de,5d@e [NEM0] (ENABLED)
/pci@0,0/pci10de,5d@e <pci.a,0> (ONLINE)
/pci@0,0/pci10de,5d@e/pci108e,534a@a,0
/pci@0,0/pci10de,5d@e <pci.a,1> (MAINTENANCE)
/pci@0,0/pci10de,5d@e/pci108e,534a@a,0
/pci@0,0/pci10de,5d@e [NEM1] (EMPTY)
/pci@0,0 pci.c,0> (OFFLINE)
/pci@0,0/pci108e,534a@4
```

EXAMPLE 2 Reporting Failure During State Change Operation

If a change of state operation fails, an explanation is displayed to describe the failure. An attempt to offline a hotplug port with dependent devices that are currently in use by the system might fail as follows:

```
# hotplug offline /pci@0,0/pci10de,5d@e pci.a,0
ERROR: devices or resources are busy.
pci108e,534a@a,0:
```

EXAMPLE 2 Reporting Failure During State Change Operation *(Continued)*

```
    { Network interface nge0 }
    { nge0: hosts IP addresses: 10.0.0.1 }
    { Plumbed IP Address }
```

EXAMPLE 3 Displaying Bus-Specific Properties and Values

The following command displays all supported bus-specific properties and their possible values:

```
# hotplug get -o help /pci@0,0 pci.2,1
power_led=<on|off|blink>
fault_led=<on|off|blink>
active_led=<on|off|blink>
attn_led=<on|off|blink>
card_type=<type description>
board_type=<type description>
```

EXAMPLE 4 Displaying Bus-Specific Options

The following command displays the card type and the current state of the Power LED of a PCI hotplug connector:

```
# hotplug get -o card_type,power_led /pci@0,0 pci.2,1
card_type=fibre
power_led=on
```

EXAMPLE 5 Setting a Bus-Specific Property

The following command turns on the attention LED of a PCI hotplug connector:

```
# hotplug set -o attn_led=on /pci@0,0 pci.2,1
```

Exit Status

`0`
Successful completion.

`1`
Invalid command line options were specified.

`2`
The specified path or connection does not exist.

`3`
A fatal error occurred. One or more error messages are displayed on standard error.

`4`
The hotplug service is not available.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs
Interface Stability	Committed

See Also cfgadm(1M), hotplugd(1M), getsubopt(3C), attributes(5)

Diagnostics The following error message is displayed on systems that do not have any supported I/O buses:

```
ERROR: there are no connections to display.
(See hotplug(1m) for more information.)
```

If this error message is seen, note that the system might still have other I/O devices that support hotplugging, through the cfgadm(1M) command instead of hotplug.

Notes The hotplug service (FMRI svc:/system/hotplug) must be enabled as a prerequisite for using the hotplug command. The service is disabled by default. See hotplugd(1M).

The authorization solaris.hotplug.modify must be granted in order to perform change-of-state operations. Alternatively, the rights profile "Hotplug Management" can be granted, which includes that authorization.

Verbose usage information is gathered from the RCM framework. Its format and content is subject to change.

The following bus specific properties are supported in PCI bus controllers:

power_led | fault_led | attn_led | active_led
: States of a specific LED of a slot. The value could be on, off, or blink.

  They can all be used with get subcommand, but only property attn_led can be used with set subcommand.

card_type | board_type
: Type of a card or board of a slot.

  They can all be used with get subcommand, but neither can be used with set subcommand.

Name hotplugd – hotplug daemon

Synopsis

```
/usr/lib/hotplugd [-d]
```

Description The hotplug daemon, `hotplugd`, provides user-level services for the management of hotplug connections. It is a system daemon started by the Service Management Facility (see `smf(5)`). Its fault management resource identifier (FMRI) is:

```
svc:/system/hotplug:default
```

Note that `hotplugd` is a Consolidation Private interface. See `attributes(5)`.

The `hotplug(1M)` command and any other client program that uses the private `libhotplug` library to query information about hotplug connections or initiate hotplug commands depends upon this daemon. The hotplug daemon is a door server which services requests from all `libhotplug` clients. The door interface is private.

Client applications use the private `libhotplug` interface to administer hotplug connections. `libhotplug` uses the door interface to administer hotplug connections through the hotplug daemon service. The hotplug daemon acts as a central location to serialize all hotplug operations and coordinate activities with all other parts of the system.

Options The following option is supported:

`-d`, `--debug`
: Run the daemon in standalone debug mode. Messages will be displayed on the controlling terminal instead of to syslog. And increased verbosity will be enabled to display more details about the internal operations of the daemon.

Examples **EXAMPLE 1** Enabling the Hotplug Service

The following command enables the hotplug service:

```
# svcadm enable svc:/system/hotplug:default
```

EXAMPLE 2 Disabling the Hotplug Service

The following command disables the hotplug service:

```
# svcadm disable svc:/system/hotplug:default
```

Errors The hotplug daemon uses `syslog(3C)` to report status and error messages. All of the messages are logged with the `LOG_DAEMON` facility. Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities, and informational messages are logged with the `LOG_INFO` priority. The default entries in the `/etc/syslog.conf` file log all of the hotplug daemon error messages to the `/var/adm/messages` log.

Files `/var/run/hotplugd_door`
: Hotplug daemon door

`/var/run/hotplugd_pid`
: Hotplug daemon lock file

`/usr/lib/hotplugd`
Hotplug daemon binary

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs, SUNWcs
Interface Stability	Consolidation Private

See Also `svcs(1)`, `hotplug(1M)`, `svcadm(1M)`, `syslog(3C)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`

Notes The `hotplugd` service is managed by the service management facility, `smf(5)`, under the fault management resource identifier (FMRI):

```
svc:/system/hotplug:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command. To administer the service, the authorization `solaris.smf.manage.hotplug` must be granted. Alternatively, the rights profile “Hotplug Management” can be granted.

The hotplug service must be enabled for the `hotplug(1M)` command and any other `libhotplug` client applications to function properly.

Name htable – convert DoD Internet format host table

Synopsis `/usr/sbin/htable` *filename*

Description `htable` converts a host table in the format specified by RFC 952 to the format used by the network library routines. Three files are created as a result of running `htable`: `hosts`, `networks`, and `gateways`. The `hosts` file is used by the `gethostbyname(3NSL)` routines in mapping host names to addresses. The `networks` file is used by the `getnetbyname(3SOCKET)` routines in mapping network names to numbers. The `gateways` file is used by the routing daemon to identify "passive" Internet gateways.

If any of the files `localhosts`, `localnetworks`, or `localgateways` are present in the current directory, the file's contents is prepended to the output file without interpretation. This allows sites to maintain local aliases and entries which are not normally present in the master database.

`htable` is best used in conjunction with the `gettable(1M)` program which retrieves the DoD Internet host table from a host.

Files `localhosts`

`localnetworks`

`localgateways`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `gettable(1M)`, `gethostbyname(3NSL)`, `getnetbyname(3SOCKET)`, `attributes(5)`
Harrenstien, Ken, Mary Stahl, and Elizabeth Feinler, *DoD Internet Host Table Specification*, RFC 952, Network Information Center, SRI International, Menlo Park, California, October 1985.

Notes `htable` does not properly calculate the `gateways` file.

Name ickey – install a client key for WAN boot

Synopsis

```
/usr/lib/inet/wanboot/ickey [-d] [-o type=3des]

/usr/lib/inet/wanboot/ickey [-d] [-o type=aes]

/usr/lib/inet/wanboot/ickey [-d] [-o type=sha1]
```

Description The `ickey` command is used to install WAN boot keys on a running UNIX system so that they can be used the next time the system is installed. You can store three different types of keys: 3DES and AES for encryption and an HMAC SHA-1 key for hashed verification.

`ickey` reads the key from standard input using `getpass(3C)` so that it does not appear on the command line. When installing keys on a remote system, you must take proper precautions to ensure that any keying materials are kept confidential. At a minimum, use `ssh(1)` to prevent interception of data in transit.

Keys are expected to be presented as strings of hexadecimal digits; they can (but need not) be preceeded by a `0x` or `0X`.

The `ickey` command has a single option, described below. An argument of the type `-o type=`*keytype* is required.

Options The `ickey` command the following option.

`-d` Delete the key specified by the *keytype* argument.

Exit Status On success, `ickey` exits with status 0; if a problem occurs, a diagnostic message is printed and `ickey` exits with non-zero status.

Files `/dev/openprom` WAN boot key storage driver

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbsup
Interface Stability	Unstable

See Also `ssh(1)`, `openprom(7D)`, `attributes(5)`

Name id – return user identity

Synopsis

```
/usr/bin/id [-p] [user]

/usr/bin/id -a [-p] [user]

/usr/xpg4/bin/id [-p] [user]

/usr/xpg4/bin/id -G [-n] [user]

/usr/xpg4/bin/id -g [-nr] [user]

/usr/xpg4/bin/id -u [-nr] [user]
```

Description If no *user* operand is provided, the `id` utility writes the user and group IDs and the corresponding user and group names of the invoking process to standard output. If the effective and real IDs do not match, both are written. If multiple groups are supported by the underlying system, `/usr/xpg4/bin/id` also writes the supplementary group affiliations of the invoking process.

If a *user* operand is provided and the process has the appropriate privileges, the user and group IDs of the selected user are written. In this case, effective IDs are assumed to be identical to real IDs. If the selected user has more than one allowable group membership listed in the group database, `/usr/xpg4/bin/id` writes them in the same manner as the supplementary groups described in the preceding paragraph.

Formats The following formats are used when the `LC_MESSAGES` locale category specifies the "C" locale. In other locales, the strings `uid`, `gid`, `euid`, `egid`, and `groups` may be replaced with more appropriate strings corresponding to the locale.

`"uid=%u(%s) gid=%u(%s)\n"` <*real user ID*>, <*user-name*>,
<*real group ID*>, <*group-name*>

If the effective and real user IDs do not match, the following are inserted immediately before the `\n` character in the previous format:

`" euid=%u(%s)"`

with the following arguments added at the end of the argument list:

<*effective user ID*>, <*effective user-name*>

If the effective and real group IDs do not match, the following is inserted directly before the `\n` character in the format string (and after any addition resulting from the effective and real user IDs not matching):

`" egid=%u(%s)"`

with the following arguments added at the end of the argument list:

<*effectivegroup-ID*>, <*effectivegroupname*>

If the process has supplementary group affiliations or the selected user is allowed to belong to multiple groups, the first is added directly before the `NEWLINE` character in the format string:

" groups=%u(%s)"

with the following arguments added at the end of the argument list:

<*supplementary group ID*>, <*supplementary group name*>

and the necessary number of the following added after that for any remaining supplementary group IDs:

",%u(%s)"

and the necessary number of the following arguments added at the end of the argument list:

<*supplementary group ID*>, <*supplementary group name*>

If any of the user ID, group ID, effective user ID, effective group ID or supplementary/multiple group IDs cannot be mapped by the system into printable user or group names, the corresponding (`%s`) and name argument is omitted from the corresponding format string.

When any of the options are specified, the output format is as described under OPTIONS.

Options The following option is supported by both `/usr/bin/id` and `/usr/xpg4/bin/id`. For `/usr/xpg4/bin/id`, `-p` is invalid if specified with any of the `-G`, `-g`, or `-u` options.

`-p` Reports additionally the current project membership of the invoking process. The project is reported using the format:

"projid=%u(%s)"

which is inserted prior to the `\n` character of the default format described in the `Formats` section. The arguments

<*project ID*>,<*project name*>

are appended to the end of the argument list. If the project ID cannot be mapped by the system into a printable project name, the corresponding `(%s)` and name argument is omitted from the corresponding format string.

/usr/bin/id The following option is supported for `/usr/bin/id` only:

`-a` Reports user name, user ID and all the groups to which the user belongs.

/usr/xpg4/bin/id The following options are supported for `/usr/xpg4/bin/id` only:

`-G` Outputs all different group IDs (effective, real and supplementary) only, using the format "`%u\n`". If there is more than one distinct group affiliation, output each such affiliation, using the format " `%u`", before the `NEWLINE` character is output.

`-g` Outputs only the effective group ID, using the format "`%u\n`".

`-n` Outputs the name in the format "`%s`" instead of the numeric ID using the format "`%u`".

`-r` Outputs the real ID instead of the effective ID.

`-u` Outputs only the effective user ID, using the format "`%u\n`".

Operands The following operand is supported:

user The user (login) name for which information is to be written.

Environment Variables See `environ(5)` for descriptions of the following environment variables that affect the execution of `id`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

/usr/bin/id

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu, SUNWcar

/usr/xpg4/bin/id

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxcu4
Interface Stability	Standard

See Also `fold(1)`, `logname(1)`, `who(1)`, `getgid(2)`, `getgroups(2)`, `getprojid(2)`, `getuid(2)`, `attributes(5)`, `environ(5)`, `standards(5)`

Notes Output produced by the `-G` option and by the default case could potentially produce very long lines on systems that support large numbers of supplementary groups.

Name idsconfig – prepare an iPlanet Directory Server (iDS) to be populated with data and serve LDAP clients

Synopsis

```
/usr/lib/ldap/idsconfig [-v] [-i input_configfile]
    [-o output_configfile]
```

Description Use the `idsconfig` tool to set up an iPlanet Directory Server (iDS). You can specify the input configuration file with the `-i` option on the command line. Alternatively, the tool will prompt the user for configuration information. The input configuration file is created by `idsconfig` with the `-o` option on a previous run.

The first time a server is set up, the user is prompted for all the required information. Future installations on that machine can use the configuration file previously generated by `idsconfig` using the `-o` option.

The output configuration file contains the directory administrator's password in clear text. Thus, if you are creating an output configuration file, take appropriate security precautions.

You should back up the directory server's configuration and data prior to running this command.

Options The following options are supported:

`-i` *input_configfile*	Specify the file name for `idsconfig` to use as a configuration file. This file will be read by `idsconfig`, and the values in the file will be used to configure the server. Do not manually edit *input_configfile*. The *input_configfile* is only partially validated, as `idsconfig` assumes that the file was created by a previous invocation of the command.
`-o` *output_configfile*	Create a configuration file.
`-v`	Verbose output.

Operands The following operands are supported:

input_configfile	Name of configuration file for `idsconfig` to use.
output_configfile	Configuration file created by `idsconfig`.

Examples

EXAMPLE 1 Prompting the User for Input

In the following example, the user is prompted for information to set up iDS.

```
example# idsconfig
```

EXAMPLE 2 Creating an Output Configuration File

In the following example, the user is prompted for information to set up iDS, and an output configuration file, `config.1`, is created when completed.

```
example# idsconfig -o config.1
```

EXAMPLE 3 Setting up iDS Using the Specified Configuration File

In the following example, iDS is set up by using the values specified in the configuration file, `config.1`. The verbose mode is specified, so detailed information will print to the screen.

```
example# idsconfig -v -i config.1
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu
Interface Stability	Evolving

See Also

`ldap(1)`, `ldapadd(1)`, `ldapdelete(1)`, `ldaplist(1)`, `ldapmodify(1)`, `ldapmodrdn(1)`, `ldapsearch(1)`, `ldap_cachemgr(1M)`, `ldapaddent(1M)`, `ldapclient(1M)`, `suninstall(1M)`, `resolv.conf(4)`, `attributes(5)`

Name ifconfig – configure network interface parameters

Synopsis

```
ifconfig interface [address_family] [address [/prefix_length]
 [dest_address]] [addif address [/prefix_length]]
 [removeif address [/prefix_length]] [arp | -arp]
 [auth_algs authentication algorithm] [encr_algs encryption algorithm]
 [encr_auth_algs authentication algorithm] [auto-revarp]
 [broadcast address] [deprecated | -deprecated]
 [preferred | -preferred] [destination dest_address]
 [ether [address]] [failover | -failover] [group
 [name | ""]] [index if_index] [metric n] [modlist]
 [modinsert mod_name@pos] [modremove mod_name@pos]
 [mtu n] [netmask mask] [plumb] [unplumb] [private
 | -private] [nud | -nud] [set [address] [/netmask]]
 [standby | -standby] [subnet subnet_address] [tdst
 tunnel_dest_address] [token address/prefix_length]
 [tsrc tunnel_src_address] [trailers | -trailers]
 [up] [down] [usesrc [name | none]] [xmit | -xmit]
 [encaplimit n | -encaplimit] [thoplimit n] [router
 | -router] [zone zonename | -zone | -all-zones]

ifconfig [address_family] interface {auto-dhcp | dhcp} [primary]
 [wait seconds] drop | extend | inform | ping
 | release | start | status
```

Description The command `ifconfig` is used to assign an address to a network interface and to configure network interface parameters. The `ifconfig` command must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters. If no option is specified, `ifconfig` displays the current configuration for a network interface. If an address family is specified, `ifconfig` reports only the details specific to that address family. Only privileged users may modify the configuration of a network interface. Options appearing within braces (`{ }`) indicate that one of the options must be specified.

DHCP Configuration The forms of `ifconfig` that use the `auto-dhcp` or `dhcp` arguments are used to control the Dynamic Host Configuration Protocol ("DHCP") configuration of the interface. In this mode, `ifconfig` is used to control operation of `dhcpagent(1M)`, the DHCP client daemon. Once an interface is placed under DHCP control by using the `start` operand, `ifconfig` should not, in normal operation, be used to modify the address or characteristics of the interface. If the address of an interface under DHCP is changed, `dhcpagent` will remove the interface from its control.

Options The following options are supported:

`addif` *address*
: Create the next unused logical interface on the specified physical interface. If the physical interface is part of a multipathing group, the logical interface can be added to a different physical interface in the same group.

`all-zones`
Make the interface available to every shared-IP zone on the system. The appropriate zone to which to deliver data is determined using the `tnzonecfg` database. This option is available only if the system is configured with the Solaris Trusted Extensions feature.

The `tnzonecfg` database is described in the `tnzonecfg(4)` man page, which is part of the *Solaris Trusted Extensions Reference Manual*.

`anycast`
Marks the logical interface as an anycast address by setting the `ANYCAST` flag. See "INTERFACE FLAGS," below, for more information on anycast.

`-anycast`
Marks the logical interface as not an anycast address by clearing the `ANYCAST` flag.

`arp`
Enable the use of the Address Resolution Protocol ("ARP") in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between IPv4 addresses and MAC addresses.

`-arp`
Disable the use of the ARP on a physical interface.

`auth_algs` *authentication algorithm*
For a tunnel, enable IPsec AH with the authentication algorithm specified. The algorithm can be either a number or an algorithm name, including *any* to express no preference in algorithm. All IPsec tunnel properties must be specified on the same command line. To disable tunnel security, specify an `auth_alg` of `none`.

It is now preferable to use the `ipsecconf(1M)` command when configuring a tunnel's security properties. If `ipsecconf` was used to set a tunnel's security properties, this keyword will not affect the tunnel.

`auto-dhcp`
Use DHCP to automatically acquire an address for this interface. This option has a completely equivalent alias called `dhcp`.

For IPv6, the interface specified must be the zeroth logical interface (the physical interface name), which has the link-local address.

primary
Defines the interface as the `primary`. The interface is defined as the preferred one for the delivery of client-wide configuration data. Only one interface can be the primary at any given time. If another interface is subsequently selected as the primary, it replaces the previous one. Nominating an interface as the primary one will not have much significance once the client work station has booted, as many applications will already have started and been configured with data read from the previous primary interface.

wait *seconds*
The `ifconfig` command will wait until the operation either completes or for the interval specified, whichever is the sooner. If no wait interval is given, and the operation is one that cannot complete immediately, `ifconfig` will wait 30 seconds for the requested operation to complete. The symbolic value `forever` may be used as well, with obvious meaning.

drop
Remove the specified interface from DHCP control without notifying the DHCP server, and record the current lease for later use. Additionally, for IPv4, set the IP address to zero and mark the interface as "down." For IPv6, unplumb all logical interfaces plumbed by `dhcpagent`.

extend
Attempt to extend the lease on the interface's IP address. This is not required, as the agent will automatically extend the lease well before it expires.

inform
Obtain network configuration parameters from DHCP without obtaining a lease on IP addresses. This is useful in situations where an IP address is obtained through mechanisms other than DHCP.

ping
Check whether the interface given is under DHCP control, which means that the interface is managed by the DHCP agent and is working properly. An exit status of `0` means success.

release
Relinquish the IP addresses on the interface by notifying the server and discard the current lease. For IPv4, mark the interface as "down." For IPv6, all logical interfaces plumbed by `dhcpagent` are unplumbed.

start
Start DHCP on the interface.

status
Display the DHCP configuration status of the interface.

`auto-revarp`
Use the Reverse Address Resolution Protocol (RARP) to automatically acquire an address for this interface. This will fail if the interface does not support RARP; for example, IPoIB (IP over InfiniBand), and on IPv6 interfaces.

`broadcast` *address*
For IPv4 only. Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all `1`'s. A "+" (plus sign) given for the broadcast value causes the broadcast address to be reset to a default appropriate for the (possibly new) address and netmask. The arguments of `ifconfig` are interpreted left to right. Therefore

```
example% ifconfig -a netmask + broadcast +
```

and

```
example% ifconfig -a broadcast + netmask +
```

may result in different values being assigned for the broadcast addresses of the interfaces.

`deprecated`
Marks the logical interface as deprecated. An address associated with a deprecated interface will not be used as source address for outbound packets unless either there are no other addresses available on the interface or the application has bound to this address explicitly. The status display shows `DEPRECATED` as part of flags. See INTERFACE FLAGS for information on the flags supported by `ifconfig`.

`-deprecated`
Marks a logical interface as not deprecated. An address associated with such an interface could be used as a source address for outbound packets.

`preferred`
Marks the logical interface as preferred. This option is only valid for IPv6 addresses. Addresses assigned to preferred logical interfaces are preferred as source addresses over all other addresses configured on the system, unless the address is of an inappropriate scope relative to the destination address. Preferred addresses are used as source addresses regardless of which physical interface they are assigned to. For example, you can configure a preferred source address on the loopback interface and advertise reachability of this address by using a routing protocol.

`-preferred`
Marks the logical interface as not preferred.

`destination` *dest_address*
Set the destination address for a point-to point interface.

`dhcp`
This option is an alias for option `auto-dhcp`

`down`
Mark a logical interface as "down". (That is, turn off the `IFF_UP` bit.) When a logical interface is marked "down," the system does not attempt to use the address assigned to that interface as a source address for outbound packets and will not recognize inbound packets destined to that address as being addressed to this host. Additionally, when all logical interfaces on a given physical interface are "down," the physical interface itself is disabled.

When a logical interface is down, all routes that specify that interface as the output (using the `-ifp` option in the `route(1M)` command or `RTA_IFP` in a `route(7P)` socket) are removed from the forwarding table. Routes marked with `RTF_STATIC` are returned to the table if the interface is brought back up, while routes not marked with `RTF_STATIC` are simply deleted.

When all logical interfaces that could possibly be used to reach a particular gateway address are brought down (specified without the interface option as in the previous paragraph), the affected gateway routes are treated as though they had the `RTF_BLACKHOLE` flag set. All matching packets are discarded because the gateway is unreachable.

`encaplimit` *n*
Set the tunnel encapsulation limit for the interface to n. This option applies to IPv4–in-IPv6 and IPv6–in-IPv6 tunnels only. The tunnel encapsulation limit controls how many more tunnels a packet may enter before it leaves any tunnels, that is, the tunnel nesting level.

`-encaplimit`
Disable generation of the tunnel encapsulation limit. This option applies only to IPv4–in-IPv6 and IPv6–in-IPv6 tunnels.

`encr_auth_algs` *authentication algorithm*
For a tunnel, enable IPsec ESP with the authentication algorithm specified. It can be either a number or an algorithm name, including `any` or `none`, to indicate no algorithm preference. If an ESP encryption algorithm is specified but the authentication algorithm is not, the default value for the ESP authentication algorithm will be `any`.

It is now preferable to use the `ipsecconf(1M)` command when configuring a tunnel's security properties. If `ipsecconf` was used to set a tunnel's security properties, this keyword will not affect the tunnel.

`encr_algs` *encryption algorithm*
For a tunnel, enable IPsec ESP with the encryption algorithm specified. It can be either a number or an algorithm name. Note that all IPsec tunnel properties must be specified on the same command line. To disable tunnel security, specify the value of `encr_alg` as `none`. If an ESP authentication algorithm is specified, but the encryption algorithm is not, the default value for the ESP encryption will be `null`.

It is now preferable to use the `ipsecconf(1M)` command when configuring a tunnel's security properties. If `ipsecconf` was used to set a tunnel's security properties, this keyword will not affect the tunnel.

`ether` [*address*]
If no address is given and the user is root or has sufficient privileges to open the underlying device, then display the current Ethernet address information.

Otherwise, if the user is root or has sufficient privileges, set the Ethernet address of the interfaces to *address*. The address is an Ethernet address represented as *x:x:x:x:x:x* where *x* is a hexadecimal number between 0 and FF. Similarly, for the IPoIB (IP over InfiniBand) interfaces, the address will be 20 bytes of colon-separated hex numbers between `0` and `FF`.

Some, though not all, Ethernet interface cards have their own addresses. To use cards that do not have their own addresses, refer to section 3.2.3(4) of the IEEE 802.3 specification for

a definition of the locally administered address space. The use of multipathing groups should be restricted to those cards with their own addresses (see `MULTIPATHING GROUPS`).

`-failover`
Mark the logical interface as a non-failover interface. Addresses assigned to non-failover logical interfaces will not failover when the interface fails. Status display shows `NOFAILOVER` as part of flags.

`failover`
Mark the logical interface as a failover interface. An address assigned to such an interface will failover when the interface fails. Status display does not show `NOFAILOVER` as part of flags.

`group` [*name* |`""`]
Insert the logical interface in the multipathing group specified by *name*. To delete an interface from a group, use a null string `""`. When invoked on the logical interface with id zero, the status display shows the group name.

`index` *n*
Change the interface index for the interface. The value of *n* must be an interface index (*if_index*) that is not used on another interface. *if_index* will be a non-zero positive number that uniquely identifies the network interface on the system.

`metric` *n*
Set the routing metric of the interface to *n*; if no value is specified, the default is `0`. The routing metric is used by the routing protocol. Higher metrics have the effect of making a route less favorable. Metrics are counted as addition hops to the destination network or host.

`modinsert` *mod_name@pos*
Insert a module with name *mod_name* to the stream of the device at position *pos*. The position is relative to the stream head. Position `0` means directly under stream head.

Based upon the example in the `modlist` option, use the following command to insert a module with name `ipqos` under the `ip` module and above the firewall module:

```
example% ifconfig eri0 modinsert ipqos@2
```

A subsequent listing of all the modules in the stream of the device follows:

```
example% ifconfig eri0 modlist
0 arp
1 ip
2 ipqos
3 firewall
4 eri
```

`modlist`
List all the modules in the stream of the device.

The following example lists all the modules in the stream of the device:

```
example% ifconfig eri0 modlist
0 arp
1 ip
2 firewall
4 eri
```

`modremove` *mod_name@pos*
Remove a module with name *mod_name* from the stream of the device at position *pos*. The position is relative to the stream head.

Based upon the example in the `modinsert` option, use the following command to remove the firewall module from the stream after inserting the `ipqos` module:

```
example% ifconfig eri0 modremove firewall@3
```

A subsequent listing of all the modules in the stream of the device follows:

```
example% ifconfig eri0 modlist
0 arp
1 ip
2 ipqos
3 eri
```

Note that the core IP stack modules, for example, `ip` and `tun` modules, cannot be removed.

`mtu` *n*
Set the maximum transmission unit of the interface to *n*. For many types of networks, the `mtu` has an upper limit, for example, `1500` for Ethernet. This option sets the `FIXEDMTU` flag on the affected interface.

`netmask` *mask*
For IPv4 only. Specify how much of the address to reserve for subdividing networks into subnetworks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion. The mask can be specified in one of four ways:

1. with a single hexadecimal number with a leading 0x,
2. with a dot-notation address,
3. with a "+" (plus sign) address, or
4. with a pseudo host name/pseudo network name found in the network database `networks(4)`.

If a "+" (plus sign) is given for the netmask value, the mask is looked up in the `netmasks(4)` database. This lookup finds the longest matching netmask in the database by starting with the interface's IPv4 address as the key and iteratively masking off more and more low order

bits of the address. This iterative lookup ensures that the netmasks(4) database can be used to specify the netmasks when variable length subnetmasks are used within a network number.

If a pseudo host name/pseudo network name is supplied as the netmask value, netmask data may be located in the hosts or networks database. Names are looked up by first using gethostbyname(3NSL). If not found there, the names are looked up in getnetbyname(3SOCKET). These interfaces may in turn use nsswitch.conf(4) to determine what data store(s) to use to fetch the actual value.

For both inet and inet6, the same information conveyed by *mask* can be specified as a *prefix_length* attached to the *address* parameter.

nud
: Enables the neighbor unreachability detection mechanism on a point-to-point physical interface.

-nud
: Disables the neighbor unreachability detection mechanism on a point-to-point physical interface.

plumb
: Open the device associated with the physical interface name and set up the streams needed for IP to use the device. When used with a logical interface name, this command is used to create a specific named logical interface. An interface must be separately plumbed for use by IPv4 and IPv6. The *address_family* parameter controls whether the ifconfig command applies to IPv4 or IPv6.

  Before an interface has been plumbed, the interface will not show up in the output of the ifconfig -a command.

private
: Tells the in.routed routing daemon that a specified logical interface should not be advertised.

-private
: Specify unadvertised interfaces.

removeif *address*
: Remove the logical interface on the physical interface specified that matches the *address* specified. When the interface is part of a multipathing group, the logical interface will be removed from the physical interface in the group that holds the address.

router
: Enable IP forwarding on the interface. When enabled, the interface is marked ROUTER, and IP packets can be forwarded to and from the interface.

`-router`
Disable IP forwarding on the interface. IP packets are not forwarded to and from the interface.

`set`
Set the *address*, *prefix_length* or both, for a logical interface.

`standby`
Marks the physical interface as a standby interface. If the interface is marked `STANDBY` and is part of the multipathing group, the interface will not be selected to send out packets unless some other interface in the group has failed and the network access has been failed over to this standby interface.

The status display shows "`STANDBY, INACTIVE`" indicating that that the interface is a standby and is also inactive. `IFF_INACTIVE` will be cleared when some other interface belonging to the same multipathing group fails over to this interface. Once a failback happens, the status display will return to `INACTIVE`.

`-standby`
Turns off standby on this interface.

`subnet`
Set the subnet *address* for an interface.

`tdst` *tunnel_dest_address*
Set the destination address of a tunnel. The address should not be the same as the `dest_address` of the tunnel, because no packets leave the system over such a tunnel.

`thoplimit` *n*
Set the hop limit for a tunnel interface. The hop limit value is used as the `TTL` in the IPv4 header for the IPv6–in-IPv4 and IPv4–in-IPv4 tunnels. For IPv6–in-IPv6 and IPv4–in-IPv6 tunnels, the hop limit value is used as the hop limit in the IPv6 header.

`token` *address*/*prefix_length*
Set the IPv6 token of an interface to be used for address autoconfiguration.

```
example% ifconfig eri0 inet6 token ::1/64
```

`trailers`
This flag previously caused a nonstandard encapsulation of IPv4 packets on certain link levels. Drivers supplied with this release no longer use this flag. It is provided for compatibility, but is ignored.

`-trailers`
Disable the use of a "trailer" link level encapsulation.

`tsrc` *tunnel_src_address*
Set the source address of a tunnel. This is the source address on an outer encapsulating IP header. It must be an address of another interface already configured using `ifconfig`.

`unplumb`
Close the device associated with this physical interface name and any streams that `ifconfig` set up for IP to use the device. When used with a logical interface name, the logical interface is removed from the system. After this command is executed, the device name will no longer appear in the output of `ifconfig -a`.

`up`
Mark a logical interface "up". This happens automatically when assigning the first address to a logical interface. The `up` option enables an interface after an `ifconfig down`, which reinitializes the hardware.

`usesrc` [*name* | `none`]
Specify a physical interface to be used for source address selection. If the keyword `none` is used, then any previous selection is cleared.

When an application does not choose a non-zero source address using `bind(3SOCKET)`, the system will select an appropriate source address based on the outbound interface and the address selection rules (see `ipaddrsel(1M)`).

When `usesrc` is specified and the specified interface is selected in the forwarding table for output, the system looks first to the specified physical interface and its associated logical interfaces when selecting a source address. If no usable address is listed in the forwarding table, the ordinary selection rules apply. For example, if you enter:

```
# ifconfig eri0 usesrc vni0
```

...and `vni0` has address 10.0.0.1 assigned to it, the system will prefer 10.0.0.1 as the source address for any packets originated by local connections that are sent through `eri0`. Further examples are provided in the EXAMPLES section.

While you can specify any physical interface (or even loopback), be aware that you can also specify the virtual IP interface (see `vni(7d)`). The virtual IP interface is not associated with any physical hardware and is thus immune to hardware failures. You can specify any number of physical interfaces to use the source address hosted on a single virtual interface. This simplifies the configuration of routing-based multipathing. If one of the physical interfaces were to fail, communication would continue through one of the remaining, functioning physical interfaces. This scenario assumes that the reachability of the address hosted on the virtual interface is advertised in some manner, for example, through a routing protocol.

Because the `ifconfig preferred` option is applied to all interfaces, it is coarser-grained than the `usesrc` option. It will be overridden by `usesrc` and `setsrc` (route subcommand), in that order.

The use of the `usesrc` option is mutually exclusive of the IP multipathing `ifconfig` options, `group` and `standby`. That is, if an interface is already part of a IP multipathing

group or specified as a `standby` interface, then it cannot be specified with a `usesrc` option, and vice-versa. For more details on IP multipathing, see `in.mpathd`(1M) and the *System Administration Guide: IP Services*.

`xmit`
Enable a logical interface to transmit packets. This is the default behavior when the logical interface is up.

`-xmit`
Disable transmission of packets on an interface. The interface will continue to receive packets.

`zone` *zonename*
Place the logical interface in zone *zonename*. The named zone must be active in the kernel in the ready or running state. The interface is unplumbed when the zone is halted or rebooted. The zone must be configure to be an shared-IP zone. `zonecfg`(1M) is used to assign network interface names to exclusive-IP zones.

`-zone`
Place IP interface in the global zone. This is the default.

Operands The *interface* operand, as well as address parameters that affect it, are described below.

interface
A string of one of the following forms:

- *name physical-unit*, for example, `eri0` or `ce1`
- *name physical-unit*:*logical-unit*, for example, `eri0:1`
- `ip.tun`*N* or `ip6.tun`*N*, for tunnels

If the interface name starts with a dash (-), it is interpreted as a set of options which specify a set of interfaces. In such a case, `-a` must be part of the options and any of the additional options below can be added in any order. If one of these interface names is given, the commands following it are applied to all of the interfaces that match.

`-a`
Apply the command to all interfaces of the specified address family. If no address family is supplied, either on the command line or by means of `/etc/default/inet_type`, then all address families will be selected.

`-d`
Apply the commands to all "down" interfaces in the system.

`-D`
Apply the commands to all interfaces not under DHCP (Dynamic Host Configuration Protocol) control.

`-u`
Apply the commands to all "up" interfaces in the system.

`-Z`
Apply the commands to all interfaces in the user's zone.

`-4`
Apply the commands to all IPv4 interfaces.

`-6`
Apply the commands to all IPv6 interfaces.

address_family
The address family is specified by the *address_family* parameter. The `ifconfig` command currently supports the following families: `inet` and `inet6`. If no address family is specified, the default is `inet`.

`ifconfig` honors the `DEFAULT_IP` setting in the `/etc/default/inet_type` file when it displays interface information . If `DEFAULT_IP` is set to `IP_VERSION4`, then `ifconfig` will omit information that relates to IPv6 interfaces. However, when you explicitly specify an address family (`inet` or `inet6`) on the `ifconfig` command line, the command line overrides the `DEFAULT_IP` settings.

address
For the IPv4 family (`inet`), the *address* is either a host name present in the host name data base (see `hosts(4)`) or in the Network Information Service (NIS) map `hosts`, or an IPv4 address expressed in the Internet standard “dot notation”.

For the IPv6 family (`inet6`), the *address* is either a host name present in the host name data base (see `hosts(4)`) or in the Network Information Service (NIS) map `ipnode`, or an IPv6 address expressed in the Internet standard colon-separated hexadecimal format represented as *x:x:x:x:x:x:x:x* where *x* is a hexadecimal number between `0` and `FFFF`.

prefix_length
For the IPv4 and IPv6 families (`inet` and `inet6`), the *prefix_length* is a number between 0 and the number of bits in the address. For `inet`, the number of bits in the address is 32; for `inet6`, the number of bits in the address is 128. The *prefix_length* denotes the number of leading set bits in the netmask.

dest_address
If the *dest_address* parameter is supplied in addition to the *address* parameter, it specifies the address of the correspondent on the other end of a point-to-point link.

tunnel_dest_address
An address that is or will be reachable through an interface other than the tunnel being configured. This tells the tunnel where to send the tunneled packets. This address must not be the same as the interface destination address being configured.

tunnel_src_address
An address that is attached to an already configured interface that has been configured “up” with `ifconfig`.

Interface Flags The `ifconfig` command supports the following interface flags. The term "address" in this context refers to a logical interface, for example, `eri0:0`, while "interface " refers to the physical interface, for example, `eri0`.

`ADDRCONF`
The address is from stateless `addrconf`. The stateless mechanism allows a host to generate its own address using a combination of information advertised by routers and locally available information. Routers advertise prefixes that identify the subnet associated with the link, while the host generates an "interface identifier" that uniquely identifies an interface in a subnet. In the absence of information from routers, a host can generate link-local addresses. This flag is specific to IPv6.

`ANYCAST`
Indicates an `anycast` address. An `anycast` address identifies the nearest member of a group of systems that provides a particular type of service. An `anycast` address is assigned to a group of systems. Packets are delivered to the nearest group member identified by the `anycast` address instead of being delivered to all members of the group.

`BROADCAST`
This `broadcast` address is valid. This flag and `POINTTOPOINT` are mutually exclusive

`CoS`
This interface supports some form of Class of Service (CoS) marking. An example is the 802.1D user priority marking supported on VLAN interfaces.

Note that this flag is only set on interfaces over VLAN links and over Ethernet links that have their `dladm(1M)` `tagmode` link property set to normal.

`DEPRECATED`
This address is deprecated. This address will not be used as a source address for outbound packets unless there are no other addresses on this interface or an application has explicitly bound to this address. An IPv6 deprecated address will eventually be deleted when not used, whereas an IPv4 deprecated address is often used with IP network multipathing IPv4 test addresses, which are determined by the setting of the `NOFAILOVER` flag. Further, the `DEPRECATED` flag is part of the standard mechanism for renumbering in IPv6.

`DHCP`
DHCP is used to manage this address.

`DUPLICATE`
The logical interface has been disabled because the IP address configured on the interface is a duplicate. Some other node on the network is using this address. If the address was configured by DHCP or is temporary, the system will choose another automatically, if possible. Otherwise, the system will attempt to recover this address periodically and the interface will recover when the conflict has been removed from the network. Changing the address or netmask, or setting the logical interface to `up` will restart duplicate detection. Setting the interface to `down` terminates recovery and removes the `DUPLICATE` flag.

FAILED
The interface has failed. New addresses cannot be created on this interface. If this interface is part of an IP network multipathing group, a failover will occur to another interface in the group, if possible

FIXEDMTU
The MTU has been set using the `-mtu` option. This flag is read-only. Interfaces that have this flag set have a fixed MTU value that is unaffected by dynamic MTU changes that can occur when drivers notify IP of link MTU changes.

INACTIVE
Indicates that the interface is not currently being used for regular traffic by the system. New addresses cannot be created on this interface. The flag is set automatically on standby interfaces. It can also be set when the system detects that a failed interface has been repaired and `FAILBACK`=no is configured in `/etc/default/mpathd`. The flag is cleared when the interface fails or when a failover to that interface occurs.

LOOPBACK
Indicates that this is the loopback interface.

MIP
Indicates that mobile IP controls this interface.

MULTI_BCAST
Indicates that the broadcast address is used for multicast on this interface.

MULTICAST
The interface supports multicast. IP assumes that any interface that supports hardware broadcast, or that is a point-to-point link, will support multicast.

NOARP
There is no address resolution protocol (ARP) for this interface that corresponds to all interfaces for a device without a broadcast address. This flag is specific to IPv4.

NOFAILOVER
This address will not failover if the interface fails. IP network multipathing test addresses must be marked `nofailover`.

NOLOCAL
The interface has no address , just an on-link subnet.

NONUD
`NUD` is disabled on this interface. `NUD` (neighbor unreachability detection) is used by a node to track the reachability state of its neighbors, to which the node actively sends packets, and to perform any recovery if a neighbor is detected to be unreachable. This flag is specific to IPv6.

NORTEXCH
The interface does not exchange routing information. For RIP-2, routing packets are not sent over this interface. Additionally, messages that appear to come over this interface

receive no response. The subnet or address of this interface is not included in advertisements over other interfaces to other routers.

`NOXMIT`
Indicates that the address does not transmit packets. RIP-2 also does not advertise this address.

`OFFLINE`
Indicates that the interface has been offlined. New addresses cannot be created on this interface. Interfaces in an IP network multipathing group are offlined prior to removal and replacement using dynamic reconfiguration.

`POINTOPOINT`
Indicates that the address is a point-to-point link. This flag and `BROADCAST` are mutually exclusive

`PREFERRED`
This address is a preferred IPv6 source address. This address will be used as a source address for IPv6 communication with all IPv6 destinations, unless another address on the system is of more appropriate scope. The `DEPRECATED` flag takes precedence over the `PREFERRED` flag.

`PRIVATE`
Indicates that this address is not advertised. For RIP-2, this interface is used to send advertisements. However, neither the subnet nor this address are included in advertisements to other routers.

`ROUTER`
Indicates that IP packets can be forwarded to and from the interface.

`RUNNING`
Indicates that the required resources for an interface are allocated. For some interfaces this also indicates that the link is up.

`STANDBY`
Indicates that this is a standby interface to be used on failures. Only interfaces in an IP network multipathing group should be designated as standby interfaces. If this interface is part of a IP network multipathing group, the interface will not be selected to send out packets unless some other interface in the group fails over to it.

`TEMPORARY`
Indicates that this is a temporary IPv6 address as defined in RFC 3041.

`UNNUMBERED`
This flag is set when the local IP address on the link matches the local address of some other link in the system

`UP`
Indicates that the interface is up, that is, all the routing entries and the like for this interface have been set up.

`VIRTUAL`
Indicates that the physical interface has no underlying hardware. It is not possible to transmit or receive packets through a virtual interface. These interfaces are useful for configuring local addresses that can be used on multiple interfaces. (See also the `-usesrc` option.)

`XRESOLV`
Indicates that the interface uses an IPv6 external resolver.

Logical Interfaces

Solaris TCP/IP allows multiple logical interfaces to be associated with a physical network interface. This allows a single machine to be assigned multiple IP addresses, even though it may have only one network interface. Physical network interfaces have names of the form *driver-name physical-unit-number*, while logical interfaces have names of the form *driver-name physical-unit-number*:*logical-unit-number*. A physical interface is configured into the system using the `plumb` command. For example:

```
example% ifconfig eri0 plumb
```

Once a physical interface has been "plumbed", logical interfaces associated with the physical interface can be configured by separate `-plumb` or `-addif` options to the `ifconfig` command.

```
example% ifconfig eri0:1 plumb
```

allocates a specific logical interface associated with the physical interface `eri0`. The command

```
example% ifconfig eri0 addif 192.168.200.1/24 up
```

allocates the next available logical unit number on the `eri0` physical interface and assigns an *address* and *prefix_length*.

A logical interface can be configured with parameters (*address*,*prefix_length*, and so on) different from the physical interface with which it is associated. Logical interfaces that are associated with the same physical interface can be given different parameters as well. Each logical interface must be associated with an existing and "up" physical interface. So, for example, the logical interface `eri0:1` can only be configured after the physical interface `eri0` has been plumbed.

To delete a logical interface, use the `-unplumb` or - `removeif` options. For example,

```
example% ifconfig eri0:1 down unplumb
```

will delete the logical interface `eri0:1`.

Multipathing Groups

Physical interfaces that share the same IP broadcast domain can be collected into a multipathing group using the `group` keyword. Interfaces assigned to the same multipathing group are treated as equivalent and outgoing traffic is spread across the interfaces on a per-IP-destination basis. In addition, individual interfaces in a multipathing group are monitored for failures; the addresses associated with failed interfaces are automatically transferred to other functioning interfaces within the group.

For more details on IP multipathing, see in.mpathd(1M) and the *System Administration Guide: IP Services*. See netstat(1M) for per-IP-destination information.

Configuring Ipv6 Interfaces

When an IPv6 physical interface is plumbed and configured "up" with ifconfig, it is automatically assigned an IPv6 link-local address for which the last 64 bits are calculated from the MAC address of the interface.

```
example% ifconfig eri0 inet6 plumb up
```

The following example shows that the link-local address has a prefix of fe80::/10.

```
example% ifconfig eri0 inet6
ce0: flags=2000841<UP,RUNNING,MULTICAST,IPv6>
          mtu 1500 index 2
        inet6 fe80::a00:20ff:fe8e:f3ad/10
```

Link-local addresses are only used for communication on the local subnet and are not visible to other subnets.

If an advertising IPv6 router exists on the link advertising prefixes, then the newly plumbed IPv6 interface will autoconfigure logical interface(s) depending on the prefix advertisements. For example, for the prefix advertisement 2001:0db8:3c4d:0:55::/64, the autoconfigured interface will look like:

```
eri0:2: flags=2080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6>
          mtu 1500 index 2
        inet6 2001:0db8:3c4d:55:a00:20ff:fe8e:f3ad/64
```

Even if there are no prefix advertisements on the link, you can still assign global addresses manually, for example:

```
example% ifconfig eri0 inet6 addif \
2001:0db8:3c4d:55:a00:20ff:fe8e:f3ad/64 up
```

To configure boot-time defaults for the interface eri0, place the following entry in the /etc/hostname6.eri0 file:

```
addif  2001:0db8:3c4d:55:a00:20ff:fe8e:f3ad/64 up
```

Configuring IPv6/IPv4 tunnels

An IPv6 over IPv4 tunnel interface can send and receive IPv6 packets encapsulated in an IPv4 packet. Create tunnels at both ends pointing to each other. IPv6 over IPv4 tunnels require the tunnel source and tunnel destination IPv4 and IPv6 addresses. Solaris 8 supports both automatic and configured tunnels. For automatic tunnels, an IPv4-compatible IPv6 address is used. The following demonstrates auto-tunnel configuration:

```
example% ifconfig ip.atun0 inet6 plumb
example% ifconfig ip.atun0 inet6 tsrc IPv4-address \
   ::IPv4 address/96 up
```

where IPv4–address is the IPv4 address of the interface through which the tunnel traffic will flow, and IPv4-address, ::<IPv4–address>, is the corresponding IPv4-compatible IPv6 address.

The following is an example of a configured tunnel:

```
example% ifconfig ip.tun0 inet6 plumb tsrc my-ipv4-address \
   tdst peer-ipv4-address up
```

This creates a configured tunnel between `my-ipv4-address` and `peer-ipv4-address` with corresponding link-local addresses. For tunnels with global or site-local addresses, the logical tunnel interfaces need to be configured in the following form:

```
example% ifconfig ip.tun0 inet6 addif my-v6-address peer-v6-address up
```

For example,

```
example% ifconfig ip.tun0 inet6 plumb tsrc 109.146.85.57 \
   tdst 109.146.85.212 up
example% ifconfig ip.tun0 inet6 addif 2::45 2::46 up
```

To show all IPv6 interfaces that are up and configured:

```
example% ifconfig -au6
ip.tun0: flags=2200851<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6>
          mtu 1480 index 3
        inet tunnel src 109.146.85.57   tunnel dst 109.146.85.212
        tunnel security settings  -->  use 'ipsecconf -ln -i ip.tun1'
        tunnel hop limit 60
        inet6 fe80::6d92:5539/10 --> fe80::6d92:55d4
ip.tun0:1: flags=2200851<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6>
          mtu 1480 index 3
        inet6 2::45/128 --> 2::46
```

In the output above, note the line that begins with "`tunnel security settings`". The content of this line varies according to whether and how you have set your security settings. See "Display of Tunnel Security Settings," below.

Configuring IPv4/IPv6 Tunnels

An IPv4 over IPv6 tunnel interface can send and receive IPv4 packets encapsulated in an IPv6 packet. Create tunnels at both ends pointing to each other. IPv4 over IPv6 tunnels require the tunnel source and tunnel destination IPv6 and IPv4 addresses. The following demonstrates auto-tunnel configuration:

```
example% ifconfig ip6.tun0 inet plumb tsrc my-ipv6-address \
   tdst peer-ipv6-address my-ipv4-address \
   peer-ipv4-address up
```

This creates a configured tunnel between *my-ipv6-address* and *peer-ipv6-address* with *my-ipv4-address* and *peer-ipv4-address* as the endpoints of the point-to-point interface, for example:

```
example% ifconfig ip6.tun0 inet plumb tsrc fe80::1 tdst fe80::2 \
10.0.0.208 10.0.0.210 up
```

To show all IPv4 interfaces that are up and configured:

```
example% ifconfig -au4
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
     inet 127.0.0.1 netmask ff000000
eri0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 \
index 2
     inet 172.17.128.208 netmask ffffff00 broadcast 172.17.128.255
ip6.tun0: flags=10008d1<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST,IPv4> \
mtu 1460
     index 3
     inet6 tunnel src fe80::1 tunnel dst fe80::2
     tunnel security settings  -->  use 'ipsecconf -ln -i ip.tun1'
     tunnel hop limit 60 tunnel encapsulation limit 4
     inet 10.0.0.208 --> 10.0.0.210 netmask ff000000
```

In the output above, note the line that begins with "`tunnel security settings`". The content of this line varies according to whether and how you have set your security settings. See "Display of Tunnel Security Settings," below.

Display of Tunnel Security Settings

The `ifconfig` output for tunneled interfaces indicates security settings, if present, for a tunnel. The content of the line showing your settings differs depending on how you have made your settings:

- If you set your security policy using the `ifconfig -auth_algs`, `-encr_algs`, and `-encr_auth_algs` options and do not use ipsecconf(1M), `ifconfig` displays your settings for each of these options.
- If you set your security policy using ipsecconf(1M) with the `tunnel` keyword (the preferred method), `ifconfig` displays:

  ```
  tunnel security settings  -->  use 'ipsecconf -ln -i ip.tun1'
  ```

  ...in effect, hiding your settings from those without privileges to view them.

  If you do net set security policy, using either `ifconfig` or `ipsecconf`, there is no tunnel security setting displayed.

Examples

EXAMPLE 1 Using the `ifconfig` Command

If your workstation is not attached to an Ethernet, the network interface, for example, `eri0`, should be marked "down" as follows:

```
example% ifconfig eri0 down
```

EXAMPLE 2 Printing Addressing Information

To print out the addressing information for each interface, use the following command:

```
example% ifconfig -a
```

EXAMPLE 3 Resetting the Broadcast Address

To reset each interface's broadcast address after the netmasks have been correctly set, use the next command:

```
example% ifconfig -a broadcast +
```

EXAMPLE 4 Changing the Ethernet Address

To change the Ethernet address for interface `ce0`, use the following command:

```
example% ifconfig ce0 ether aa:1:2:3:4:5
```

EXAMPLE 5 Configuring an IP-in-IP Tunnel

To configure an IP-in-IP tunnel, first plumb it with the following command:

```
example% ifconfig ip.tun0 plumb
```

Then configure it as a point-to-point interface, supplying the tunnel source and the tunnel destination:

```
example% ifconfig ip.tun0 myaddr mydestaddr tsrc another_myaddr \
          tdst a_dest_addr up
```

Use `ipsecconf(1M)`, as described above, to configure tunnel security properties.

EXAMPLE 6 Configuring 6to4 Tunnels

To configure 6to4 tunnels, use the following commands:

```
example% ifconfig ip.6to4tun0 inet6 plumb
example% ifconfig ip.6to4tun0 inet6 tsrc IPv4-address 6to4-address/64 up
```

IPv4-address denotes the address of the encapsulating interface. *6to4-address* denotes the address of the local IPv6 address of form 2002:*IPv4-address*:*SUBNET-ID*:*HOSTID*.

The long form should be used to resolve any potential conflicts that might arise if the system administrator utilizes an addressing plan where the values for `SUBNET-ID` or `HOSTID` are reserved for something else.

After the interface is plumbed, a 6to4 tunnel can be configured as follows:

```
example% ifconfig ip.6to4tun0 inet6 tsrc IPv4-address up
```

This short form sets the address. It uses the convention:

```
2002:IPv4-address::1
```

The `SUBNET-ID` is 0, and the `HOSTID` is 1.

EXAMPLE 7 Configuring IP Forwarding on an Interface

To enable IP forwarding on a single interface, use the following command:

```
example% ifconfig eri0 router
```

To disable IP forwarding on a single interface, use the following command:

```
example% ifconfig eri0 -router
```

EXAMPLE 8 Configuring Source Address Selection Using a Virtual Interface

The following command configures source address selection such that every packet that is locally generated with no bound source address and going out on `qfe2` prefers a source address hosted on `vni0`.

```
example% ifconfig qfe2 usesrc vni0
```

The `ifconfig -a` output for the `qfe2` and `vni0` interfaces displays as follows:

```
qfe2: flags=1100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4> mtu
  1500 index 4
  usesrc vni0
  inet 1.2.3.4 netmask ffffff00 broadcast 1.2.3.255
  ether 0:3:ba:17:4b:e1
vni0: flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL>
  mtu 0 index 5
  srcof qfe2
  inet 3.4.5.6 netmask ffffffff
```

Observe, above, the `usesrc` and `srcof` keywords in the `ifconfig` output. These keywords also appear on the logical instances of the physical interface, even though this is a per-physical interface parameter. There is no `srcof` keyword in `ifconfig` for configuring interfaces. This information is determined automatically from the set of interfaces that have `usesrc` set on them.

The following command, using the `none` keyword, undoes the effect of the preceding `ifconfig usersrc` command.

```
example% ifconfig qfe2 usesrc none
```

Following this command, `ifconfig -a` output displays as follows:

```
qfe2: flags=1100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4> mtu
  1500 index 4
  inet 1.2.3.4 netmask ffffff00 broadcast 1.2.3.255
  ether 0:3:ba:17:4b:e1
vni0: flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL>
  mtu 0 index 5
  inet 3.4.5.6 netmask ffffffff
```

Note the absence of the `usesrc` and `srcof` keywords in the output above.

EXAMPLE 9 Configuring Source Address Selection for an IPv6 Address

The following command configures source address selection for an IPv6 address, selecting a source address hosted on `vni0`.

```
example% ifconfig qfe1 inet6 usesrc vni0
```

Following this command, `ifconfig -a` output displays as follows:

```
qfe1: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 3
  usesrc vni0
  inet6 fe80::203:baff:fe17:4be0/10
  ether 0:3:ba:17:4b:e0
vni0: flags=2002210041<UP,RUNNING,NOXMIT,NONUD,IPv6,VIRTUAL> mtu 0
  index 5
  srcof qfe1
  inet6 fe80::203:baff:fe17:4444/128
vni0:1: flags=2002210040<RUNNING,NOXMIT,NONUD,IPv6,VIRTUAL> mtu 0
  index 5
  srcof qfe1
  inet6 fec0::203:baff:fe17:4444/128
vni0:2: flags=2002210040<RUNNING,NOXMIT,NONUD,IPv6,VIRTUAL> mtu 0
  index 5
  srcof qfe1
  inet6 2000::203:baff:fe17:4444/128
```

Depending on the scope of the destination of the packet going out on `qfe1`, the appropriately scoped source address is selected from `vni0` and its aliases.

EXAMPLE 10 Using Source Address Selection with Shared-IP Zones

The following is an example of how the `usesrc` feature can be used with the `zones(5)` facility in Solaris. The following commands are invoked in the global zone:

```
example% ifconfig hme0 usesrc vni0
example% ifconfig eri0 usesrc vni0
example% ifconfig qfe0 usesrc vni0
```

Following the preceding commands, the `ifconfig -a` output for the virtual interfaces would display as:

```
vni0: flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL>
   mtu 0 index 23
   srcof hme0 eri0 qfe0
   inet 10.0.0.1 netmask ffffffff
vni0:1:
   flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL> mtu 0
   index 23
   zone test1
   srcof hme0 eri0 qfe0
   inet 10.0.0.2 netmask ffffffff
```

EXAMPLE 10 Using Source Address Selection with Shared-IP Zones *(Continued)*

```
vni0:2:
   flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL> mtu 0
   index 23
   zone test2
   srcof hme0 eri0 qfe0
   inet 10.0.0.3 netmask ffffffff
vni0:3:
   flags=20011100c1<UP,RUNNING,NOARP,NOXMIT,ROUTER,IPv4,VIRTUAL> mtu 0
   index 23
   zone test3
   srcof hme0 eri0 qfe0
   inet 10.0.0.4 netmask ffffffff
```

There is one virtual interface alias per zone (`test1`, `test2`, and `test3`). A source address from the virtual interface alias in the same zone is selected. The virtual interface aliases were created using `zonecfg(1M)` as follows:

```
example% zonecfg -z test1
zonecfg:test1> add net
zonecfg:test1:net> set physical=vni0
zonecfg:test1:net> set address=10.0.0.2
```

The `test2` and `test3` zone interfaces and addresses are created in the same way.

EXAMPLE 11 Turning Off DHCPv6

The following example shows how to disable automatic use of DHCPv6 on all interfaces, and immediately shut down DHCPv6 on the interface named `hme0`. See `in.ndpd(1M)` and `ndpd.conf(4)` for more information on the automatic DHCPv6 configuration mechanism.

```
example% echo ifdefault StatefulAddrConf false >> /etc/inet/ndpd.conf
example% pkill -HUP -x in.ndpd
example% ifconfig hme0 dhcp release
```

Files

`/etc/netmasks`
: Netmask data.

`/etc/default/inet_type`
: Default Internet protocol type.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability for command-line options	Committed

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability for command output	Uncommitted

See Also dhcpinfo(1), dhcpagent(1M), dladm(1M), in.mpathd(1M), in.ndpd(1M), in.routed(1M), ipsecconf(1M), ndd(1M), netstat(1M), zoneadm(1M), zonecfg(1M), ethers(3SOCKET), gethostbyname(3NSL), getnetbyname(3SOCKET), hosts(4), inet_type(4), ndpd.conf(4), netmasks(4), networks(4), nsswitch.conf(4), attributes(5), privileges(5), zones(5), arp(7P), ipsecah(7P), ipsecesp(7P), tun(7M)

System Administration Guide: IP Services

Diagnostics `ifconfig` sends messages that indicate if:

- the specified interface does not exist
- the requested address is unknown
- the user is not privileged and tried to alter an interface's configuration

Notes Do not select the names `broadcast`, `down`, `private`, `trailers`, `up` or other possible option names when you choose host names. If you choose any one of these names as host names, it can cause unusual problems that are extremely difficult to diagnose.

Name if_mpadm – change operational status of interfaces within a multipathing group

Synopsis

```
/usr/sbin/if_mpadm -d interface_name

/usr/sbin/if_mpadm -r interface_name
```

Description Use the `if_mpadm` utility to change the operational status of interfaces that are part of an IP multipathing group. If the interface is operational, you can use `if_mpadm -d` to detach or off-line the interface. If the interface is off-lined, use `if_mpadm -r` to revert it to its original state.

When a network interface is off-lined, all network access fails over to a different interface in the IP multipathing group. Any addresses that do not failover are brought down. Network access includes unicast, broadcast, and multicast for IPv4 and unicast and multicast for IPv6. Addresses marked with `IFF_NOFAILOVER` do not failover. They are marked down. After an interface is off-lined, the system will not use the interface for any outbound or inbound traffic, and the interface can be safely removed from the system without any loss of network access.

The `if_mpadm` utility can be applied only to interfaces that are part of an IP multipathing group.

Options The `if_mpadm` utility supports the following options:

`-d` *interface_name* Detach or off-line the interface specified by *interface_name*.

`-r` *interface_name* Reattach or undo the previous detach or off-line operation on the interface specified by *interface_name*. Unless the `-d` option was used to detach or off-line the interface, this option will fail.

Examples **EXAMPLE 1** Detaching an Interface

Use the following command to off-line or detach the interface. All network access will failover from `hme0` to other interfaces in the same IP multipathing group. If no other interfaces are in the same group, the operation will fail.

```
example% if_mpadm -d hme0
```

EXAMPLE 2 Reattaching an Off-line Interface

Use the following command to undo the previous operation. Network access will failback to `hme0`.

```
example% if_mpdadm -r hme0
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Unstable

See Also ifconfig(1M), in.mpathd(1M), attributes(5)

Diagnostics

`off-line failed as there is no other functional interface available in the multipathing group for failing over the network access.`

Description: This message means that other interfaces in the group are failed over already or the multipathing configuration was not suitable for completing a failover.

`off-line cannot be undone because multipathing configuration is not consistent across all the interfaces in the group.`

Description: This message means that some interfaces in the IP multipathing group are not configured consistently with other interfaces in the group, for example, one of the interfaces in the group does not have an `IFF_NOFAILOVER` address.

Name ifparse – parse `ifconfig` command line

Synopsis `/sbin/ifparse [-fs]` *addr_family commands*

Description Use the `ifparse` command to parse the `ifconfig(1M)` command line options and output substrings, one per line, as appropriate. If no options are specified, `ifparse` returns the entire `ifconfig` command line as a series of substrings, one per line.

Options The `ifparse` command supports the following options:

`-f` Lists only substrings of the `ifconfig` command line that are relevant to IP network multipath failover

`-s` Lists only substrings of the `ifconfig` command line that are not relevant to IP network multipath failover

Operands The `ifparse` command *does not* support the *interface* operand of the `ifconfig` command.

Examples **EXAMPLE 1** Parsing Command Line Options Relevant to Failover

The following example shows the use of the `ifparse` command to parse the command line options relevant to IP network multipath failover:

```
example# ifparse -f inet 1.2.3.4 up group one addif 1.2.3.5 -failover up
set 1.2.3.4 up
```

EXAMPLE 2 Parsing Command Line Options That Are Not Relevant to Failover

The following example shows the use of the `ifparse` command to parse the command line options that are not relevant to IP network multipath failover:

```
example# ifparse -s inet 1.2.3.4 up group one addif 1.2.3.5 -failover up
group one
addif 1.2.3.5 -failover up
```

EXAMPLE 3 Parsing the Command Line For All Options

The following example shows the use of the `ifparse` command to parse the command line for all `ifconfig` options:

```
example# ifparse inet 1.2.3.4 up group one addif 1.2.3.5 -failover up
group one
set 1.2.3.4 up
addif 1.2.3.5 -failover up
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Stability Level	Obsolete

See Also ifconfig(1M), attributes(5)

Diagnostics

`usage: -fs <addr_family> <commands>`
Description: This message indicates an invalid command line.

`ifparse: Not enough space`
Description: This message indicates insufficient memory.

`ifparse: dhcp not supported for inet6`
Description: DHCP operations are not supported for the `inet6` address family.

`ifparse: Operation <operation> not supported for <addr_family>`
Description: Most operations cannot be used with all address families. For example, the `broadcast` operation is not supported on the `inet6` address family.

`ifparse: no argument for <operation>`
Description: Some operations, for example `broadcast`, require an argument.

Notes The `ifparse` command is classified as an obsolete interface. It will likely be removed in a future release. You should not develop applications that depend upon this interface.

Name ikeadm – manipulate Internet Key Exchange (IKE) parameters and state

Synopsis

```
ikeadm [-np]

ikeadm [-np] get [debug | priv | stats | defaults]

ikeadm [-np] set [debug | priv] [level] [file]

ikeadm [-np] [get | del] [p1 | rule | preshared] [id]

ikeadm [-np] add [rule | preshared] { description }

ikeadm [-np] token [login | logout] PKCS#11_Token_Object

ikeadm [-np] [read | write] [rule | preshared | certcache] file

ikeadm [-np] [dump | pls | rule | preshared]

ikeadm [-np] flush [p1 | certcache]

ikeadm help
     [get | set | add | del | read | write | dump | flush | token]
```

Description The `ikeadm` utility retrieves information from and manipulates the configuration of the Internet Key Exchange (IKE) protocol daemon, `in.iked`(1M).

`ikeadm` supports a set of operations, which may be performed on one or more of the supported object types. When invoked without arguments, `ikeadm` enters interactive mode which prints a prompt to the standard output and accepts commands from the standard input until the end-of-file is reached.

Because `ikeadm` manipulates sensitive keying information, you must be superuser to use this command. Additionally, some of the commands available require that the daemon be running in a privileged mode, which is established when the daemon is started.

For details on how to use this command securely see Security.

Options The following options are supported:

`-n`
Prevent attempts to print host and network names symbolically when reporting actions. This is useful, for example, when all name servers are down or are otherwise unreachable.

`-p`
Paranoid. Do not print any keying material, even if saving Security Associations. Instead of an actual hexadecimal digit, print an `X` when this flag is turned on.

Usage

Commands The following commands are supported:

`add`
Add the specified object. This option can be used to add a new policy rule or a new preshared key to the current (running) in.iked configuration. When adding a new

preshared key, the command cannot be invoked from the command line, as it will contain keying material. The rule or key being added is specified using appropriate id-value pairs as described in the `ID FORMATS` section.

`del`
Delete a specific object or objects from `in.iked`'s current configuration. This operation is available for IKE (Phase 1) SAs, policy rules, and preshared keys. The object to be deleted is specified as described in the `Id Formats`.

`dump`
Display all objects of the specified type known to `in.iked`. This option can be used to display all Phase 1 SAs, policy rules, preshared keys, or the certificate cache. A large amount of output may be generated by this command.

`flush`
Remove all IKE (Phase 1) SAs or cached certificates from `in.iked`.

Note that flushing the `certcache` will also (as a side-effect) update IKE with any new certificates added or removed.

`get`
Lookup and display the specified object. May be used to view the current debug or privilege level, global statistics and default values for the daemon, or a specific IKE (Phase 1) SA, policy rule, or preshared key. The latter three object types require that identifying information be passed in; the appropriate specification for each object type is described below.

`help`
Print a brief summary of commands, or, when followed by a command, prints information about that command.

`read`
Update the current `in.iked` configuration by reading the policy rules or preshared keys from either the default location or from the file specified.

`set`
Adjust the current debug or privilege level. If the debug level is being modified, an output file may optionally be specified; the output file *must* be specified if the daemon is running in the background and is not currently printing to a file. When changing the privilege level, adjustments may only be made to lower the access level; it cannot be increased using ikeadm.

`write`
Write the current `in.iked` policy rule set or preshared key set to the specified file. A destination file must be specified. This command should not be used to overwrite the existing configuration files.

`token`
Log into a PKCS#11 token object and grant access to keying material or log out and invalidate access to keying material.

`token` can be run as a normal user with the following authorizations:

- `token` login: `solaris.network.ipsec.ike.token.login`
- `token` logout: `solaris.network.ipsec.ike.token.logout`

Object Types debug
Specifies the daemon's debug level. This determines the amount and type of output provided by the daemon about its operations. The debug level is actually a bitmask, with individual bits enabling different types of information.

Description	Flag	Nickname
Certificate management	0x0001	cert
Key management	0x0002	key
Operational	0x0004	op
Phase 1 SA creation	0x0008	phase1
Phase 2 SA creation	0x0010	phase2
PF_KEY interface	0x0020	pfkey
Policy management	0x0040	policy
Proposal construction	0x0080	prop
Door interface	0x0100	door
Config file processing	0x0200	config
All debug flags	0x3ff	all

When specifying the debug level, either a number (decimal or hexadecimal) or a string of nicknames may be given. For example, `88`, `0x58`, and `phase1+phase2+policy` are all equivalent, and will turn on debug for `phase 1 sa` creation, `phase 2 sa` creation, and policy management. A string of nicknames may also be used to remove certain types of information; `all-op` has the effect of turning on all debug *except* for operational messages; it is equivalent to the numbers `1019` or `0x3fb`.

priv
Specifies the daemon's access privilege level. The possible values are:

```
Description                 Level   Nickname
Base level                  0       base
Access to preshared key info 1      modkeys
Access to keying material   2       keymat
```

By default, `in.iked` is started at the base level. A command-line option can be used to start the daemon at a higher level. `ikeadm` can be used to lower the level, but it cannot be used to raise the level.

Either the numerical level or the nickname may be used to specify the target privilege level.

In order to get, add, delete, dump, read, or write preshared keys, the privilege level must at least give access to preshared key information. However, when viewing preshared keys (either using the get or dump command), the key itself will only be available if the privilege level gives access to keying material. This is also the case when viewing Phase 1 SAs.

stats

Global statistics from the daemon, covering both successful and failed Phase 1 SA creation.

Reported statistics include:

- Count of current P1 SAs which the local entity initiated
- Count of current P1 SAs where the local entity was the responder
- Count of all P1 SAs which the local entity initiated since boot
- Count of all P1 SAs where the local entity was the responder since boot
- Count of all attempted P1 SAs since boot, where the local entity was the initiator; includes failed attempts
- Count of all attempted P1 SAs since boot, where the local entity was the responder; includes failed attempts
- Count of all failed attempts to initiate a P1 SA, where the failure occurred because the peer did not respond
- Count of all failed attempts to initiate a P1 SA, where the peer responded
- Count of all failed P1 SAs where the peer was the initiator
- Whether a PKCS#11 library is in use, and if applicable, the PKCS#11 library that is loaded. See Example 11.

defaults

Display default values used by the `in.iked` daemon. Some values can be overriden in the daemon configuration file (see `ike.config(4)`); for these values, the token name is displayed in the `get defaults` output. The output will reflect where a configuration token has changed the default.

Default values might be ignored in the event a peer system makes a valid alternative proposal or they can be overriden by per-rule values established in `ike.config`. In such instances, a `get defaults` command continues to display the default values, not the values used to override the defaults.

p1

An IKE Phase 1 SA. A `p1` object is identified by an IP address pair or a cookie pair; identification formats are described below.

rule
An IKE policy rule, defining the acceptable security characteristics for Phase 1 SAs between specified local and remote identities. A rule is identified by its label; identification formats are described below.

preshared
A preshared key, including the local and remote identification and applicable IKE mode. A preshared key is identified by an IP address pair or an identity pair; identification formats are described below.

Id Formats

Commands like `add`, `del`, and `get` require that additional information be specified on the command line. In the case of the delete and get commands, all that is required is to minimally identify a given object; for the add command, the full object must be specified.

Minimal identification is accomplished in most cases by a pair of values. For IP addresses, the local addr and then the remote addr are specified, either in dot-notation for IPv4 addresses, colon-separated hexadecimal format for IPv6 addresses, or a host name present in the host name database. If a host name is given that expands to more than one address, the requested operation will be performed multiple times, once for each possible combination of addresses.

Identity pairs are made up of a local type-value pair, followed by the remote type-value pair. Valid types are:

prefix
An address prefix.

fqdn
A fully-qualified domain name.

domain
Domain name, synonym for fqdn.

user_fqdn
User identity of the form *user*@fqdn.

mailbox
Synonym for `user_fqdn`.

A cookie pair is made up of the two cookies assigned to a Phase 1 Security Association (SA) when it is created; first is the initiator's, followed by the responder's. A cookie is a 64-bit number.

Finally, a label (which is used to identify a policy rule) is a character string assigned to the rule when it is created.

Formatting a rule or preshared key for the add command follows the format rules for the in.iked configuration files. Both are made up of a series of id-value pairs, contained in curly braces (`{` and `}`). See `ike.config(4)` and `ike.preshared(4)` for details on the formatting of rules and preshared keys.

Security The `ikeadm` command allows a privileged user to enter cryptographic keying information. If an adversary gains access to such information, the security of IPsec traffic is compromised. The following issues should be taken into account when using the `ikeadm` command.

- Is the TTY going over a network (interactive mode)?

  If it is, then the security of the keying material is the security of the network path for this TTY's traffic. Using `ikeadm` over a clear-text telnet or rlogin session is risky. Even local windows may be vulnerable to attacks where a concealed program that reads window events is present.

- Is the file accessed over the network or readable to the world (read/write commands)?

  A network-mounted file can be sniffed by an adversary as it is being read. A world-readable file with keying material in it is also risky.

If your source address is a host that can be looked up over the network, and your naming system itself is compromised, then any names used will no longer be trustworthy.

Security weaknesses often lie in misapplication of tools, not the tools themselves. It is recommended that administrators are cautious when using the `ikeadm` command. The safest mode of operation is probably on a console, or other hard-connected TTY.

For additional information regarding this subject, see the afterward by Matt Blaze in Bruce Schneier's *Applied Cryptography: Protocols, Algorithms, and Source Code in C.*

Examples **EXAMPLE 1** Emptying out all Phase 1 Security Associations

The following command empties out all Phase 1 Security Associations:

```
example# ikeadm flush p1
```

EXAMPLE 2 Displaying all Phase 1 Security Associations

The following command displays all Phase 1 Security Associations:

```
example# ikeadm dump p1
```

EXAMPLE 3 Deleting a Specific Phase 1 Security Association

The following command deletes the specified Phase 1 Security Associations:

```
example# ikeadm del p1 local_ip remote_ip
```

EXAMPLE 4 Adding a Rule From a File

The following command adds a rule from a file:

```
example# ikeadm add rule rule_file
```

EXAMPLE 5 Adding a Preshared Key

The following command adds a preshared key:

EXAMPLE 5 Adding a Preshared Key *(Continued)*

```
example# ikeadm
     ikeadm> add preshared { localidtype ip localid local_ip
             remoteidtype ip remoteid remote_ip ike_mode main
             key 1234567890abcdef1234567890abcdef }
```

EXAMPLE 6 Saving All Preshared Keys to a File

The following command saves all preshared keys to a file:

```
example# ikeadm write preshared target_file
```

EXAMPLE 7 Viewing a Particular Rule

The following command views a particular rule:

```
example# ikeadm get rule rule_label
```

EXAMPLE 8 Reading in New Rules from `ike.config`

The following command reads in new rules from the ike.config file:

```
example# ikeadm read rules
```

EXAMPLE 9 Lowering the Privilege Level

The following command lowers the privilege level:

```
example# ikeadm set priv base
```

EXAMPLE 10 Viewing the Debug Level

The following command shows the current debug level

```
example# ikeadm get debug
```

EXAMPLE 11 Using stats to Verify Hardware Accelerator

The following example shows how stats may include an optional line at the end to indicate if IKE is using a PKCS#11 library to accelerate public-key operations, if applicable.

```
example# ikeadm get stats
Phase 1 SA counts:
Current:  initiator:     0    responder:      0
Total:    initiator:    21   responder:      27
Attempted:initiator:    21   responder:      27
Failed:   initiator:     0   responder:       0
             initiator fails include 0 time-out(s)
PKCS#11 library linked in from /opt/SUNWconn/lib/libpkcs11.so
example#
```

EXAMPLE 12 Displaying the Certificate Cache

The following command shows the certificate cache and the status of associated private keys, if applicable:

```
example# ikeadm dump certcache
```

EXAMPLE 13 Logging into a PKCS#11 Token

The following command shows logging into a PKCS#11 token object and unlocking private keys:

```
example# ikeadm token login "Sun Metaslot"
Enter PIN for PKCS#11 token:
ikeadm: PKCS#11 operation successful
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred. Writes an appropriate error message to standard error.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Not an Interface

See Also

`in.iked(1M)`, `ike.config(4)`, `ike.preshared(4)`, `attributes(5)`, `ipsec(7P)`

Schneier, Bruce, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, Second Edition, John Wiley & Sons, New York, NY, 1996.

Notes

As `in.iked` can run only in the global zone and exclusive-IP zones, this command is not useful in shared-IP zones.

Name ikecert – manipulates the machine's on-filesystem public-key certificate databases

Synopsis

```
ikecert certlocal
     [-a | -e | -h | -k | -l | -r | -U | -C | -L]
     [[-p] -T PKCS#11 token identifier]
     [option_specific_arguments]...

ikecert certdb [-a | -e | -h | -l | -r | -U | -C | -L]
     [[-p] -T PKCS#11 token identifier]
     [option_specific_arguments]...

ikecert certrldb [-a | -e | -h | -l | -r]
     [option_specific_arguments]...

ikecert tokens
```

Description The `ikecert` command manipulates the machine's on-filesystem public-key certificate databases. See the "Files" section, below.

`ikecert` has three subcommands, one for each of the three major repositories, plus one for listing available hardware tokens:

- `certlocal` deals with the private-key repository,
- `certdb` deals with the public-key repository, and:
- `certrldb` deals with the certificate revocation list (CRL) repository.
- `tokens` shows the available PKCS#11 tokens for a given PKCS#11 library.

The only supported PKCS#11 library and hardware is the Sun Cryptographic Accelerator 4000.

Options Except for `tokens`, each subcommand requires one option, possibly followed by one or more option-specific arguments.

The `tokens` subcommand lists all available tokens in the PKCS#11 library specified in `/etc/inet/ike/config`.

The following options are supported:

`-a`

certlocal
: When specified with the `certlocal` subcommand, this option installs (adds) a private key into the Internet Key Exchange (IKE) local ID database. The key data is read from standard input, and is in either Solaris-only format or unencrypted PKCS#8 DER format. Key format is automatically

detected. PKCS#8 key files in PEM format and files in password protected, encrypted format are not recognized, but can be converted appropriately using tools available in OpenSSL.

This option cannot be used with PKCS#11 hardware objects when the corresponding public certificate is not already present in the IKE database. When importing both a public certificate and a private key, the public portion must be imported first using the `certdb` subcommand.

certdb

When specified with the `certdb` subcommand, this option reads a certificate from standard input and adds it to the IKE certificate database. The certificate must be a `X.509` certificate in `PEM` `Base64` or `ASN.1` `BER` encoding. The certificate adopts the name of its identity.

This option can import a certificate into a PKCS#11 hardware key store one of two ways: Either a matching public key object *and* an existing private key object were created using the `certlocal -kc` option, or if a PKCS#11 token is explicitly specified using the `-T` option.

certrldb

When specified with the `certrldb` subcommand, this option installs (adds) a CRL into the IKE database. The CRL reads from standard input.

`-e` [`-f` pkcs8] *slot*

certlocal
: When specified with the `certlocal` subcommand, this option extracts a private key from the IKE local ID database. The key data are written to standard output. The slot specifies which private key to extract. Private keys are only extracted in binary/ber format.

  Use this option with extreme caution. See the "Security" section, below.

  This option will not work with PKCS#11 hardware objects.

  When used in conjunction with "`-f pkcs8`", the private key is extracted in unencrypted PKCS#8 format.

`-e` [`-f` *output-format*] `certspec`

certdb
: When specified with the `certdb` subcommand, this option extracts a certificate from the IKE certificate database which matches the certspec and writes it to standard output. The *output-format* option specifies the encoding format. Valid options are `PEM` and `BER`. This extracts the first matching identity. The default output format is `PEM`.

certrldb
: When specified with the `certrldb` subcommand, this option extracts a CRL from the IKE database. The key data are written to standard output. The `certspec` specifies which CRL that is extracted. The first one that matches in the database is extracted. See `NOTES`, below, for details on `certspec`

patterns.

`-kc -m` *keysize* `-t` *keytype* `-D` *dname* `-A` *altname*[...]
[`-S` *validity start_time*][`-F` *validity end_time*]
[`-T` *PKCS#11 token identifier*]

certlocal
When specified with the `certlocal` subcommand, this option generates a IKE public/private key pair and adds it into the local ID database. It also generates a certificate request and sends that to standard output. For details on the above options see Notes for details on the *dname* argument and see ALTERNATIVE NAMES for details on the *altname* argument(s) to this command.

If `-T` is specified, the hardware token will generate the pair of keys.

If `-p` is specified with `-T`, the PKCS#11 token pin is stored in the clear on-disk, with root-protected file permissions. If not specified, one must unlock the token with `ikeadm`(1M) once `in.iked`(1M) is running.

`-ks -m` *keysize* `-t` *keytype* `-D` *dname* `-A` *altname*[...]
[`-S` *validity start_time*][`-F` *validity end_time*]
[`-f` *output-format*][[`-p`] `-T` *PKCS#11 token identifier*]

certlocal
When specified with the `certlocal` subcommand, generates a public/private key pair and adds it into the local ID database. This option also generates a self-signed certificate and installs it into the certificate database. See `NOTES`, below, for details on the *dname* and *altname* arguments to this command.

If `-T` is specified, the hardware token will generate the pair of keys, and the self-signed certificate will also be stored in the hardware.

`-l` [`-v`] [*slot*]

certlocal
When specified with the `certlocal` subcommand, this option lists private keys in the local ID database. The `-v` option switches output to a verbose mode where the entire certificate is printed.

Use the `-v`*option with extreme caution.* See the "Security" section, below. The `-v` option will not work with PKCS#11 hardware objects.

`-l` [`-v`] [certspec]

certdb
When specified with the `certdb` subcommand, this option lists certificates in the IKE certificate database matching the certspec, if any pattern is given. The list displays the identity string of the certificates, as well as, the private key if in the key database. The `-v` switches the output to a verbose mode where the entire certificate is printed.

If the matching ceritifcate is on a hardware token, the token ID is also listed.

certrldb
When specified with the `certrldb` subcommand, this option lists the CRLs in the IKE database along with any certificates that reside in the database and match the Issuer Name. `certspec` can be used to

specify to list a specific CRL. The `-v` option switches the output to a verbose mode where the entire certificate is printed. See `NOTES`, below, for details on`certspec` patterns.

`-r` *slot*

certlocal
When specified with the `certlocal` subcommand, deletes the local ID in the specified slot. If there is a corresponding public key, it is not be deleted. If this slot is deemed as "corrupted" or otherwise unrecognizable, it is deleted as well.

If this is invoked on a PKCS#11 hardware object, it will also delete the PKCS#11 public key and private key objects. If the public key object was already deleted by `certdb -r`, that is not a problem.

`-r` certspec

certdb
Removes certificates from the IKE certificate database. Certificates matching the specified certificate pattern are deleted. Any private keys in the `certlocal` database corresponding to these certificates are not deleted. This removes the first matching identity.

If the pattern specifies a slot and the slot is deemed as "corrupted" or otherwise unrecognizable, it is deleted as well.

If this is invoked on a PKCS#11 hardware object, it will also delete the certificate and the PKCS#11 public key object. If the public key

object was already deleted by `certlocal -r`, that is not a problem.

certrldb
When specified with the `certrldb` subcommand, this option deletes the CRL with the given `certspec`.

`-U` slot

`certlocal`
When specified with the `certlocal` subcommand and the `-T` flag, this option unlinks a PKCS#11 private key object from the IKE database. There will be no attempt to access the hardware keystore or to validate or remove the on-token private key object. The object is simply disassociated from the IKE database.

`certdb`
When specified with the `certdb` subcommand and the `-T` flag, this option unlinks a PKCS#11 certificate object from the IKE database. There will be no attempt to access the hardware keystore or to validate or remove the on-token certificate or public key objects. The objects are simply disassociated from the IKE database.

`-C` certspec

certlocal
When specified with the `certlocal` subcommand, this option copies both the private key and its corresponding certificate and the public key from the on-disk keystore to the hardware keystore specified by its PKCS#11 token. This subcommand attempts

to create each of these components, even if one part fails. In all cases, the original on-disk private key and public certificate are still retained and must be deleted separately. Some hardware keystores, such as FIPS-140 compliant devices, may not support migration of private key objects in this manner.

certdb
When specified with the `certdb` subcommand, this option copies the certificate matching the given `certspec` and corresponding public key from the on-disk keystore to the hardware keystore specified by its PKCS#11 token. The original public certificate is still retained and must be deleted separately, if desired.

If `-p` is specified, the PKCS#11 token pin is stored in the clear on-disk, with root-protected file permissions. If not specified, one must unlock the token with `ikeadm(1M)` once `in.iked(1M)` is running.

-L pattern

certlocal
When specified with the `certlocal` subcommand, this option links an existing on-token private key object to the IKE database. The object itself remains on the token. This option simply lets the IKE infrastructure know that the object exists, as if it had been originally created on-token with the Solaris IKE utilities.

certdb
When specified with the `certdb` subcommand, this option links an

existing on-token certificate object to the IKE database. The object itself remains on the token. This option simply lets the IKE infrastructure know that the object exists, as if it had been originally created on-token with the Solaris IKE utilities.

If `-p` is specified, the PKCS#11 token pin is stored in the clear on-disk, with root-protected file permissions. If not specified, one must unlock the token with `ikeadm(1M)` once `in.iked(1M)` is running.

Parameters The following parameters are supported:

certspec

Specifies the pattern matching of certificate specifications. Valid `certspecs` are the Subject Name, Issuer Name, and Subject Alternative Names.

These can be specified as certificates that match the given `certspec` values and that do not match other `certspec` values. To signify a `certspec` value that is not supposed to be present in a certificate, place an `!` in front of the tag.

Valid `certspecs` are:

```
<Subject Names>
SUBJECT=<Subject Names>
ISSUER=<Issuer Names>
SLOT=<Slot Number in the certificate database>

Example:"ISSUER=C=US, O=SUN" IP=1.2.3.4 !DNS=example.com
Example:"C=US,   O=CALIFORNIA" IP=5.4.2.1 DNS=example.com
```

Valid arguments to the alternative names are as follows:

```
IP=<IPv4 address>
DNS=<Domain Name Server address>
EMAIL=<email (RFC 822) address>
URI=<Uniform Resource Indicator value>
DN=<LDAP Directory Name value>
RID=<Registered Identifier value>
```

Valid Slot numbers can be specified without the keyword tag. Alternative name can also be issued with keyword tags.

`-A`
Subject Alternative Names the certificate. The argument that follows the `-A` option should be in the form of *tag*=*value*. Valid tags are `IP`, `DNS`, `EMAIL`, `URI`, `DN`, and `RID` (See example below).

`-D`
`X.509` distinguished name for the certificate subject. It typically has the form of: `C`=country, `O`=organization, `OU`=organizational unit, `CN`=common name. Valid tags are: `C`, `O`, `OU`, and `CN`.

`-f`
Encoding output format. `pem` for `PEM` `Base64` or `ber` for `ASN.1` `BER`. If `-f` is not specified, `pem` is assumed.

`-F` *validity end_time*
Finish certificate validity time. If the `-F` flag is not specified, the validity end time is calculated at four years from the validity start time. See `NOTES` for an explanation for the validity date and time syntax.

`-m`
Key size. It can be `512`, `1024`, `2048`, `3072`, or `4096`. Use the following command to determine the key sizes supported by the Solaris Cryptographic Framework:

```
% cryptoadm list -vm
```

The mechanisms displayed by the preceding command are described in `pkcs11_softtoken(5)`. If your system has hardware acceleration, the mechanisms supported by the hardware will be listed in a separate section for each provider. Mechanisms can be any of:

```
CKM_RSA_PKCS_KEY_PAIR_GEN
CKM_DSA_KEY_PAIR_GEN
CKM_DH_PKCS_KEY_PAIR_GEN
```

Note – Some hardware does not support all key sizes. For example, the Sun Cryptographic Accelerator 4000's keystore (when using the `-T` option, below), supports only up to 2048-bit keys for RSA and 1024-bit keys for DSA.

`-S` *validity start_time*
Start certificate validity time. If the `-S` flag is not specified, the current date and time is used for the validity start time. See `NOTES`, below, for an explanation for the validity date and time syntax.

`-t`
Key type. It can be `rsa-sha1`, `rsa-md5`, or `dsa-sha1`.

`-T`
PKCS#11 token identifier for hardware key storage. This specifies a hardware device instance in conformance to the PKCS#11 standard. A PKCS#11 library must be specified in `/etc/inet/ike/config`. (See `ike.config(4)`.)

A token identifier is a 32-character space-filled string. If the token given is less than 32 characters long, it will be automatically padded with spaces.

If there is more than one PKCS#11 library on a system, keep in mind that only one can be specified at a time in `/etc/inet/ike/config`. There can be multiple tokens (each with individual key storage) for a single PKCS#11 library instance.

Security This command can save private keys of a public-private key pair into a file. Any exposure of a private key may lead to compromise if the key is somehow obtained by an adversary.

The PKCS#11 hardware object functionality can address some of the shortcomings of on-disk private keys. Because IKE is a system service, user intervention at boot is not desireable. The token's PIN, however, is still needed. The PINfor the PKCS#11 token, therefore, is stored where normally the on-disk cryptographic keys would reside. This design decision is deemed acceptable because, with a hardware key store, *possession* of the key is still unavailable, only *use* of the key is an issue if the host is compromised. Beyond the PIN, the security of `ikecert` then reduces to the security of the PKCS#11 implementation. The PKCS#11 implementation should be scrutinized also.

Refer to the afterword by Matt Blaze in Bruce Schneier's *Applied Cryptography: Protocols, Algorithms, and Source Code in C* for additional information.

Examples **EXAMPLE 1** Generating a Self-Signed Certificate

The following is an example of a self-signed certificate:

```
example# ikecert certlocal -ks -m 512 -t rsa-md5 -D "C=US, O=SUN" -A
IP=1.2.3.4
Generating, please wait...
Certificate generated.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIIBRDCB76ADAgECAgEBMA0GCSqGSIb3DQEBBAUAMBsxCzAJBgNVBAYTAlVTMQww
CgYDVQQKEwNTVU4wHhcNMDEwMzE0MDEzMDM1WhcNMDUwMzE0MDEzMDM1WjAbMQsw
CQYDVQQGEwJVUzEMMAoGA1UEChMDU1VOMFowDQYJKoZIhvcNAQEBBQADSQAwRgJB
APDhqpKgjgRoRUr6twTMTtSuNsReEnFoReVer!ztpXpQK6ybYlRH18JIqU/uCV/r
26R/cVXTy5qc5NbMwA40KzcCASOjIDAeMAsGA1UdDwQEAwIFoDAPBgNVHREECDAG
hwQBAgMEMA0GCSqGSIb3DQEBBAUAA0EApTRD23KzN95GMvPD71hwwCluksl KLVg8
f1xm9ZsHLPJLRxHFwsqqjAad4j4wwwriiUmGAHLTGB0lJMl8xsgxag==
-----END X509 CERTIFICATE-----
```

EXAMPLE 2 Generating a CA Request

Generating a `CA` request appears the same as the self-signed certificate. The only differences between the two is the option `-c` instead of `-s`, and the certificate data is a `CA` request.

```
example# ikecert certlocal -kc -m 512 -t rsa-md5 \
   -D "C=US, O=SUN" -A IP=1.2.3.4
```

EXAMPLE 3 A CA Request Using a Hardware Key Store

The following example illustrates the specification of a token using the `-T` option.

```
example# # ikecert certlocal -kc -m 1024 -t rsa-md5 -T vca0-keystore \
  -D "C=US, O=SUN" -A IP=1.2.3.4
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`non-zero`
An error occurred. Writes an appropriate error message to standard error.

Files

`/etc/inet/secret/ike.privatekeys/*`
Private keys. A private key *must* have a matching public-key certificate with the same filename in `/etc/inet/ike/publickeys/`.

`/etc/inet/ike/publickeys/*`
Public-key certificates. The names are only important with regard to matching private key names.

`/etc/inet/ike/crls/*`
Public key certificate revocation lists.

`/etc/inet/ike/config`
Consulted for the pathname of a PKCS#11 library.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also

`ikeadm(1M)`, `in.iked(1M)`, `getdate(3C)`, `ike.config(4)`, `attributes(5)`, `pkcs11_softtoken(5)`

Schneier, Bruce. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Second Edition. John Wiley & Sons. New York, NY. 1996.

RSA Labs, PKCS#11 v2.11: *Cryptographic Token Interface Standards*, November 2001.

Notes

The following is the validity date and time syntax when the `-F` or `-S` flags are used:

For relative dates, the syntax is as follows:

```
{+,-}[Ns][Nm][Nh][Nd][Nw][NM][Ny]
```

where:

N
represents an integer

s
represents seconds

m
represents minutes

h
represents hours

d
represents days

w
represents weeks

M
represents months

y
represents years

These parameters can be given in any order. For example, "+3d12h" is three and a half days from now, and "-3y2M" is three years and 2 months ago.

All parameters with fixed values can be added up in absolute seconds. Months and years, which have variable numbers of seconds, are calculated using calendar time. Months and years, which are not of fixed length, are defined such that adding a year or month means the same day next year or month. For instance, if it is Jan 26, 2005 and the certificate should expire 3 years and 1 month from today, the expiration (end validity time) date will be Feb 26, 2008. Overflows are dealt with accordingly. For example, one month from Jan 31, 2005 is March 3, 2005, since February has only 28 days.

For absolute dates, the syntax of the date formats included in the file `/etc/datemsk` are accepted (See `getdate(3C)` for details). Any date string prepended with a "+" or "-" is treated as a time relative to the current time, while others are treated as absolute dates. Sanity checking is also done to ensure that the end validity date is greater than the start validity date. For example, the following command would create a certificate with start date 1 day and 2 hours ago and an end date of Jan 22nd, 2007 at 12:00:00 local time.

```
# ikecert certlocal -ks -t rsa-sha1 -m 1024 \
    -D "CN=mycert, O=Sun, C=US" \
    -S -1d2h -F "01/22/2007 12:00:00"
```

As `in.iked(1M)` can run only in the global zone and exclusive-IP zones, this command is not useful in shared-IP zones.

Name imqadmin – launch the Message Queue administration console

Synopsis

```
/usr/bin/imqadmin [-javahome path]

/usr/bin/imqadmin -h

/usr/bin/imqadmin -v
```

Description `imqadmin` launches the graphical user interface application that performs most Message Queue administration tasks. These tasks include managing broker instances (including physical destinations) and administered objects.

Options The following options are supported:

`-h` Display usage help. The application is not launched.

`-javahome` *path* Specify a path to an alternate Java 2 compatible runtime.

`-v` Display version information.

Environment Variables The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` Specify the Java 2 compatible runtime. When this environment variable is not set it defaults to `/usr/j2se`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `$HOME/.imq/admin/brokerlist.properties`
Contains user settings, a list of broker instances being managed.

`$HOME/.imq/admin/objectstorelist.properties`
Contains user settings, a list of object stores being managed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiqu

See Also `imqbrokerd(1M)`, `imqcmd(1M)`, `imqdbmgr(1M)`, `imqkeytool(1M)`, `imqobjmgr(1M)`, `imqusermgr(1M)`, `attributes(5)`

Sun Java System Message Queue Administrator's Guide

Name imqbrokerd – start a Message Queue broker instance

Synopsis

```
/usr/bin/imqbrokerd [option...]

/usr/bin/imqbrokerd -h
```

Description `imqbrokerd` starts an instance of the Message Queue broker. The Message Queue broker is the main component of a Message Queue message server. The broker performs reliable delivery of messages to and from Java Message Service (JMS) clients.

`imqbrokerd` uses command line options to specify broker configuration properties.

Options The following options are supported:

`-backup` *fileName*
: Back up a Master Broker's configuration change record to *fileName*. This option only applies to broker clusters.

`-cluster` *brokerList*
: Specify the list of broker instances which are connected in a cluster. This list is merged with the list in the imq.cluster.brokerlist property. This option only applies to broker clusters.

  brokerList is a comma-separated list of broker instances, each specified by *hostName*:*port* (the host on which the broker instance is running and the port number it is using) If you don't specify a value for *hostName*, `localhost` is used. If you don't specify a value for *port*, the value of `7676` is used. For example: `host1:8899,host2,:7878`.

`-dbpassword` *password*
: Specify the password for a plugged-in JDBC-compliant database used as a Message Queue data store.

`-dbuser` *userName*
: Specify the user name for a plugged-in JDBC-compliant data store.

`-D`*property*`-=`*value*
: Set the specified broker configuration property to the *value*. The system does not validate either the configuration property or *value*. Therefore, spelling, formatting, and case is important. Message Queue can not set incorrect values passed using the `-D` option.

`-force`
: Perform action without user confirmation. This option only applies when you use the `-remove` *instance* option, which normally requires confirmation.

`-h`	Display usage help. Execute nothing else on the command line.
`-javahome` *path*	Specify the path to an alternate Java 2-compatible Java Development Kit (JDK) or Java Runtime Environment (JRE) The default is to use the runtime bundled with the operating system.
`-ldappassword` *password*	Specify the password for accessing an LDAP user repository when using an LDAP server (as opposed to a built-in flat-file repository) to authenticate users of a Message Queue message server.
`-license` [*name*]	Specify the license to load, if different from the default for your Message Queue product edition. If you don't specify a license name, this lists all licenses installed on the system. Depending on the installed Message Queue edition, the values for *name* are `pe` (Platform Edition-basic features), `try` (Platform Edition-90-day trial enterprise features), and `unl` (Enterprise Edition).
`-loglevel` *level*	Specify the logging level. Valid values for *level* are `NONE`, `ERROR`, `WARNING`, or `INFO`. The default value is `INFO`.
`-metrics` *int*	Report metrics at a specific interval. Specify *int* as the number of seconds.
`-name` *brokerName*	Specify the instance name of this broker and use the corresponding instance configuration file. If you do not specify a broker name, the name of the file is set to `imqbroker`. If you run more than one instance of a broker on the same host, each must have a unique name.
`-passfile` *filename*	Specify the name of the file from which to read the passwords for the SSL keystore, LDAP user repository, or JDBC-compliant database.
`-password` *keypassword*	Specify the password for the SSL certificate keystore.
`-port` *number*	Specify the broker's Port Mapper port number. By default, this is set to `7676`. To run two instances of a broker on the same server, each broker's Port

Mapper must have a different port number. JMS clients connect to the broker instance using this port number.

`-remove` *instance*

Remove the broker instance. Delete the instance configuration file, log files, data store, and other files and directories associated with the broker instance. This option requires user confirmation unless you also specify the `-force` option.

`-reset` store|messages|durables|props

Reset the data store (or a subset of the store) or resets the configuration properties of the broker instance when the broker instance is started. The action depends on the argument provided.

store	Clear all persistent data in the data store, including messages, durable subscriptions, and transaction information store.
messages	Clear all persistent messages durable.
durables	Clear all durable subscriptions.
props	Clear all configuration information in the `config.props` instance configuration file. All properties assume default values.

`-restore` *filename*

Replace the Master Broker's configuration change record with the specified backup file. This file must have been previously created using the `-backup` option. This option only applies to broker clusters.

`-shared`

Specify that the jms connection service be implemented using the shared threadpool model, in which threads are shared among connections to increase the number of connections supported by a broker instance.

`-silent`

Turn off logging to the console.

`-tty`

Display all messages be to the console. `WARNING` and `ERROR` level messages are displayed on the console by default.

`-upgrade-store-nobackup`
: Specify that an earlier, incompatible version Message Queue data store is automatically removed when migrating to Message Queue 3.5 format.

  If you do not use this option, you must manually delete the earlier data store. This applies to both built-in (flat-file) persistence and plugged-in (JDBC-compliant) persistence. Migration of the earlier data store to a Message Queue 3.5 data store takes place the first time you start a Message Queue 3.5 broker instance on an earlier version data store.

`-version`
: Display the version number of the installed product.

`-vmargs` *are* [[*arg*]…]
: Specify arguments to pass to the Java VM. Separate arguments with spaces. If you want to pass more than one argument or if an argument contains a space, use enclosing quotation marks. For example:

```
imqbrokerd -tty -vmargs " -Xmx128m -Xincgc"
```

Environment Variables

The following environment variables affect the execution of this command:

`IMQ_JAVAHOME`
: Specify the Java 2 compatible runtime. When this environment variable is not set it defaults to `/usr/j2se`.

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Files

`/etc/init.d/imq`
: Shell script for starting `imqbrokerd`. This file looks at the `/etc/imq/imqbrokerd.conf` file.

`/etc/imq/imqbrokerd.conf`
: Configuration file which controls the behavior of the broker startup script.

`/etc/imq/passwd`
: Flat file user repository for authenticating users.

`/etc/imq/accesscontrol.properties`
: Controls client access to broker functionality.

`/etc/imq/passfile.sample`
: Sample passfile used by the `-passfile` option.

`/var/imq/instances/brokerName/props/config.properties`
Broker instance configuration file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiqu

See Also `imqadmin(1M)`, `imqcmd(1M)`, `imqdbmgr(1M)`, `imqkeytool(1M)`, `imqobjmgr(1M)`, `imqusermgr(1M)`, `attributes(5)`

Sun Java System Message Queue Administrator's Guide

Name imqcmd – manage Message Queue brokers

Synopsis

```
/usr/bin/imqcmd subcommand argument [option...]

/usr/bin/imqcmd [-h | -H]

/usr/bin/imqcmd -v
```

Description `imqcmd` manages the Message Queue broker, including resources such as connection services, physical destinations, durable subscriptions, and transactions. The utility provides a number of subcommands for managing these resources.

`imqcmd` supports many subcommands. Basic connection and authentication is required for the execution of every `imqcmd` subcommand. Use the `-secure` option to specify secure connections. Subcommands and their corresponding arguments and options follow the `imqcmd` command on the command line. See `USAGE` and `OPTIONS`.

Options The following options are supported:

`-b` *hostName*`:`*port*
: Specify the name of the host on which the broker instance is running and the port number it is using.

  The default value is `localhost:7676`. If you do not specify the `-b` option, `imqcmd` uses the default.

  To specify `port` only, use: `-b :7878`. This is equivalent to `-b localhost:7878`

  To specify `name` only, use: `-b somehost`. This is equivalent to `-b somehost:7676`.

`-c` *clientID*
: Specify the ID of the durable subscriber to a topic.

`-d` *topicName*
: Specify the name of the topic.

  Use this option with the `list dur` and `destroy dur` subcommands.

`-f`
: Perform action without user confirmation.

  Use this option with any subcommand.

`-h`
: Display usage help. Execute nothing else on the command line.

`-H`
: Display usage help, attribute list, and examples. Execute nothing else on the command line.

`-int` *interval*
: Specify the interval, in seconds, at which the `metrics bkr`, `metrics dst`, and `metrics svc` subcommands display metrics output.

  Use this option with the `metrics` subcommand.

`-javahome`
: Specify an alternate Java 2 compatible runtime to use.

`-m` *metricType*

Specify the type of metric information to diplay.

Use this option with the `metrics bkr`, `metrics dst`, and `metrics svc` subcommands. The value of *metricType* depends on whether the metrics are generated for a destination, a service, or a broker.

Use one of the following values to specify *metricType*:

`ttl` Total of messages in and out of the broker (default)

`rts` Provides the same information as `ttl`, but specifies the number of messages per second

`cxn` Connections, virtual memory heap, threads

The following command displays connection, VM heap, and threads metric information for the default broker instance (`localhost:7676`) every five seconds:

```
imqcmd metrics bkr -m cxn -int 5
```

`-msp` *numSamples*

Specify the number of samples the `metrics bkr`, `metrics dst`, and `metrics svc` subcommands display in the metrics output.

`-n` *argumentName*

Specify the name of the subcommand argument. Depending on the subcommand, this might be the name of a service, a physical destination, a durable subscription, or a transaction ID.

`-o` *attribute=value*

Specify the value of an attribute. Depending on the subcommand argument, this might be the attribute of a broker, service, or destination.

`-p` *password*

Specify the administrator password.

This option is deprecated. Use the `-passfile` option instead.

`-passfile`

Specify the administrator password.

`-pst` *pauseType*

Specify whether producers, consumers, or both are paused when pausing a destination.

Use this option with the `pause dst` subcommand. Use one of the following values:

CONSUMERS Pause delivery of messages to consumers.

PRODUCERS Pause delivery of messages from producers.

ALL Pause delivery of messages to consumers and from producers.

If the `-pst` option is not specified, pauses both consumers and producers (the equivalent of `-pst ALL`).

`-rtm` *timeout*
: Specify the timeout period in seconds of an `imqcmd` subcommand. The default value is `10`.

`-rtr` *numRetries*
: Specify the number of retries attempted after an `imqcmd` subcommand times out.

  The default value is `5`.

`-s`
: Silent mode. No output is displayed.

  Use this option with any subcommand.

`-secure`
: Specify a secure administration connection to the broker instance. You must first configure the broker to enable a secure connection service.

  Use this option whenever you want a secure communication with the broker.

`-svn` *serviceName*
: Specify the service for which the connections are listed.

  Use this option with the `list cxn` subcommand.

`-t` *destinationType*
: Specify the type of a destination: `t` (topic) or `q` (queue).

`-tmp`
: Include temporary destinations when listing destinations using the list `dst` subcommand.

`-u` *name*
: Specify the administrator user name.

  If you omit this value, you are prompted for it.

`-v`
: Display version information. Execute nothing else on the command line.

Usage

Subcommands and Options

The following subcommands and associated arguments and options are supported:

compact dst [`-t` *type* `-n` *destName*]
: Compact the flat-file data store for the destination of the specified type and name. If no type and name are specified, all destinations are compacted. Destinations must be paused before they can be compacted.

commit txn `-n` *transaction_id*
: Commit the specified transaction

create dst `-t` *destinationType* `-n` *destName* [`-o` *attribute=value*] [`-o` *attribute=value1*]...
Create a destination of the specified type, with the specified name, and the specified attributes. Destination names must contain only alphanumeric characters (no spaces) and can begin with an alphabetic character or the underscore character (_).

destroy dst `-t` *destinationType* `-n` *destName*
Destroy the destination of the specified type and name.

destroy dur `-n` *subscrName* `-c` *client_id*
Destroy the specified durable subscription for the specified Client Identifier.

list cxn [`-svn` *serviceName*] [`-b` *hostName:port*]
List all connections of the specified service name on the default broker or on a broker at the specified host and port. If the service name is not specified, all connections are listed.

list dst [`-tmp`]
List all destinations, with option of listing temporary destinations as well .

list dur `-d` *destination*
List all durable subscriptions for the specified destination.

list svc
List all connection services on the broker instance.

list txn
List all transactions, being tracked by the broker.

metrics bkr [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*]
Display broker metrics for the broker instance.

Use the `-m` option to specify the type of metric to display. Use one of the following values to specify *metricType*:

`ttl` Specifies the total of messages in and out of the broker (default) .

`rts` Provides the same information as `ttl`, but specifies the number of messages per second.

`cxn` Connections, virtual memory heap, threads.

Use the `-int` option to specify the interval (in seconds) at which to display the metrics. The default is 5 seconds.

Use the `-msp` option to specify the number of samples displayed in the output. A value of -1 means an unlimited number. The default value is -1.

metrics dst `-t` *type* `-n` *destName* [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*]
Displays metrics information for the destination of the specified type and name.

Use the `-m` option to specify the type of metrics to display. Use one of the following values to specify *metricType*:

`ttl` Specifies the number of messages flowing in and out of the broker and residing in memory.

`rts` Provides the same information as `ttl`, but specifies the number of messages per second.

`con` Displays consumer related metrics.

`dsk` Displays disk usage metrics.

Use the `-int` option to specify the interval (in seconds) at which to display the metrics. The default is 5 seconds.

Use the `-msp` option to specify the number of samples displayed in the output. A value of -1 means an unlimited number. The default value is 5.

metrics svc `-n` *serviceName* [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*]
List metrics for the specified service on the broker instance. Use the `-m` option to specify the type of metric to display. Use one of the following values to specify *metricType*:

`ttl` Total of messages in and out of the broker (default)

`rts` Provides the same information as `ttl`, but specifies the number of messages per second

`cxn` Connections, virtual memory heap, threads

Use the `-int` option to specify the interval (in seconds) at which to display the metrics. The default is 5 seconds.

Use the `-msp` option to specify the number of samples displayed in the output. A value of -1 means an unlimited number. The default value is -1.

pause bkr
Pause the broker instance.

pause dst [`-t` *type* `-n` *destName*] [`-pst` *pauseType*]
Pause the delivery of messages to consumers (`-pst CONSUMERS`), or from producers (`-pst PRODUCERS`), or both (`-pst ALL`), for the destination of the specified type and name. If no destination type or name are specified, all destinations are paused.

pause svc `-n` *serviceName*
Pause the specified service running on the broker instance. You cannot pause the administative service.

purge dst `-t` *destinationType* `-n` *destName*
Purge messages at the destination with the specified type and name.

purge dur `-n` *subscrName* `-c` *client_id*
Purge all messages for the specified client identifier.

query bkr
List the current settings of properties of the broker instance. Show the list of running brokers (in a multi-broker cluster) that are connected to the specified broker.

query dst `-t` *destinationType* `-n` *destName*
List information about the destination of the specified type and name.

query svc `-n` *serviceName*
Display information about the specified service running on the broker instance.

query txn `-n` *transaction_id*
List information about the specified transaction.

reload cls
Forces all the brokers in a cluster to reload the `imq.cluster.brokerlist` property and update cluster information. This subcommand only applies to broker clusters.

restart bkr
Shut down and restart the broker instance. This command restarts the broker using the options specified when the broker was first started. If you want different options to be in effect, you must shut down the broker and then start it again, specifying the options you want.

resume bkr
Resume the broker instance.

resume dst [`-t` *type*] [`-n -destName`]
Resumes the delivery of messages for the paused destination of the specified type and name. If no destination type and name are specified, all destinations are resumed.

resume svc `-n` *serviceName*
Resume the specified service running on the broker instance.

rollback txn `-n` *transaction_id*
Roll back the specified transaction.

shutdown bkr
Shut down the broker instance

update bkr `-o` *attribute=value* [`-o` *attribute=value*]...
Change the specified attributes for the broker instance.

update dst `-t` *destinationType* `-n` *destName* `-o` *attribute=value* [`-o` *attribute=value1*]...
Update the value of the specified attributes at the specified destination..

update svc `-n` *serviceName* `-o` *attribute=value* [`-o` *attribute=value1*]...
Update the specified attribute of the specified service running on the broker instance.

Attribute Value Pairs

You can specify attributes with the create and update subcommands. Applicable attributes depend on the subcommand arguments.

The following attributes are supported:

Queue (dst):

maxTotalMsgBytes — Value: Integer (maximum total size of messages, in bytes)

Default: `0` (unlimited)

maxBytesPerMsg — Value: Integer (maximum size of a single message, in bytes)

Default: `0` (unlimited)

maxNumMsgs — Value: Integer (maximum total number of messages)

Default: `0` (unlimited)

consumerFlowLimit — Value: Integer Initial number of queued messages sent to active consumers before load-balancing starts A value of `-1` means an unlimited number.

Default: `1000`

isLocalOnly — Value: Boolean (destination limited to delivering messages to local consumers only) Default: `false`

limitBehavior — Value: Specify how broker responds when memory-limit is reached. Use one of the following values:

FLOW_CONTROL	Slows down producers
REMOVE_OLDEST	Purges oldest messages
REJECT_NEWEST	Rejects the newest messages

Default: REJECT_NEWEST

localDeliveryPreferred — Value: Boolean Specify messages be delivered to remote consumers only if there are no consumers on the local broker. Requires that the destination not be restricted to local-only delivery (`isLocalOnly = false`)

Default: `false`

maxNumActiveConsumers — Value: Integer (maximum number of active consumers in load-balanced delivery) A value of `-1` means an unlimited number.

Default: `1`

maxNumBackupConsumers — Value: Integer (maximum number of backup consumers in load-balanced delivery) A value of `-1` means an unlimited number.

Default: `0`

maxNumProducers — Value: (maximum total number of producers) A value of `-1` means an unlimited number.

Default: `-1`

useDMQ — Specify whether a destination's dead messages are discarded or put on the dead message queue.

Default: `true`

Topic (dst):

consumerFlowLimit — Value: Integer Maximum number of messages delivered to a consumer in a single batch. A value of `-1` means an unlimited number.

Default: `1000`

isLocalOnly — Value: Boolean (destination limited to delivering messages to local consumers only)

Default: `false`

limitBehavior — Value: Specify how broker responds when memory-limit is reached. Use one of the following values:

FLOW_CONTROL	Slows down producers
REMOVE_OLDEST	Purges the oldest messages
REJECT_NEWEST	Rejects the newest messages

Default: REJECT_NEWEST

maxBytesPerMsg — Value: Integer (maximum size of a single message, in bytes)

Default: `0` (unlimited)

maxNumMsgs — Value: Integer (maximum total number of messages) A value of `-1` means an unlimited number.

Default: `-1`

maxNumProducers — Value: (maximum total number of producers)

Default: `0` (unlimited)

maxTotalMsgBytes — Value: Integer (maximum total size of messages, in bytes) A value of `-1` means an unlimited number.

Default: `-1`

useDMQ	Specify whether a destination's dead messages are discarded or put on the dead message queue. Default: `true`
Broker (bkr):	
imq.autocreate.destination.useDMQ	Value: Boolean. Set the `useDMQ` attribute to `true` to enable all autocreated physical destinations on a broker to use the dead message queue. Set the `useDMQ` attribute to `false` to disable all autocreated physical destinations on a broker from using the dead message queue. Default: `true`
imq.autocreate.queue	Value: Boolean Default: `true`
imq.autocreate.queue.maxNumActiveConsumers	Value: Integer (maximum number of consumers that can be active in load-balanced delivery from an autocreated queue destination) A value of `-1` means an unlimited number. Default: `1`
imq.autocreate.queue.maxNumBackupConsumers	Value: Integer (maximum number of backup consumers that can take the place of active consumers) A value of `-1` means an unlimited number. Default: `0`
imq.autocreate.topic	Value: Boolean Default: `true`
imq.cluster.url	Value: String (location of cluster configuration file) Default: none
imq.log.file.rolloverbytes	Value: Integer (maximum size of a log file, in bytes)

	Default: `0` (no rollover based on size)
imq.log.file.rolloversecs	Value: Integer (maximum age of a log file, in seconds)
	Default: `0` (no rollover based on age)
imq.log.level	Value: String (`NONE`, `ERROR`, `WARNING`, `INFO`)
	Default: `INFO`
imq.message.max_size	Value: Integer (maximum size of a single message, in bytes)
	Default: `70m`
imq.portmapper.port	Value: Integer
	Default: `7676`
imq.system.max_count	Value: Integer (maximum total number of messages)
	Default: `0` (no limit)
imq.system.max_size	Value: Integer (maximum total size of messages, in bytes)
	Default: `0` (no limit)

Service (svc):

maxThreads	Value: Integer (maximum threads assigned)
	Default: Depends on service
minThreads	Value: Integer (minimum threads assigned)
	Default: Depends on service
port	Value: Integer
	Default: `0` (dynamically allocated)

Examples

EXAMPLE 1 Shutting Down a Broker

The following command shuts down a broker for hostname `myserver` on port `7676`:

```
mqcmd shutdown bkr -b myserver:7676
```

EXAMPLE 2 Restarting a Broker

The following command restarts a broker for hostname `myserver`:

```
imqcmd restart bkr -b myserver
```

EXAMPLE 3 Pausing a Service

The following command pauses a broker for hostname `localhost` on port `7676`, with a *serviceName* of `jms`:

```
imqcmd pause svc -n jms -b :7676
```

EXAMPLE 4 Resuming a Service

The following command resumes a service for hostname `localhost` on port `7676`, with a *serviceName* of `jms`:

```
imqcmd resume svc -n jms -b myserver:7676
```

EXAMPLE 5 Creating a Queue Destination

The following command creates a queue destination for hostname `myserver` on port `7676`, with a *destName* of `myFQ`, a *queueDeliveryPolicy* of `Failover`, and a *maxBytesPerMsg* of 10000:

```
imqcmd create dst -n myFQ -t q -o "queueDeliveryPolicy=f" \
     -o "maxBytesPerMsg=10000" -b myserver:7676
```

EXAMPLE 6 Purging a Queue Destination

The following command purges a queue destination for hostname `myserver` on port `7676`, with a *destName* of `myFQ`:

```
imqcmd purge dst -n myFQ -t q -b myserver:7676
```

EXAMPLE 7 Listing Destinations on a Broker

The following command lists destinations for hostname `myserver` on port `7676`:

```
imqcmd list dst -b myserver:7676
```

EXAMPLE 8 Updating a Portmapper Port

The following command updates a portmapper port on hostname `myserver` from port `7676` to `7878`:

```
imqcmd update bkr -o "imq.portmapper.port=7878"
```

EXAMPLE 9 Updating the Maximum Number of Messages in the Queue

The following command updates the maximum number of messages in the queue to `2000` for `myserver` on port `8080` with a *destName* of `TestQueue`:

```
imqcmd update dst -b myserver:8080 -n TestQueue -t q -o "maxNumMsgs=2000"
```

EXAMPLE 10 Updating the Maximum Threads

The following command updates the maximum threads `jms` connection service to `200` for hostname `localhost` on port `7676`:

```
imqcmd update svc -n jms -o "minThreads=200"
```

EXAMPLE 11 Listing Durable Subscriptions

The following command lists durable subscriptions for a topic with hostname `localhost` on port `7676` with a *destName* of `myTopic`:

```
imqcmd list dur -d myTopic
```

EXAMPLE 12 Destroying Durable Subscriptions

The following command destroys subscriptions for hostname `localhost` on port `7676` with a *dursubName* of `myDurSub` and a *client_ID* of `111.222.333.444`:

```
imqcmd destroy dur -n myDurSub -c "111.222.333.444"
```

EXAMPLE 13 Listing All Transactions

The following command lists all transactions on a broker with hostname `localhost` on port `7676`:

```
imqcmd list txn
```

EXAMPLE 14 Displaying Information About a Transaction

The following command displays information about a transaction with hostname `localhost` on port `7676`, and a *transactionID* of `1234567890`

```
imqcmd query txn -n 1234567890
```

EXAMPLE 15 Committing a Transaction

The following command commits a transaction with hostname `localhost` on port `7676`, and a *transactionID* of `1234567890`:

```
imqcmd commit txn -n 1234567890
```

Environment Variables

The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` Specify the Java 2 compatible runtime. When this environment variable is not set it defaults to `/usr/j2se`.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiqu

See Also imqadmin(1M), imqbrokerd(1M), imqdbmgr(1M), imqkeytool(1M), imqobjmgr(1M), imqusermgr(1M), attributes(5)

Sun Java System Message Queue Administrator's Guide

Name imqdbmgr – manage a plugged-in JDBC-compliant Message Queue data store

Synopsis

```
/usr/bin/imqdbmgr subcommand argument [ [option...]]

/usr/bin/imqdbmgr -h | -help

/usr/bin/imqdbmgr -v | -version
```

Description The `imqdbmgr` utility creates and manages a Java DataBase Connectivity (JDBC) compliant database used for Message Queue persistent storage.

The database can be either embedded or external. To use a JDBC-compliant database (and the `imdbmgr` utility), you need to first set a number of JDBC-related properties in the broker instance configuration file. See the *Sun Java System Message Queue Administrator's Guide* for additional information.

`imqdbmgr` supports four management subcommands. These *subcommand*s, and their corresponding *argument*s and *option*s follow the `imqdbmgr` command on the command line. See `USAGE` and `OPTIONS`.

The following subcommands are supported:

create	Create a Message Queue database schema.
delete	Delete Message Queue database tables in the current data store.
recreate	Delete Message Queue database tables and recreate Message Queue database schema in the curent data store.
reset	Reset the database table lock to allow other processes to access database tables.

The `imqdbmgr` subcommands support the following arguments:

all	Indicates the subcommand applies to the data store, as well as the database tables.
lck	Indicates the subcommand applies to the database table lock.
oldtbl	Indicates the subcommand applies to an older version of the database tables.
tbl	Indicates the subcommand applies to the database tables only.

Options The following options are supported:

`-b` *brokerName*	Specify the broker instance name and corresponding instance configuration properties. If this option is not specified, the default broker instance is assumed. Use this option with the `create`, `delete`, `recreate`, or `reset` subcommands.
`-D` *property=value*	Set system property *property* to *value*.

Use this option with the `create`, `delete`, `recreate`, or `reset` subcommands.

`-h` | `-help` — Display usage help. Execute nothing else on the command line.

`-p` *password* — Specify the database password.

Use this option with the `create`, `delete`, `recreate`, or `reset` subcommands.

`-u` *userName* — Specify the database user name.

Use this option with the `create`, `delete`, `recreate`, or `reset` subcommands.

`-v` | `-version` — Display version information. Execute nothing else on the command line.

Usage The following subcommands and associated arguments are supported:

create all	Create a new embedded data store and Message Queue database schema for a specified or default broker instance.
create tbl [`-u` *userName*] [`-p` *password*]	Create Message Queue database schema in an existing data store for a specified or default broker instance.
delete tbl [`-u` *userName*] [`-p` *password*]	Delete Message Queue database tables in the current data store for a specified or default broker instance.
delete oldtbl [`-u` *userName*] [`-p` *password*]	Delete the earlier version of Message Queue database tables. Used after the data store has been automatically migrated to the current version of Message Queue.
recreate tbl [`-u` *userName*] [`-p` *password*]	Delete Message Queue database tables and recreate Message Queue database schema in the current data store for a specified or default broker instance.
reset lck	Reset the database table lock to allow other processes to access database tables.

Environment Variables The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` — Specify the Java 2 compatible runtime. When this environment variable is not set it defaults to `/usr/j2se`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/var/imq/instances/`*brokerName*`/dbstore`	Recommended directory in which to create an embedded database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiqu

See Also `imqadmin(1M)`, `imqbrokerd(1M)`, `imqcmd(1M)`, `imqusermgr(1M)`, `imqkeytool(1M)`, `imqobjmgr(1M)`, `attributes(5)`

Sun Java System Message Queue Administrator's Guide

Name imqkeytool – generate a self-signed certificate for secure communication

Synopsis

```
/usr/bin/imqkeytool [-broker] [-servlet keystore_location]

/usr/bin/imqkeytool -h
```

Description The `imqkeytool` utility generates a self-signed certificate for secure communication. The certificate can be used by a broker instance to establish a secure connection with a client, or by a Message Queue-supplied HTTPS servlet to establish a secure connection with a broker instance. An HTTPS servlet is an SSL-enabled varient of the HyperText Transfer Protocol that establishes a secure connection with a broker instance.

Without an option, `imqkeytool` generates a self-signed certificate for a broker instance.

`imqkeytool` uses command line options to specify whether the certificate is used by a broker instance or by a servlet.

Options The following options are supported:

`-broker` Generate a self-signed certificate for the broker and places it in the Message Queue keystore. All broker instances running on a system must use the same certificate.

`-h` Display usage help. Do not execute anything else on the command line.

`-servlet` *keystore_location* Generate a self-signed certificate for an HTTPS servlet and places it in *keystore_location*.

keystore_location refers to the location of the keystore. You should move this keystore to a location where it is accessible and readable by the Message Queue HTTPS servlet to establish a secure connection with a broker.

Environment Variables The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` Specify the Java 2 compatible runtime. When this environment variable is not set it defaults to `/usr/j2se`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/etc/imq/keystore` Contains Message Queue keystore in which `imqkeytool` stores a self-signed certificate for brokers.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiqu

See Also imqadmin(1M), imqbrokerd(1M), imqcmd(1M), imqdbmgr(1M), imqobjmgr(1M), imqusermgr(1M), attributes(5)

Sun Java System Message Queue Administrator's Guide

Name imqobjmgr – manage Message Queue administered objects

Synopsis

```
/usr/bin/imqobjmgr subcommand [[option]...]

/usr/bin/imqobjmgr -i fileName

/usr/bin/imqobjmgr -h | [-H] | -help | -Help

/usr/bin/imqobjmgr -v
```

Description `imqobjmgr` manages Message Queue administered objects in an object store accessible using JNDI. Administered objects allow JMS clients to be provider-independent by insulating them from provider-specific naming and configuration formats.

`imqobjmgr` supports five management subcommands. These *subcommand*s, and their corresponding *option*s follow the `imqobjmgr` command on the command line. See `USAGE` and `OPTIONS`.

The following subcommands are supported:

add	Add a new administered object
delete	Delete an administered object
list	Display a list of administered objects
query	Display information about administered objects
update	Update administered objects

You can use the `-i` option to specify the name of an input file that uses java property file syntax to represent all or part of any `imqobjmgr` subcommand clause. The `-f`, `-s`, and `-pre` options can be used with any `imqobjmgr` subcommand.

Options The following options are supported:

`-f`	Perform action without user confirmation.
`-h` | `-help`	Display usage help. Execute nothing else on the command line.
`-H` | `-Help`	Display usage help, attribute list, and examples. Execute nothing else on the command line.
`-i` *fileName*	Specify the name of an input file containing all or part of the subcommand clause, specifying object type, lookup name, object attributes, object store attributes, or other options. Use this option for repetitive information, such as object store attributes.
`-j` *attribute=value*	Specify attributes necessary to identify and access a JNDI object store.
`-javahome`	Specify an alternate Java 2 compatible runtime to use. `imqobjmgr` uses the runtime bundled with the operating system by default.

`-l` *lookupName*	Specify the JNDI lookup name of an administered object. This name must be unique in the object store's context.
`-o` *attribute=value*	Specify the attributes of an administered object.
`-pre`	Run command in preview mode. Preview mode indicates what will be done without performing the command.
`-r` *read-only_state*	Specify if an administered object is a read-only object. A value of true indicates the administered object is a read-only object. JMS clients cannot modify the attributes of read-only administered objects. The read-only state is set to false by default.
`-s`	Silent mode. No output is displayed.

`-t` *type*	Specify the type of an administered object:
	`q = queue`
	`t = topic`
	`cf = ConnectionFactory`
	`qf = queueConnectionFactory`
	`tf = topicConnectionFactory`
	`xcf = XA ConnectionFactory (distributed transactions)`
	`xqf = XA queueConnectionFactory (distributed transactions)`
	`xtf = XA topicConnectionFactory (distributed transactions)`
	`e = SOAP endpoint (used to support SOAP messaging)`

`-v`	Display version information. Execute nothing else on the command line.

Usage

This section provides information on subcommands, options, and attribute value pairs.

Subcommands and Options

The following subcommands and corresponding options are supported:

add `-t` *type* `-l` *lookupName* [`-o` *attribute=value*]... `- j` *attribute=value...*
: Add a new administered object of the specified type, lookup name , and object attributes to an object store.

delete `-t` *type* `-l` *lookupName* `-j` *attribute=value...*
: Delete an administered object, of the specified type and lookup name from an object store.

list [`-t` *type*] `-j` *attribute=value...*
: Display a list of administered objects of a specified type, or all administered objects, in an object store.

query `-l` *lookupName* `-j` *attribute=value...*
: Display information about an administered object of a specified lookup name in an object store.

update `-l` *lookupName* [`-o` *attribute=value*]... `-j` *attribute=value*...
Update the specified attribute values of an administered object of the specified lookup name in an object store.

Attribute Value Pairs

The following attribute value pairs are supported for the specified administered object types:

Type = ConnectionFactories: ConnectionFactory, TopicConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XATopicConnectionFactory, and XAQueueConnectionFactory

imqAckOnAcknowledge
Value: String (`true`, `false`, not specified)

Default: not specified

imqAckOnProduce
Value: String (`true`, `false`, not specified)

Default: not specified

imqAckTimeout
Value: String (time in milliseconds)

Default: `0` (no timeout)

imqAddressList
Value: String

Default: not specified

imqAddressListBehavior
Value: String

Default: Priority

imqAddressListIterations
Value: Integer

Default: 1

imqBrokerHostName
Used if imqConnectionType is `TCP` or `TLS`. This attribute type is only supported in Message Queue 3.0.

Value: String

Default:`localhost`

imqBrokerHostPort
Used if imqConnectionType is `TCP` or `TLS`. This attribute type is only supported in Message Queue 3.0.

Value: Integer

Default: `7676`

imqBrokerServicePort
Used if imqConnectionType is `TCP` or `TLS`. This attribute type is only supported in Message Queue 3.0.

Value: Integer

Default: `0`

imqConfiguredClientID
Value: String (ID number)

Default: no ID specified

imqConnectionFlowCount
Value: Integer

Default: 100

imqConnectionFlowLimit
Value: Integer

Default: 1000

imqConnectionFlowLimitEnabled
Value: Boolean

Default: `false`

imqConnectionType
This attribute type is only supported in Message Queue 3.0.

Value: String (`TCP`, `TLS`, `HTTP`).

Default: `TCP`

imqConnectionURL
Used if imqConnectionType is `HTTP`. This attribute type is only supported in Message Queue 3.0.

Value: String

Default: `http://`*localhost*`/imq/tunnel`

imqConsumerFlowLimit
Value: Integer

Default: 1000

imqConsumerFlowThreshold
Value: Integer

Default: 50

imqDefaultPassword
Value: String

Default: `guest`

imqDefaultUsername
Value: String

Default: `guest`

imqDisableSetClientID
Value: Boolean

Default: `false`

imqJMSDeliveryMode
Value: Integer (`1`=non-persistent, `2`=persistent)

Default: `2`

imqJMSExpiration
Value: Long (time in milliseconds)

Default: `0` (does not expire)

imqJMSPriority
Value: Integer (`0` to `9`)

Default: `4`

imqLoadMaxToServerSession
Value: Boolean

Default: `true`

imqOverrideJMSDeliveryMode
Value: Boolean

Default: `false`

imqOverrideJMSExpiration
Value: Boolean

Default: `false`

imqOverrideJMSHeadersToTemporaryDestinations
Value: Boolean

Default: `false`

imqOverrideJMSPriority
Value: Boolean

Default: `false`

imqQueueBrowserMaxMessagesPerRetrieve
Value: Integer

Default: `1000`

imqBrowserRetrieveTimeout
Value: Long (time in milliseconds)

Default: `60,000`

imqReconnectAttempts
Value: Integer

Default: `0`

imqReconnectEnabled
Value: Boolean

Default: `false`

imqReconnectInterval
Value: Long (time in milliseconds)

Default: `3000`

imqSetJMSXAppID
Value: Boolean

Default: `false`

imqSetJMSXConsumerTXID
Value: Boolean

Default: `false`

imqSetJMSXProducerTXID
Value: Boolean

Default: `false`

imqSetJMSXRcvTimestamp
Value: Boolean

Default: `false`

imqSetJMSXUserID
Value: Boolean

Default: `false`

imqSSLIsHostTrusted
Used if imqConnectionType is `TLS`. This attribute type is only supported in Message Queue 3.0.

Value: Boolean

Default: `true`

Type = Destinations: Topic and Queue

imqDestinationDescription — Value: String

Default: no description

imqDestinationName — Value: String

Default: `Untitled_Destination_Object`

Type = Endpoint (SOAP Endpoint)

imqEndpointDescription — Value: String

Default: `A description for the endpoint object`

imqEndpointName — Value: String

Default: `Untitled_Endpoint_Object`

imqSOAPEndpointList — Value: String (one or more space-separated URLs)

Default: no url

Examples

EXAMPLE 1 Adding a Topic Administered Object to an Object Store

Where JNDI lookup name=`myTopic` and `imqDestinationName=MyTestTopic`, the following command adds to an LDAP server object store:

```
imqobjmgr add -t t -l "cn=myTopic"\
-o "imqDestinationName=MyTestTopic"\
-j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"\
-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
```

Where JNDI lookup name=`myTopic` and `imqDestinationName=MyTestTopic`, the following command adds to a file system object store:

```
imqobjmgr add -t -l "cn=myTopic"\
-o "imqDestinationName=MyTestTopic"\
-j \
  "java.naming.factory.initial=com.sun.jndi.fscontext.RefFSContextFactory"\
-j "java.naming.provider.url=file:/home/foo/imq_admin_objects"
```

EXAMPLE 1 Adding a Topic Administered Object to an Object Store *(Continued)*

Where JNDI lookup name=`myTopic` and `imqDestinationName=MyTestTopic`, the following command adds to a file system object store, using an input file:

```
imqobjmgr -i inputfile
```

The associated input file consists of the following:

```
cmdtype=add
obj.type=t
obj.lookupName=cn=myTopic
obj.attrs.imqDestinationName=MyTestTopic
objstore.attrs.java.naming.factory.initial=com.sun.jndi.fscontext.\
  RefFSContextFactory
objstore.attrs.java.naming.provider.url=file:/home/foo/imq_admin_objects
```

EXAMPLE 2 Adding a `QueueConnectionFactory` Administered Object to an Object Store

Where JNDI lookup name=`myQCF`, `read-only state=true`, `imqAddressList=mq://foohost:777/jms`, the following command adds to an LDAP server object store:

```
imqobjmgr add -t qf -l "cn=myQCF" -r true\

-o "imqAddressList=mq://foohost:777/jms"\
-j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"\
-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"\
```

Where JNDI lookup name=`myQCF`, `read-only state=true`, `imqAddressList=mq://foohost:777/jms`, the following command adds to an LDAP server object store using an input file:

```
imqobjmgr -i inputfile
```

The associated input file consists of the following:

```
cmdtype=add
obj.type=qf
obj.lookupName=cn=myQCF
obj.readOnly=true
obj.attrs.imqAddressList=mq://foohost:777/jms
objstore.attrs.java.naming.factory.initial=com.sun.jndi.\
  ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=ldap://mydomain.com:389/o=imq
```

Where JNDI lookup name=`myQCF`, read-only state=`true`, `imqAddressList=mq://foohost:777/jms`, the following command adds to an LDAP server object store, using both an input file and command options:

EXAMPLE 2 Adding a `QueueConnectionFactory` Administered Object to an Object Store *(Continued)*

```
imqobjmgr add -t qf -l "cn=myQCF"\
-o "imqAddressList=mq://foohost:777/jms"\
-i inputfile
```

The associated input file consists of the following:

```
objstore.attrs.java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=ldap://mydomain.com:389/o=imq
```

EXAMPLE 3 Deleting a Topic Administered Object from an Object Store

Where JNDI lookup name=`myTopic` and no confirmation is requested, the following command deletes from an LDAP server object store:

```
imqobjmgr delete -f -l "cn=myTopic"\
-j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"\
-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
```

EXAMPLE 4 Querying Information About a Topic Administered Object

Where JNDI lookup name=`myTopic`, the following command queries from an LDAP server object store using simple authentication scheme:

```
imqobjmgr query -l "cn=myTopic"\
-j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"\
-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"\
-j "java.naming.security.authentication=simple"\
-j "java.naming.security.principal=uid=foo,ou=imqobjmgr,o=imq"\
-j "java.naming.security.credentials=foo"
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiqu

See Also

`imqadmin(1M)`, `imqcmd(1M)`, `imqbrokerd(1M)`, `imqkeytool(1M)`, `imqusermgr(1M)`, `attributes(5)`

Sun Java System Message Queue Administrator's Guide

Name imqusermgr – command utility for managing a Message Queue user repository

Synopsis

```
/usr/bin/imqusermgr subcommand [[option]...]

/usr/bin/imqusermgr -h

/usr/bin/imqusermgr -v
```

Description The `imqusermgr` utility manages a file-based user repository to authenticate and authorize users of a Message Queue message server.

`imqusermgr` provides subcommands for adding, deleting, updating, and listing user entries in the repository.

`imqusermgr` supports four management subcommands. These *subcommand*s, and their corresponding *option*s follow the `imqusermgr` command on the command line. See `USAGE` and `OPTIONS`.

The following subcommands are supported:

add	Add a new user and associated password to the repository.
delete	Delete a user from the repository.
list	Display information users in the repository.
update	Update the password or state of a user in the repository.

Options The following options are supported:

`-a` *active_state*	Specify if user's state is active or inactive. An inactive user cannot create connections to the Message Queue message server. Valid values for *active_state* are `true` or `false`. Specify `true` for active or `false` for inactive. the default is `true`. Use this option with the `update` subcommmand.
`-f`	Perform action without user confirmation. Use this option with the `delete` and `update` subcommmands.
`-g` *group*	Specify the group of the user. Valid values for group are `admin`, `user`, and `anonymous`. Use this option with the `add` subcommmand.
`-h`	Display usage help. Exceute nothing else on the command line.
`-i` *brokerName*	Specify the broker instance user repository to which the command applied. If you do not specify *brokerName*, the default *brokerName* is assumed.

Use this option with the `add`, `delete`, `list`, and `update` subcommmands.

`-p` *password* Specify user password.

Use this option with the `add` and `update` subcommmands.

`-s` Silent mode. Display no output

Use this option with the `add`, `delete`, and `update` subcommands.

`-u` *userName* Specify user name.

userName cannot contain the following characters: asterisk (`*`), colon (`:`), NEWLINE, or RETURN.

Use this option with the `add`, `delete`, `update` and `list` subcommands.

`-v` Display version information. Exceute nothing else on the command line.

Usage The following subcommands and corresponding options are supported:

add `-u` *userName* `-p` *password* [`-g` *group*] [`-s`] [`-i` *brokerName*]
Add a new user and associated password to the repository, and optionally specify the user's group.

delete `-u` *userName* [`-s`] [`-f`] [`-i` *brokerName*]
Delete a user from the repository.

list [`-u` *user_name*] [`-i` *brokerName*]
Display information about the specified user in the repository. If no user is specified, all users are displayed.

update `-u` *userName* `-p` *password* [`-a` *state*] [`-s`] [`-f`] [`-i` *brokerName*]
update `-u` *userName* `-a` *state* [`-p` *password*] [`-s`] [`-f`] [`-i` *brokerName*]
Update the password or state (or both) of a user.

Environment Variables The following environment variables affect the execution of this command:

`IMQ_JAVAHOME` Specify the Java 2 compatible runtime. When this environment variable is not set, it defaults to `/usr/j2se`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/etc/imq/passwd` Flat-file user repository.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiqu

See Also imqadmin(1M), imqbrokerd(1M), imqcmd(1M), imqdbmgr(1M), imqkeytool(1M), imqobjmgr(1M), attributes(5)

Sun Java System Message Queue Administrator's Guide

Name in.chargend – UDP or TCP character generator service daemon

Synopsis

```
in.chargend

FMRI svc:/internet/chargen:default
```

Description FMRI stands for Fault Management Resource Identifier. It is used to identify resources managed by the Fault Manager. See `fmd(1M)` and `smf(5)`.

The `in.chargend` service provides the server-side of the character-generator protocol. This protocol is used for debugging and bandwidth measurement and is available on both TCP and UDP transports, through port 19.

The in.chargend service is an `inetd(1M)` `smf(5)` delegated service. The `in.chargend` detects which transport is requested by examining the socket it is passed by the `inetd` daemon.

TCP-based service — Once a connection is established, the `in.chargend` generates a stream of data. Any data received is discarded. The server generates data until the client program terminates the connection. Note that the data flow is limited by TCP flow control mechanisms.

UDP-based service — The `in.chargend` listens for UDP datagrams. When a datagram is received, the server generates a UDP datagram in response containing a random number of ASCII characters (ranging from 0 to 512 characters). Any received data is ignored.

The `in.chargend` data consists of a pattern of 72 character lines containing the printable, 7–bit ASCII characters. Each line is terminated with a carriage return and a line feed character.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcnsu
Interface Stability	Evolving

See Also `inetd(1M)`, `attributes(5)`, `smf(5)`

RFC 864

Name in.comsat, comsat – biff server

Synopsis

```
/usr/sbin/in.comsat
```

Description `comsat` is the server process which listens for reports of incoming mail and notifies users who have requested to be told when mail arrives. It is invoked as needed by `inetd(1M)`, and times out if inactive for a few minutes.

`comsat` listens on a datagram port associated with the `biff` service specification (see `services(4)`) for one line messages of the form

user@*mailbox*-*offset*

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a `biff y`), the *offset* is used as a seek offset into the appropriate mailbox file, and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the `From`, `To`, `Date`, or `Subject` lines are not printed when displaying the message.

Files `/var/adm/utmpx` user access and administration information

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`,`services(4)`, `attributes(5)`, `smf(5)`

Notes The message header filtering is prone to error.

The `in.comsat` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/comsat:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name in.daytimed – UDP or TCP daytime protocol service daemon

Synopsis

```
in.daytimed

FMRI svc:/internet/daytime:default
```

Description FMRI stands for Fault Management Resource Identifier. It is used to identify resources managed by the Fault Manager. See `fmd(1M)` and `smf(5)`.

The `in.daytimed` service provides the server-side of the daytime protocol. This protocol is used for debugging and bandwidth measurement and is available on both TCP and UDP transports, through port 13.

The `in.daytimed` service is an `inetd(1M)` `smf(5)` delegated service. The `in.daytimed` detects which transport is requested by examining the socket it is passed by the `inetd` daemon.

TCP-based service — Once a connection is established, the `in.daytimed` generates the current date and time in `ctime(3C)` format as 7-bit ASCII and sends it through the connection. The server then closes the connection. Any data received from the client side of the connection is discarded.

UDP-based service — The `in.daytimed` listens for UDP datagrams. When a datagram is received, the server generates the current date and time in `ctime(3C)` format as 7-bit ASCII and inserts it in a UDP datagram sent in response to the client's request. Any received data is ignored.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcnsu
Interface Stability	Evolving

See Also `inetd(1M)`, `attributes(5)`, `smf(5)`

RFC 867

Name in.dhcpd – Dynamic Host Configuration Protocol server

Synopsis

```
/usr/lib/inet/in.dhcpd  [-denv] [-h relay_hops] [-i interface, ...]
 [-l syslog_local_facility] [-b automatic | manual]
 [-o DHCP_offer_time] [-t dhcptab_rescan_interval]

/usr/lib/inet/in.dhcpd  [-dv] [-h relay_hops] [-i interface,]...
 [-l syslog_local_facility] -r IP_address | hostname, ...
```

Description `in.dhcpd` is a daemon that responds to Dynamic Host Configuration Protocol (DHCP) requests and optionally to BOOTP protocol requests. The daemon forks a copy of itself that runs as a background process. It must be run as root. The daemon has two run modes, DHCP server (with optional BOOTP compatibility mode) and BOOTP relay agent mode.

The first line in the `SYNOPSIS` section illustrates the options available in the DHCP/BOOTP server mode. The second line in the SYNOPSIS section illustrates the options available when the daemon is run in BOOTP relay agent mode.

The DHCP and BOOTP protocols are used to provide configuration parameters to Internet hosts. Client machines are allocated their IP addresses as well as other host configuration parameters through this mechanism.

The DHCP/BOOTP daemon manages two types of DHCP data tables: the `dhcptab` configuration table and the DHCP network tables.

See `dhcptab(4)` regarding the dhcptab configuration table and `dhcp_network(4)` regarding the DHCP network tables.

The `dhcptab` contains macro definitions defined using a `termcap`-like syntax which permits network administrators to define groups of DHCP configuration parameters to be returned to clients. However, a DHCP/BOOTP server always returns hostname, network broadcast address, network subnet mask, and IP maximum transfer unit (MTU) if requested by a client attached to the same network as the server machine. If those options have not been explicitly configured in the `dhcptab`, `in.dhcpd` returns reasonable default values.

The `dhcptab` is read at startup, upon receipt of a `SIGHUP` signal, or periodically as specified by the `-t` option. A `SIGHUP` (sent using the command `svcadm refresh network/dhcp-server`) causes the DHCP/BOOTP daemon to reread the `dhcptab` within an interval from `0-60` seconds (depending on where the DHCP daemon is in its polling cycle). For busy servers, users should run `svcadm restart network/dhcp-server` to force the `dhcptab` to be reread.

The DHCP network tables contain mappings of client identifiers to IP addresses. These tables are named after the network they support and the datastore used to maintain them.

The DHCP network tables are consulted during runtime. A client request received from a network for which no DHCP network table exists is ignored.

This command may change in future releases of Solaris software. Scripts, programs, or procedures that use this command might need modification when upgrading to future Solaris software releases.The command line options provided with the `in.dhcpd` daemon are used

only for the current session, and include only some of the server options you can set. The dhcpsvc.conf(4) contains all the server default settings, and can be modified by using the dhcpmgr utility. See dhcpsvc.conf(4) for more details.

Options The following options are supported:

`-b automatic | manual` This option enables BOOTP compatibility mode, allowing the DHCP server to respond to BOOTP clients. The option argument specifies whether the DHCP server should automatically allocate permanent lease IP addresses to requesting BOOTP clients if the clients are not registered in the DHCP network tables (`automatic`) or respond only to BOOTP clients who have been manually registered in the DHCP network tables (`manual`). This option only affects DHCP server mode.

`-d` Debugging mode. The daemon remains as a foreground process, and displays verbose messages as it processes DHCP and/or BOOTP datagrams. Messages are displayed on the current TTY. This option can be used in both DHCP/BOOTP server mode and BOOTP relay agent mode.

`-h` *relay_hops* Specifies the maximum number of relay agent hops that can occur before the daemon drops the DHCP/BOOTP datagram. The default number of relay agent hops is 4. This option affects both DHCP/BOOTP server mode and BOOTP relay agent mode.

`-i` *interface, . . .* Selects the network interfaces that the daemon should monitor for DHCP/BOOTP datagrams. The daemon ignores DHCP/BOOTP datagrams on network interfaces not specified in this list. This option is only useful on machines that have multiple network interfaces. If this option is not specified, then the daemon listens for DHCP/BOOTP datagrams on all network interfaces. The option argument consists of a comma-separated list of interface names. It affects both DHCP/BOOTP server and BOOTP relay agent run modes.

`-l` *syslog_local_facility* The presence of this option turns on transaction logging for the DHCP server or BOOTP relay agent. The value specifies the `syslog` local facility (an integer from `0` to `7` inclusive) the DHCP daemon should use for tagging the transactions. Using a facility separate from the `LOG_DAEMON` facility allows the network administrator to capture these transactions separately from other DHCP daemon events for such

purposes as generating transaction reports. See syslog(3C), for details about local facilities. Transactions are logged using a record with 9 space-separated fields as follows:

1. Protocol:

```
Relay mode:     "BOOTP"
Server mode:    "BOOTP" or "DHCP" based upon client
                     type.
```

2. Type:

```
Relay mode:     "RELAY-CLNT", "RELAY-SRVR"
Server mode:    "ASSIGN", "EXTEND", "RELEASE",
                    "DECLINE", "INFORM", "NAK" "ICMP-ECHO."
```

3. Transaction time: absolute time in seconds (unix time)
4. Lease time:

```
Relay mode:     Always 0.
Server mode:    0 for ICMP-ECHO events, absolute time in
                    seconds (unix time)  otherwise
```

5. Source IP address: Dotted Internet form

   Relay mode: Relay interface IP on RELAY-CLNT, INADDR_ANY on RELAY-SRVR.
   Server mode: Client IP.

6. Destination IP address: Dotted Internet form

   Relay mode: Client IP on RELAY-CLNT, Server IP on RELAY-SRVR.
   Server mode: Server IP.

7. Client Identifier: Hex representation (0-9, A-F)

   Relay mode: MAC address
   Server mode: BOOTP - MAC address; DHCP - client id

8. Vendor Class identifier (white space converted to periods (.)).

   Relay mode: Always "N/A"
   Server mode: Vendor class ID tokenized by converting white space characters to periods (.)

9. MAC address: Hex representation (0-9, A-F)

Relay mode: MAC address
Server mode: MAC address

The format of this record is subject to change between releases.

Transactions are logged to the console if daemon is in debug mode (`-d`).

Logging transactions impact daemon performance.

It is suggested that you periodically rotate the DHCP transaction log file to keep it from growing until it fills the filesystem. This can be done in a fashion similar to that used for the general system message log `/var/adm/messages` and is best accomplished using the facilities provided by `logadm(1M)`.

`-n` Disable automatic duplicate IP address detection. When this option is specified, the DHCP server does not attempt to verify that an IP address it is about to offer a client is not in use. By default, the DHCP server pings an IP address before offering it to a DHCP/BOOTP client, to verify that the address is not in use by another machine.

`-o` *DHCP_offer_time* Specifies the number of seconds the DHCP server should cache the offers it has extended to discovering DHCP clients. The default setting is `10` seconds. On slow network media, this value can be increased to compensate for slow network performance. This option affects only DHCP server mode.

`-r` *IP_address* | *hostname, . . .* This option enables BOOTP relay agent mode. The option argument specifies a comma-separated list of IP addresses or hostnames of DHCP or BOOTP servers to which the relay agent is to forward BOOTP requests. When the daemon is started in this mode, any DHCP tables are ignored, and the daemon simply acts as a BOOTP relay agent.

A BOOTP relay agent listens to UDP port 68, and forwards BOOTP request packets received on this port to the destinations specified on the command line. It supports the `BROADCAST` flag described in RFC 1542. A BOOTP relay

agent can run on any machine that has knowledge of local routers, and thus does not have to be an Internet gateway machine.

Note that the proper entries must be made to the `netmasks` database so that the DHCP server being served by the BOOTP relay agents can identify the subnet mask of the foreign BOOTP/DHCP client's network. See `netmasks(4)` for the format and use of this database.

`-t` *dhcptab_rescan_interval*
: Specifies the interval in minutes that the DHCP server should use to schedule the automatic rereading of the `dhcptab` information. Typically, you would use this option if the changes to the `dhcptab` are relatively frequent. Once the contents of the `dhcptab` have stabilized, you can turn off this option to avoid needless reinitialization of the server.

`-v`
: Verbose mode. The daemon displays more messages than in the default mode. Note that verbose mode can reduce daemon efficiency due to the time taken to display messages. Messages are displayed to the current TTY if the debugging option is used; otherwise, messages are logged to the `syslogd` facility. This option can be used in both DHCP/BOOTP server mode and BOOTP relay agent mode.

Examples

EXAMPLE 1 Starting a DHCP Server in BOOTP Compatibility Mode

The following command starts a DHCP server in BOOTP compatibility mode, permitting the server to automatically allocate permanent IP addresses to BOOTP clients which are not registered in the server's table; limits the server's attention to incoming datagrams on network devices `le2` and `tr0`; drops BOOTP packets whose hop count exceeds 2; configures the DHCP server to cache extended DHCP offers for 15 seconds; and schedules `dhcptab` rescans to occur every 10 minutes:

```
# in.dhcpd -i le2,tr0 -h 2 -o 15 -t 10 -b automatic
```

EXAMPLE 2 Starting the Daemon in BOOTP Relay Agent Mode

The following command starts the daemon in BOOTP relay agent mode, registering the hosts `bladerunner` and `10.0.0.5` as relay destinations, with debugging and verbose modes enabled, and drops BOOTP packets whose hop count exceeds 5:

```
# in.dhcpd -d -v -h 5 -r bladerunner,10.0.0.5
```

Files

`/etc/inet/dhcpsvc.conf`

`/etc/init/hosts`

`/usr/lib/inet/dhcp/nsu/rfc2136.so.1`

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdhcsu
Interface Stability	Evolving

See Also svcs(1), cron(1M), dhcpmgr(1M), dhtadm(1M), inetadm(1M), inetd(1M), logadm(1M), pntadm(1M), svcadm(1M), syslogd(1M), syslog(3C), dhcpsvc.conf(4), dhcp_network(4), dhcptab(4), ethers(4), hosts(4), netmasks(4), nsswitch.conf(4), attributes(5), dhcp(5), smf(5)

System Administration Guide: IP Services

Alexander, S., and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, RFC 2132, Silicon Graphics, Inc., Bucknell University, March 1997.

Droms, R., *Interoperation Between DHCP and BOOTP*, RFC 1534, Bucknell University, October 1993.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 2131, Bucknell University, March 1997.

Wimer, W., *Clarifications and Extensions for the Bootstrap Protocol*, RFC 1542, Carnegie Mellon University, October 1993.

Notes The in.dhcpd service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/dhcp-server
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). Responsibility for initiating and restarting this service is delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

Name in.discardd – UDP or TCP discard protocol service

Synopsis

```
in.discardd

FMRI svc:/internet/discard:default
```

Description FMRI stands for Fault Management Resource Identifier. It is used to identify resources managed by the Fault Manager. See `fmd(1M)` and `smf(5)`.

The `in.discardd` service provides the server-side of the discard protocol. This protocol is used for debugging and bandwidth measurement and is available on both TCP and UDP transports through port 9.

The `in.discardd` service is an `inetd(1M)` `smf(5)` delegated service. The `in.discardd` detects which transport is requested by examining the socket it is passed by the `inetd` daemon.

The discard service simply throws away any data it receives from the client.

TCP-based service — Once a connection is established, the `in.discardd` discards any data received. No response is generated. The connection remains open until the client terminates it.

UDP-based service — The `in.discardd` listens for UDP datagrams. When a datagram is received, the server discards it. No response is sent.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcnsu
Interface Stability	Evolving

See Also `inetd(1M)`, `attributes(5)`, `smf(5)`

RFC 863

Name in.echod – UDP or TCP echo protocol service daemon

Synopsis

```
in.echod

FMRI svc:/internet/echo:default
```

Description FMRI stands for Fault Management Resource Identifier. It is used to identify resources managed by the Fault Manager. See `fmd(1M)` and `smf(5)`.

The `in.echod` service provides the server-side of the echo protocol. This protocol is used for debugging and bandwidth measurement and is available on both TCP and UDP transports, through port 7.

The `in.echod` service is an `inetd(1M)` `smf(5)` delegated service. The `in.echod` detects which transport is requested by examining the socket it is passed by the `inetd` daemon.

TCP-based service — Once a connection is established, the `in.echod` echoes any data received from the client back to the client. The server echoes data until the client program terminates the connection.

UDP-based service — The `in.echod` listens for UDP datagrams. When a datagram is received, the server creates a UDP datagram containing the data it received and sends it to the client.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNW
Interface Stability	Evolving

See Also `inetd(1M)`, `attributes(5)`, `smf(5)`

RFC 862

Name inetadm – observe or configure inetd-controlled services

Synopsis

```
inetadm
inetadm -?
inetadm -p
inetadm -l {FMRI | pattern}
inetadm -e {FMRI | pattern}
inetadm -d {FMRI | pattern}
inetadm -m {FMRI | pattern}... {name=value}...
inetadm -M {name=value}...
```

Description The `inetadm` utility provides the following capabilities for `inetd`-managed SMF services:

- Provides a list of all such services installed.
- Lists the services' properties and values.
- Allows enabling and disabling of services.
- Allows modification of the services'property values, as well as the default values provided by `inetd`.

See `smf(5)` for a description of an SMF service.

With no arguments, `inetadm` lists all services under `inetd(1M)` control, including such attributes as their current run state and whether or not they are enabled.

Options For options taking one or more FMRI operands (see `smf(5)` for a description of an FMRI), if the operand specifies a service (instead of a service instance), and that service has only a single instance, `inetadm` operates on that instance.

If a service name is supplied and it contains more than one instances or a pattern is supplied and and it matches more than one instance, a warning message is displayed and that operand is ignored.

For those options taking *name=value* parameters, a description of each of the possible names and the allowed values is found in the `inetd(1M)` man page.

The following options are supported:

`-?`	Display a usage message.
`-p`	Lists all default `inet` service property values provided by `inetd` in the form of *name=value* pairs. If the value is of boolean type, it is listed as TRUE or FALSE.

`-l` {*FMRI* | *pattern*}...	List all properties for the specified service instances as *name*=*value* pairs. In addition, if the property value is inherited from the default value provided by `inetd`, the *name*=*value* pair is identified by the token (default). Property inheritance occurs when properties do not have a specified service instance default.
`-e` {*FMRI* | *pattern*}...	Enable the specified service instances.
`-d` {*FMRI* | *pattern*}...	Disable the specified service instances.
`-m` {*FMRI* | *pattern*}...{*name*=*value*}...	Change the values of the specified properties of the identified service instances. Properties are specified as whitespace-separated *name*=*value* pairs. To remove an instance-specific value and accept the default value for a property, simply specify the property without a value, for example, `name=` .
`-M` {*name*=*value*}...	Change the values of the specified `inetd` default properties. Properties are specified as whitespace-separated *name*=*value* pairs.

Examples

EXAMPLE 1 Displaying Properties for a Service

The following command displays the properties for the `spray` service.

```
# inetadm -l network/rpc/spray:default
SCOPE    NAME=VALUE
         name="sprayd"
         endpoint_type="tli"
         proto="datagram_v"
         isrpc=TRUE
         rpc_low_version=1
         rpc_high_version=1
         wait=TRUE
         exec="/usr/lib/netsvc/spray/rpc.sprayd"
         user="root"
default  bind_addr=""
default  bind_fail_max=-1
default  bind_fail_interval=-1
default  max_con_rate=-1
default  max_copies=-1
default  con_rate_offline=-1
default  failrate_cnt=40
default  failrate_interval=60
default  inherit_env=TRUE
default  tcp_trace=FALSE
```

EXAMPLE 1 Displaying Properties for a Service *(Continued)*

```
default  tcp_wrappers=FALSE
default  connection_backlog=10
```

EXAMPLE 2 Displaying Default Properties

The following command displays default properties.

```
# inetadm -p
NAME=VALUE
bind_addr=""
bind_fail_max=-1
bind_fail_interval=-1
max_con_rate=-1
max_copies=-1
con_rate_offline=-1
failrate_cnt=40
failrate_interval=60
inherit_env=TRUE
tcp_trace=FALSE
tcp_wrappers=FALSE
default  connection_backlog=10
```

EXAMPLE 3 Changing Property Values for a Service

The following command changes `rpc_high_version` to `3` and `tcp_trace` to `TRUE` for the `spray` service.

```
# inetadm -m network/rpc/spray:default \
         rpc_high_version=3 tcp_trace=TRUE
# inetadm -l network/rpc/spray:default
SCOPE    NAME=VALUE
         name="sprayd"
         endpoint_type="tli"
         proto="datagram_v"
         isrpc=TRUE
         rpc_low_version=1
         rpc_high_version=3
         wait=TRUE
         exec="/usr/lib/netsvc/spray/rpc.sprayd"
         user="root"
default  bind_addr=""
default  bind_fail_max=-1
default  bind_fail_interval=-1
default  max_con_rate=-1
default  max_copies=-1
default  con_rate_offline=-1
default  failrate_cnt=40
```

EXAMPLE 3 Changing Property Values for a Service *(Continued)*

```
default  failrate_interval=60
default  inherit_env=TRUE
         tcp_trace=TRUE
default  tcp_wrappers=FALSE
default  connection_backlog=10
```

Exit Status The following exit values are returned:

`0` Operation completed successfully.

`1` A fatal error occurred. An accompanying error message will provide further information.

`2` Invalid arguments were supplied, such as an ambiguous service FMRI or pattern.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also `inetd(1M)`, `svccfg(1M)`, `attributes(5)`, `smf(5)`

Name inetconv – convert `inetd.conf` entries into smf service manifests, import them into smf repository

Synopsis

```
inetconv -?

inetconv [-f] [-n] [-i srcfile] [-o destdir]

inetconv -e [-n] [-i srcfile]
```

Description The `inetconv` utility converts a file containing records of `inetd.conf(4)` into `smf(5)` service manifests, and then import those manifests into the `smf` repository. Once the `inetd.conf` file has been converted, the only way to change aspects of an inet service is to use the `inetadm(1M)` utility.

There is a one-to-one correspondence between a service line in the input file and the manifest generated. By default, the manifests are named using the following template:

<svcname>-<proto>`.xml`

The *<svcname>* token is replaced by the service's name and the *<proto>* token by the service's protocol. Any slash (`/`) characters that exist in the source line for the service name or protocol are replaced with underscores (_).

The service line is recorded as a property of the converted service.

During the conversion process, if a service line is found to be malformed or to be for an internal `inetd` service, no manifest is generated and that service line is skipped.

The input file is left untouched by the conversion process.

Options The following options are supported:

`-?` Display a usage message.

`-e` Enable `smf` services which are listed in the input file.

`-f` If a service manifest of the same name as the one to be generated is found in the destination directory, `inetconv` will overwrite that manifest if this option is specified. Otherwise, an error message is generated and the conversion of that service is not performed.

`-i` *srcfile* Permits the specification of an alternate input file *srcfile*. If this option is not specified, then the `inetd.conf(4)` file is used as input.

`-n` Turns off the auto-import of the manifests generated during the conversion process. Later, if you want to import a generated manifest into the `smf(5)` repository, you can do so through the use of the `svccfg(1M)` utility.

If the `-e` option is specified, the `-n` option only displays the `smf` services that would be enabled.

`-o` Permits the specification of an alternate destination directory *destdir* for the generated manifests. If this option is not specified, then the manifests are placed in `/var/svc/manifest/network/rpc`, if the service is a RPC service, or `/var/svc/manifest/network` otherwise.

Examples

EXAMPLE 1 Generating `smf` Manifests from `inetd.conf`

The following command generates `smf(5)` manifests from `inetd.conf(4)` and places them in `/var/tmp`, overwriting any preexisting manifests of the same name, and then imports them into the `smf` repository.

```
# inetconv -f -o /var/tmp
100232/10 -> /var/tmp/100232_10-rpc_udp.xml
Importing 100232_10-rpc_udp.xml ...Done
telnet -> /var/tmp/telnet-tcp6.xml
Importing telnet-tcp6.xml ...Done
```

EXAMPLE 2 Generating Manifests from an Alternate Input File

The following command specifies a different input file and does not load the resulting manifests into the `smf` repository.

```
# inetconv -n -i /export/test/inet.svcs -o /var/tmp
100232/10 -> /var/tmp/100232_10-rpc_udp.xml
telnet -> /var/tmp/telnet-tcp6.xml
```

Exit Status

The following exit values are returned:

`0` Operation completed successfully (no errors).

`1` Invalid options specified.

`2` One or more service lines are malformed, and thus no manifest(s) were generated for them.

`3` An error occurred importing one or more of the generated manifests.

`4` A system error occurred.

Files

`/var/svc/manifest/network/{rpc}/<`*svcname*`>-<`*proto*`>.xml`
default output manifest file name

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also inetadm(1M), inetd(1M), svccfg(1M), inetd.conf(4), attributes(5), smf(5)

Name inetd – Solaris Management Facility delegated restarter for inet services

Synopsis

```
inetd [configuration-file] start | stop | refresh

svc:/network/inetd:default
```

Description `inetd` is the delegated restarter for internet services for the Service Management Facility (SMF). Its basic responsibilities are to manage service states in response to administrative requests, system failures, and service failures; and, when appropriate, to listen for network requests for services.

Services are no longer managed by editing the inetd configuration file, `inetd.conf(4)`. Instead, you use `inetconv(1M)` to convert the configuration file content into SMF format services, then manage these services using `inetadm(1M)` and `svcadm(1M)`. Once a service has been converted by `inetconv`, any changes to the legacy data in the `inetd` config file will not become effective. However, `inetd` does alert the administrator when it notices change in the configuration file. See the start description under the "inetd Methods" section for further information.

Also note that the current `inetd` cannot be run from outside the SMF. This means it cannot be run from the command line, as was supported by the previous `inetd`. If you attempt to do this, a message is sent to stderr displaying mappings between the options supported by the previous `inetd` to the SMF version of `inetd`.

`inetd` listens for connections on behalf of all services that are in either the `online` or `degraded` state. A service enters one of these states when the service is enabled by the user and `inetd` manages to listen on its behalf. A listen attempt can fail if another server (whether standalone or a third-party internet service) is already listening on the same port. When this occurs, `inetd` logs this condition and continues trying to bind to the port at configured intervals a configured number of times. See the property `bind_fail_max` under "Service Properties," below, for more details.

The configuration of all `inetd`'s managed SMF services is read when it is started. It is reread when `inetd` is refreshed, which occurs in response to an SMF request, or when it receives a SIGHUP signal. See the `refresh` description under "inetd Methods" for the behavior on configuration refresh.

You can use the `inetadm(1M)` or `svccfg(1M)` utilities to make configuration changes to Internet services within the SMF repository. `inetadm` has the advantage over `svccfg` in that it provides an Internet/RPC service context.

Service States As part of its service management duties, `inetd` implements a state machine for each of its managed services. The states in this machine are made up of the `smf(5)` set of states. The semantics of these states are as follows:

`uninitialized`
: `inetd` has yet to process this service.

`online`
The service is handling new network requests and might have existing connections active.

`degraded`
The service has entered this state because it was able to listen and process requests for some, but not all, of the protocols specified for the service, having exhausted its listen retries. Existing network connections might be active.

`offline`
Connections might be active, but no new requests are being handled. This is a transient state. A service might be `offline` for any of the following reasons:

- The service's dependencies are unmet. When its dependencies become met the service's state will be re-evaluated.
- The service has exceeded its configured connection rate limit, `max_con_rate`. The service's state is re-evaluated when its connection offline timer, `con_rate_offline`, expires.
- The service has reached its allowed number of active connections, `max_copies`. The service's state is re-evaluated when the number of active connections drops below `max_copies`.
- `inetd` failed to listen on behalf of the service on all its protocols. As mentioned above, `inetd` retries up to a configured maximum number of times, at configured intervals.The service's state is re-evaluated when either a listen attempt is successful or the retry limit is reached.

`disabled`
The service has been turned off by an administrator, is not accepting new connections, and has none active. Administrator intervention is required to exit this state.

`maintenance`
A service is in this state because it is either malfunctioning and needs adminstrator attention or because an administrator has requested it.

Events constituting malfunctioning include: `inetd`'s inability to listen on behalf on any of the service's protocols before exceeding the service's bind retry limit, non-start methods returning with non-success return values, and the service exceeding its failure rate.

You request the maintenance state to perform maintenance on the service, such as applying a patch. No new requests are handled in this state, but existing connections might be active. Administrator intervention is required to exit this state.

Use `inetadm(1M)` to obtain the current state of a managed service.

Service Methods As part of certain state transitions `inetd` will execute, if supplied, one of a set of methods provided by the service. The set of supported methods are:

`inetd_start`
Executed to handle a request for an `online` or `degraded` service. Since there is no separate state to distinguish a service with active connections, this method is not executed as part of a state transition.

`inetd_offline`
Executed when a service is taken from the `online` or `degraded` state to the `offline` state. For a `wait`-type service that at the time of execution is performing its own listening, this method should result in it ceasing listening. This method will be executed before the `disable` method in the case an `online`/`degraded` service is disabled. This method is required to be implemented for a `wait`-type service.

`inetd_online`
Executed when a service transitions from the `offline` state to the `online` state. This method allows a service author to carry out some preparation prior to a service starting to handle requests.

`inetd_disable`
Executed when a service transitions from the `offline` state to the `disabled` state. It should result in any active connections for a service being terminated.

`inetd_refresh`
Executed when both of the following conditions are met:

- `inetd` is refreshed, by means of the framework or a SIGHUP, or a request comes in to refresh the service, and
- the service is currently in the `online` state and there are no configuration changes that would result in the service needing to be taken `offline` and brought back again.

The only compulsory method is the `inetd_start` method. In the absence of any of the others, `inetd` runs no method but behaves as if one was run successfully.

Service Properties

Configuration for SMF–managed services is stored in the SMF repository. The configuration is made up of the basic configuration of a service, the configuration for each of the service's methods, and the default configuration applicable to all `inetd`-managed services.

For details on viewing and modifying the configuration of a service and the defaults, refer to `inetadm(1M)`.

The basic configuration of a service is stored in a property group named `inetd` in the service. The properties comprising the basic configuration are as follows:

`bind_addr`
The address of the network interface to which the service should be bound. An empty string value causes the service to accept connections on any network interface.

`bind_fail_interval`
The time interval in seconds between a failed bind attempt and a retry. The values `0` and `-1` specify that no retries are attempted and the first failure is handled the same as exceeding `bind_fail_max`.

`bind_fail_max`
The maximum number of times `inetd` retries binding to a service's associated port before giving up. The value `-1` specifies that no retry limit is imposed. If none of the service's protocols were bound to before any imposed limit is reached, the service goes to the `maintenance` state; otherwise, if not all of the protocols were bound to, the service goes to the `degraded` state.

`con_rate_offline`
The time in seconds a service will remain offline if it exceeds its configured maximum connection rate, `max_con_rate`. The values `0` and `-1` specify that connection rate limiting is disabled.

`connection_backlog`
The backlog queue size. Represents a limit on the number of incoming client requests that can be queued at the listening endpoints for servers.

`endpoint_type`
The type of the socket used by the service or the value `tli` to signify a TLI-based service. Valid socket type values are: `stream`, `dgram`, `raw`, `seqpacket`.

`failrate_cnt`
The count portion of the service's failure rate limit. The failure rate limit applies to `wait`-type services and is reached when *count* instances of the service are started within a given time. Exceeding the rate results in the service being transitioned to the `maintenance` state. This is different from the behavior of the previous `inetd`, which continued to retry every 10 minutes, indefinitely. The `failrate_cnt` check accounts for badly behaving servers that fail before consuming the service request and which would otherwise be continually restarted, taxing system resources. Failure rate is equivalent to the `-r` option of the previous `inetd`. The values `0` and `-1` specify that this feature is disabled.

`failrate_interval`
The time portion in seconds of the service's failure rate. The values `0` and `-1` specify that the failure rate limit feature is disabled.

`inherit_env`
If true, pass `inetd`'s environment on to the service's start method. Regardless of this setting, `inetd` will set the variables `SMF_FMRI`, `SMF_METHOD`, and `SMF_RESTARTER` in the start method's environment, as well as any environment variables set in the method context. These variables are described in `smf_method(5)`.

`isrpc`
If true, this is an RPC service.

`max_con_rate`
The maximum allowed connection rate, in connections per second, for a `nowait`-type service. The values `0` and `-1` specify that that connection rate limiting is disabled.

`max_copies`
The maximum number of copies of a `nowait` service that can run concurrently. The values `0` and `-1` specify that copies limiting is disabled.

`name`
Can be set to one of the following values:

- a service name understood by `getservbyname(3SOCKET)`;
- if `isrpc` is set to `true`, a service name understood by `getrpcbyname(3NSL)`;
- if `isrpc` is set to `true`, a valid RPC program number.

`proto`
In the case of socket-based services, this is a list of protocols supported by the service. Valid protocols are: `tcp`, `tcp6`, `tcp6only`, `udp`, `udp6`, and `udp6only`. In the case of TLI services, this is a list of netids recognized by `getnetconfigent(3NSL)` supported by the service, plus the values `tcp6only` and `udp6only`. RPC/TLI services also support nettypes in this list, and `inetd` first tries to interpret the list member as a nettype for these service types. The values `tcp6only` and `udp6only` are new to `inetd`; these values request that `inetd` listen only for and pass on true `IPv6` requests (not IPv4 mapped ones). See "Configuring Protocols for Sockets-Based Services," below.

`rpc_low_version`
Lowest supported RPC version. Required when `isrpc` is set to `true`.

`rpc_high_version`
Highest supported RPC version. Required when `isrpc` is set to `true`.

`tcp_trace`
If true, and this is a `nowait`-type service, `inetd` logs the client's IP address and TCP port number, along with the name of the service, for each incoming connection, using the `syslog(3C)` facility. `inetd` uses the `syslog` facility `code` daemon and `notice` priority level. See `syslog.conf(4)` for a description of `syslog` codes and severity levels. This logging is separate from the logging done by the TCP wrappers facility.

`tcp_trace` is equivalent to the previous `inetd`'s `-t` option (and the `/etc/default/inetd` property `ENABLE_CONNECTION_LOGGING`).

`tcp_wrappers`
If `true`, enable TCP wrappers access control. This applies only to services with `endpoint_type` set to `streams` and `wait` set to `false`. The `syslog` facility `code` daemon is used to log allowed connections (using the `notice` severity level) and denied traffic (using the `warning` severity level). See `syslog.conf(4)` for a description of `syslog` codes and severity levels. The stability level of the TCP wrappers facility and its configuration files is External. As the TCP wrappers facility is not controlled by Sun, intra-release incompatibilities are not uncommon. See `attributes(5)`.

For more information about configuring TCP wrappers, you can refer to the `tcpd(1M)` and `hosts_access(4)` man pages, which are delivered as part of the Solaris operating system at `/usr/sfw/man`. These pages are not part of the standard Solaris man pages, available at `/usr/man`.

`tcp_wrappers` is equivalent to the previous inetd's `/etc/default/inetd` property `ENABLE_TCPWRAPPERS`.

`wait`
If `true` this is a `wait`-type service, otherwise it is a `nowait`-type service. A `wait`-type service has the following characteristics:

- Its `inetd_start` method will take over listening duties on the service's bound endpoint when it is executed.
- `inetd` will wait for it to exit after it is executed before it resumes listening duties.

Datagram servers must be configured as being of type `wait`, as they are always invoked with the original datagram endpoint that will participate in delivering the service bound to the specified service. They do not have separate "listening" and "accepting" sockets. Connection-oriented services, such as TCP stream services can be designed to be either of type `wait` or `nowait`.

A number of the basic properties are optional for a service. In their absence, their values are taken from the set of default values present in the `defaults` property group in the `inetd` service. These properties, with their seed values, are listed below. Note that these values are configurable through `inetadm(1M)`.

```
bind_fail_interval  -1
bind_fail_max       -1
con_rate_offline    -1
connection_backlog  10
failrate_count      40
failrate_time       60
inherit_env         true
max_con_rate        -1
max_copies          -1
tcp_trace           false
tcp_wrappers        false
```

Each method specified for a service will have its configuration stored in the SMF repository, within a property group of the same name as the method. The set of properties allowable for these methods includes those specified for the services managed by `svc.startd(1M)`. (See `svc.startd(1M)` for further details.) Additionally, for the `inetd_start` method, you can set the `arg0` property.

The `arg0` property allows external wrapper programs to be used with `inetd` services. Specifically, it allows the first argument, `argv[0]`, of the service's start method to be something other than the path of the server program.

In the case where you want to use an external wrapper program and pass arguments to the service's daemon, the arguments should be incorporated as arguments to the wrapper program in the exec property. For example:

```
exec='/path/to/wrapper/prog service_daemon_args'
arg0='/path/to/service/daemon'
```

In addition to the special method tokens mentioned in smf_method(5), inetd also supports the :kill_process token for wait-type services. This results in behavior identical to that if the :kill token were supplied, except that the kill signal is sent only to the parent process of the wait-type service's start method, not to all members of its encompassing process contract (see process(4)).

Configuring Protocols for Sockets-Based Services

When configuring inetd for a sockets-based service, you have the choice, depending on what is supported by the service, of the alternatives described under the proto property, above. The following are guidelines for which proto values to use:

- For a service that supports only IPv4: tcp and udp
- For a service that supports only IPv6: tcp6only and udp6only
- For a service that supports both IPv4 and IPv6:
  - Obsolete and not recommended: tcp6 and udp6
  - Recommended: use two separate entries that differ only in the proto field. One entry has tcp and the other has tcp6only, or udp plus udp6only.

See EXAMPLES for an example of a configuration of a service that supports both IPv4 and IPv6.

inetd Methods

inetd provides the methods listed below for consumption by the master restarter, svc.startd(1M).

start
: Causes inetd to start providing service. This results in inetd beginning to handle smf requests for its managed services and network requests for those services that are in either the online or degraded state.

  In addition, inetd also checks if the inetd.conf(4)–format configuration file it is monitoring has changed since the last inetconv(1M) conversion was carried out. If it has, then a message telling the administrator to re-run inetconv to effect the changes made is logged in syslog.

stop
: Causes inetd to stop providing service. At this point, inetd transitions each of its services that are not in either the maintenance or disabled states to the offline state, running any appropriate methods in the process.

`refresh`
Results in a refresh being performed for each of its managed services and the `inetd.conf(4)` format configuration file being checked for change, as in the `start` method. When a service is refreshed, its behavior depends on its current state:

- if it is in the `maintenance` or `disabled` states, no action is performed because the configuration will be read and consumed when the service leaves the state;
- if it is in the `offline` state, the configuration will be read and any changes consumed immediately;
- if it is in the `online` or `degraded` state and the configuration has changed such that a re-binding is necessary to conform to it, then the service will be transitioned to the `offline` state and back again, using the new configuration for the bind;
- if it is in the `online` state and a re-binding is not necessary, then the `inetd_refresh` method of the service, if provided, will be run to allow `online` `wait`–type services to consume any other changes.

Options No options are supported.

Operands *configuration-file*
Specifies an alternate location for the legacy service file (`inetd.conf(4)`).

`start|stop|refresh`
Specifies which of `inetd`'s methods should be run.

Examples **EXAMPLE 1** Configuring a Service that Supports Both IPv4 and IPv6

The following commands illustrate the existence of services that support both IPv4 and IPv6 and assign `proto` properties to those services.

```
example# svcs -a | grep mysvc
online         15:48:29 svc:/network/mysvc:dgram4
online         15:48:29 svc:/network/mysvc:dgram6
online         15:51:47 svc:/network/mysvc:stream4
online         15:52:10 svc:/network/mysvc:stream6

# inetadm -M network/rpc/mysvc:dgram4 proto=udp
# inetadm -M network/rpc/mysvc:dgram6 proto=udp6only
# inetadm -M network/rpc/mysvc:stream4 proto=tcp
# inetadm -M network/rpc/mysvc:stream6 proto=tcp6only
```

See `svcs(1)` and `inetadm(1M)` for descriptions of those commands.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also fmd(1M), inetadm(1M), inetconv(1M), svcadm(1M), svccfg(1M), svcs(1), svc.startd(1M), syslog(3C), getnetconfigent(3NSL), getrpcbyname(3NSL), getservbyname(3SOCKET), inetd.conf(4), process(4), syslog.conf(4), attributes(5), smf(5), smf_method(5)

Notes The inetd daemon performs the same function as, but is implemented significantly differently from, the daemon of the same name in Solaris 9 and prior Solaris operating system releases. In the current Solaris release, inetd is part of the Solaris Management Facility (see smf(5)) and will run only within that facility.

The /etc/default/inetd file has been deprecated. The functionality represented by the properties ENABLE_CONNECTION_LOGGING and ENABLE_TCP_WRAPPERS are now available as the tcp_trace and tcp_wrappers properties, respectively. These properties are described above, under "Service Properties".

Name in.fingerd, fingerd – remote user information server

Synopsis

```
/usr/sbin/in.fingerd
```

Description `fingerd` implements the server side of the Name/Finger protocol, specified in *RFC 742*. The Name/Finger protocol provides a remote interface to programs which display information on system status and individual users. The protocol imposes little structure on the format of the exchange between client and server. The client provides a single command line to the finger server which returns a printable reply.

`fingerd` waits for connections on TCP port 79. Once connected, it reads a single command line terminated by RETURN-LINEFEED and passes the arguments to `finger(1)`, prepended with `-s`. `fingerd` closes its connections as soon as the output is finished.

Files

`/var/adm/utmpx`	User and accounting information.
`/etc/passwd`	System password file.
`/var/adm/lastlog`	Last login times.
`$HOME/.plan`	User's plans.
`$HOME/.project`	User's projects.

Usage `fingerd` and `in.fingerd` are IPv6–enabled. See `ip6(7P)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also `finger(1)`, `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`, `ip6(7P)`

Harrenstien, Ken, *RFC 742, NAME/FINGER*, Network Information Center, SRI International, Menlo Park, Calif., December 1977.

Notes Connecting directly to the server from a TIP or an equally narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which foul up the command line interpretation. `fingerd` should be taught to filter out IAC's and perhaps even respond negatively (IAC does not) to all option commands received.

The `in.fingerd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/finger:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is

delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

Name infocmp – compare or print out terminfo descriptions

Synopsis

```
/usr/bin/infocmp [-d] [-c] [-n] [-I] [-L] [-C] [-r] [-u]
     [-s | d | i | l | c] [-v] [-V] [-1] [-w width]
     [-A directory] [-B directory] [termname]...
```

Description `infocmp` compares a binary `terminfo` entry with other terminfo entries, rewrites a `terminfo` description to take advantage of the `use=` terminfo field, or prints out a `terminfo` description from the binary file (`term`) in a variety of formats. It displays boolean fields first, then numeric fields, followed by the string fields. If no options are specified and zero, or one *termname* is specified, the `-I` option is assumed. If more than one *termname* is specified, the `-d` option is assumed.

Options The `-d` , `-c` , and `-n` options can be used for comparisons. `infocmp` compares the `terminfo` description of the first terminal *termname* with each of the descriptions given by the entries for the other terminal's *termname*. If a capability is defined for only one of the terminals, the value returned will depend on the type of the capability: `F` for boolean variables, –`1` for integer variables, and `NULL` for string variables.

`-d` Produce a list of each capability that is different between two entries. This option is useful to show the difference between two entries, created by different people, for the same or similar terminals.

`-c` Produce a list of each capability that is common between two entries. Capabilities that are not set are ignored. This option can be used as a quick check to see if the `-u` option is worth using.

`-n` Produce a list of each capability that is in neither entry. If no *termname* is given, the environment variable `TERM` will be used for both of the *termname*s. This can be used as a quick check to see if anything was left out of a description.

The `-I` , `-L` , and `-C` options will produce a source listing for each terminal named.

`-I` Use the `terminfo` names.

`-L` Use the long C variable name listed in < `term.h` >.

`-C` Use the `termcap` names. The source produced by the `-C` option may be used directly as a `termcap` entry, but not all of the parameterized strings may be changed to the `termcap` format. `infocmp` will attempt to convert most of the parameterized information, but anything not converted will be plainly marked in the output and commented out. These should be edited by hand.

`-r` When using `-C` , put out all capabilities in `termcap` form.

If no *termname* is given, the environment variable `TERM` will be used for the terminal name.

All padding information for strings will be collected together and placed at the beginning of the string where `termcap` expects it. Mandatory padding (padding information with a trailing '/') will become optional.

All termcap variables no longer supported by terminfo , but are derivable from other terminfo variables, will be displayed. Not all terminfo capabilities will be translated; only those variables which were part of termcap will normally be displayed. Specifying the -r option will take off this restriction, allowing all capabilities to be displayed in termcap form.

Note that because padding is collected to the beginning of the capability, not all capabilities are displayed. Mandatory padding is not supported. Because termcap strings are not as flexible, it is not always possible to convert a terminfo string capability into an equivalent termcap format. A subsequent conversion of the termcap file back into terminfo format will not necessarily reproduce the original terminfo source.

Some common terminfo parameter sequences, their termcap equivalents, and some terminal types which commonly have such sequences, are:

```
terminfo     termcap      Representative Terminals
%p1%c    %.    adm
%p1%d    %d    hp, ANSI standard, vt100
%p1%'x'%+%c    %+x    concept
%i    %i    ANSI standard, vt100
%p1%?%'x'%>%t%p1%'y'%+%;    %>xy    concept
%p2 is printed before %p1    %r    hp
```

-u Produce a terminfo source description of the first terminal *termname* which is relative to the sum of the descriptions given by the entries for the other terminals' *termname*s. It does this by analyzing the differences between the first *termname* and the other *termnames* and producing a description with use= fields for the other terminals. In this manner, it is possible to retrofit generic terminfo entries into a terminal's description. Or, if two similar terminals exist, but were coded at different times, or by different people so that each description is a full description, using infocmp will show what can be done to change one description to be relative to the other.

A capability is displayed with an at-sign (@) if it no longer exists in the first *termname*, but one of the other *termname* entries contains a value for it. A capability's value is displayed if the value in the first *termname* is not found in any of the other *termname* entries, or if the first of the other *termname* entries that has this capability gives a different value for that capability.

The order of the other *termname* entries is significant. Since the terminfo compiler tic does a left-to-right scan of the capabilities, specifying two use= entries that contain differing entries for the same capabilities will produce different results, depending on the order in which the entries are given. infocmp will flag any such inconsistencies between the other *termname* entries as they are found.

Alternatively, specifying a capability *after* a use= entry that contains, it will cause the second specification to be ignored. Using infocmp to recreate a description can be a useful check to make sure that everything was specified correctly in the original source description.

Another error that does not cause incorrect compiled files, but will slow down the compilation time, is specifying superfluous `use=` fields. `infocmp` will flag any superfluous `use=` fields.

`-s` Sorts the fields within each type according to the argument below:

- `d` Leave fields in the order that they are stored in the `terminfo` database.
- `i` Sort by `terminfo` name.
- `l` Sort by the long C variable name.
- `c` Sort by the `termcap` name.

If the `-s` option is not given, the fields are sorted alphabetically by the `terminfo` name within each type, except in the case of the `-C` or the `-L` options, which cause the sorting to be done by the `termcap` name or the long C variable name, respectively.

`-v` Print out tracing information on standard error as the program runs.

`-V` Print out the version of the program in use on standard error and exit.

`–1` Print the fields one to a line. Otherwise, the fields are printed several to a line to a maximum width of 60 characters.

`-w`*width* Changes the output to *width* characters.

The location of the compiled `terminfo` database is taken from the environment variable `TERMINFO` . If the variable is not defined, or the terminal is not found in that location, the system `terminfo` database, usually in `/usr/share/lib/terminfo`, is used. The options `-A` and `-B` may be used to override this location.

`-A` *directory* Set `TERMINFO` for the first *termname*.

`-B` *directory* Set `TERMINFO` for the other *termname*s. With this, it is possible to compare descriptions for a terminal with the same name located in two different databases. This is useful for comparing descriptions for the same terminal created by different people.

Files `/usr/share/lib/terminfo/?/*` Compiled terminal description database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `captoinfo(1M)`, `tic(1M)`, `curses(3CURSES)`, `terminfo(4)`, `attributes(5)`

Name in.ftpd, ftpd – File Transfer Protocol Server

Synopsis

```
in.ftpd [-4] [-A] [-a] [-C] [-d] [-I] [-i] [-K] [-L] [-l]
     [-o] [-P dataport] [-p ctrlport] [-Q] [-q]
     [-r rootdir] [-S] [-s] [-T maxtimeout] [-t timeout]
     [-u umask] [-V] [-v] [-W] [-w] [-X]
```

Description `in.ftpd` is the Internet File Transfer Protocol (FTP) server process. The server may be invoked by the Internet daemon `inetd(1M)` each time a connection to the FTP service is made or run as a standalone server. See `services(4)`.

Options `in.ftpd` supports the following options:

`-4` When running in standalone mode, listen for connections on an `AF_INET` type socket. The default is to listen on an `AF_INET6` type socket.

`-a` Enables use of the `ftpaccess(4)` file.

`-A` Disables use of the `ftpaccess(4)` file. Use of `ftpaccess` is disabled by default.

`-C` Non-anonymous users need local credentials (for example, to authenticate to remote fileservers). So they should be prompted for a password unless they forwarded credentials as part of authentication.

`-d` Writes debugging information to `syslogd(1M)`.

`-i` Logs the names of all files received by the FTP Server to `xferlog(4)`. You can override the `-i` option through use of the `ftpaccess(4)` file.

`-I` Disables the use of `AUTH` and `ident` to determine the username on the client. See *RFC 931*. The FTP Server is built not to use `AUTH` and `ident`.

`-K` Connections are only allowed for users who can authenticate through the `ftp AUTH` mechanism. (Anonymous `ftp` may also be allowed if it is configured.) `ftpd` will ask the user for a password if one is required.

`-l` Logs each FTP session to `syslogd(1M)`.

`-L` Logs all commands sent to `in.ftpd` to `syslogd(1M)`. When the `-L` option is used, command logging will be on by default, once the FTP Server is invoked. Because the FTP Server includes `USER` commands in those logged, if a user accidentally enters a password instead of the username, the password will be logged. You can override the `-L` option through use of the `ftpaccess(4)` file.

`-o` Logs the names of all files transmitted by the FTP Server to `xferlog(4)`. You can override the `-o` option through use of the `ftpaccess(4)` file.

`-P` *dataport* The FTP Server determines the port number by looking in the `services(4)` file for an entry for the `ftp-data` service. If there is no entry,

	the daemon uses the port just prior to the control connection port. Use the `-P` option to specify the data port number.
`-p` *ctrlport*	When run in standalone mode, the FTP Server determines the control port number by looking in the `services(4)` file for an entry for the `ftp` service. Use the `-p` option to specify the control port number.
`-Q`	Disables PID files. This disables user limits. Large, busy sites that do not want to impose limits on the number of concurrent users can use this option to disable `PID` files.
`-q`	Uses PID files. The `limit` directive uses `PID` files to determine the number of current users in each access class. By default, `PID` files are used.
`-r` *rootdir*	`chroot(2)` to *rootdir* upon loading. Use this option to improve system security. It limits the files that can be damaged should a break in occur through the daemon. This option is similar to anonymous FTP. Additional files are needed, which vary from system to system.
`-S`	Places the daemon in standalone operation mode. The daemon runs in the background. This is useful for startup scripts that run during system initialization. See `init.d(4)`.
`-s`	Places the daemon in standalone operation mode. The daemon runs in the foreground. This is useful when run from `/etc/inittab` by `init(1M)`.
`-T` *maxtimeout*	Sets the maximum allowable timeout period to *maxtimeout* seconds. The default maximum timeout limit is 7200 second (two hours). You can override the `-T` option through use of the `ftpaccess(4)` file.
`-t` *timeout*	Sets the inactivity timeout period to *timeout* seconds. The default timeout period is 900 seconds (15 minutes). You can override the `-t` option through use of the `ftpaccess(4)` file.
`-u` *umask*	Sets the default `umask` to *umask*.
`-V`	Displays copyright and version information, then terminate.
`-v`	Writes debugging information to `syslogd(1M)`.
`-W`	Does not record user `login` and `logout` in the `wtmpx(4)` file.
`-w`	Records each user `login` and `logout` in the `wtmpx(4)` file. By default, logins and logouts are recorded.
`-X`	Writes the output from the `-i` and `-o` options to the `syslogd(1M)` file instead of `xferlog(4)`. This allows the collection of output from several hosts on one central loghost. You can override the `-X` option through use of the `ftpaccess(4)` file.

Requests The FTP Server currently supports the following FTP requests. Case is not distinguished.

`ABOR` Abort previous command.

`ADAT` Send an authentication protocol message.

`ALLO` Allocate storage (vacuously).

`AUTH` Specify an authentication protocol to be performed. Currently only "`GSSAPI`" is supported.

`APPE` Append to a file.

`CCC` Set the command channel protection mode to "`Clear`" (no protection). Not allowed if data channel is protected.

`CDUP` Change to parent of current working directory.

`CWD` Change working directory.

`DELE` Delete a file.

`ENC` Send a privacy and integrity protected command (given in argument).

`EPRT` Specify extended address for the transport connection.

`EPSV` Extended passive command request.

`HELP` Give help information.

`LIST` Give list files in a directory (`ls -lA`).

`LPRT` Specify long address for the transport connection.

`LPSV` Long passive command request.

`MIC` Send an integrity protected command (given in argument).

`MKD` Make a directory.

`MDTM` Show last time file modified.

`MODE` Specify data transfer *mode*.

`NLST` Give name list of files in directory (`ls`).

`NOOP` Do nothing.

`PASS` Specify password.

`PASV` Prepare for server-to-server transfer.

`PBSZ` Specify a protection buffer size.

`PROT` Specify a protection level under which to protect data transfers. Allowed arguments:

`clear` No protection.

`safe` Integrity protection

`private` Integrity and encryption protection

`PORT` Specify data connection port.

`PWD` Print the current working directory.

`QUIT` Terminate session.

`REST` Restart incomplete transfer.

`RETR` Retrieve a file.

`RMD` Remove a directory.

`RNFR` Specify rename-from file name.

`RNTO` Specify rename-to file name.

`SITE` Use nonstandard commands.

`SIZE` Return size of file.

`STAT` Return status of server.

`STOR` Store a file.

`STOU` Store a file with a unique name.

`STRU` Specify data transfer *structure*.

`SYST` Show operating system type of server system.

`TYPE` Specify data transfer `type`.

`USER` Specify user name.

`XCUP` Change to parent of current working directory. This request is deprecated.

`XCWD` Change working directory. This request is deprecated.

`XMKD` Make a directory. This request is deprecated.

`XPWD` Print the current working directory. This request is deprecated.

`XRMD` Remove a directory. This request is deprecated.

The following nonstandard or UNIX specific commands are supported by the `SITE` request:

`ALIAS` List aliases.

`CDPATH` List the search path used when changing directories.

`CHECKMETHOD` List or set the `checksum` method.

`CHECKSUM` Give the `checksum` of a file.

CHMOD	Change mode of a file. For example, SITE CHMOD 755 *filename*.
EXEC	Execute a program. For example, SITE EXEC program params
GPASS	Give special group access password. For example, SITE GPASS bar.
GROUP	Request special group access. For example, SITE GROUP foo.
GROUPS	List supplementary group membership.
HELP	Give help information. For example, SITE HELP.
IDLE	Set idle-timer. For example, SITE IDLE 60.
UMASK	Change umask. For example, SITE UMASK 002.

The remaining FTP requests specified in *RFC 959* are recognized, but not implemented.

The FTP server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in *RFC 959*. If a STAT command is received during a data transfer that has been preceded by a Telnet IP and Synch, transfer status will be returned.

in.ftpd interprets file names according to the "globbing" conventions used by csh(1). This allows users to utilize the metacharacters: * ? [ ] { } ~

in.ftpd authenticates users according to the following rules:

First, the user name must be in the password data base, the location of which is specified in nsswitch.conf(4). An encrypted password (an authentication token in PAM) must be present. A password must always be provided by the client before any file operations can be performed. For non-anonymous users, the PAM framework is used to verify that the correct password was entered. See SECURITY below.

Second, the user name must not appear in either the /etc/ftpusers or the /etc/ftpd/ftpusers file. Use of the /etc/ftpusers files is deprecated, although it is still supported.

Third, the users must have a standard shell returned by getusershell(3C).

Fourth, if the user name is anonymous or ftp, an anonymous ftp account must be present in the password file for user ftp. Use ftpconfig(1M) to create the anonymous ftp account and home directory tree.

Fifth, if the GSS-API is used to authenticate the user, then gss_auth_rules(5) determines user access without a password needed.

The FTP Server supports virtual hosting, which can be configured by using ftpaddhost(1M).

The FTP Server does not support sublogins.

General FTP Extensions

The FTP Server has certain extensions. If the user specifies a filename that does not exist with a RETR (retrieve) command, the FTP Server looks for a conversion to change a file or directory that does into the one requested. See ftpconversions(4).

By convention, anonymous users supply their email address when prompted for a password. The FTP Server attempts to validate these email addresses. A user whose FTP client hangs on a long reply, for example, a multiline response, should use a dash (-) as the first character of the user's password, as this disables the Server's `lreply()` function.

The FTP Server can also log all file transmission and reception. See xferlog(4) for details of the log file format.

The SITE EXEC command may be used to execute commands in the `/bin/ftp-exec` directory. Take care that you understand the security implications before copying any command into the `/bin/ftp-exec` directory. For example, do not copy in `/bin/sh`. This would enable the user to execute other commands through the use of `sh -c`. If you have doubts about this feature, do not create the `/bin/ftp-exec` directory.

Security

For non-anonymous users, `in.ftpd` uses pam(3PAM) for authentication, account management, and session management, and can use Kerberos v5 for authentication.

The PAM configuration policy, listed through `/etc/pam.conf`, specifies the module to be used for `in.ftpd`. Here is a partial `pam.conf` file with entries for the `in.ftpd` command using the UNIX authentication, account management, and session management module.

```
ftp   auth        requisite   pam_authtok_get.so.1
ftp   auth        required    pam_dhkeys.so.1
ftp   auth        required    pam_unix_auth.so.1

ftp   account     required    pam_unix_roles.so.1
ftp   account     required    pam_unix_projects.so.1
ftp   account     required    pam_unix_account.so.1

ftp   session     required    pam_unix_session.so.1
```

If there are no entries for the `ftp` service, then the entries for the "other" service will be used. Unlike `login`, `passwd`, and other commands, the `ftp` protocol will only support a single password. Using multiple modules will prevent `in.ftpd` from working properly.

To use Kerberos for authentication, a `host/`<*FQDN*> Kerberos principal must exist for each Fully Qualified Domain Name associated with the `in.ftpd` server. Each of these `host/`<*FQDN*> principals must have a `keytab` entry in the `/etc/krb5/krb5.keytab` file on the `in.ftpd` server. An example principal might be:

```
host/bigmachine.eng.example.com
```

See kadmin(1M) or gkadmin(1M) for instructions on adding a principal to a krb5.keytab file. See *System Administration Guide: Security Services* for a discussion of Kerberos authentication.

For anonymous users, who by convention supply their email address as a password, in.ftpd validates passwords according to the passwd-check capability in the ftpaccess file.

Usage The in.ftpd command is IPv6–enabled. See ip6(7P).

Files

/etc/ftpd/ftpaccess	FTP Server configuration file
/etc/ftpd/ftpconversions	FTP Server conversions database
/etc/ftpd/ftpgroups	FTP Server enhanced group access file
/etc/ftpd/ftphosts	FTP Server individual user host access file
/etc/ftpd/ftpservers	FTP Server virtual hosting configuration file.
/etc/ftpd/ftpusers	File listing users for whom FTP login privileges are disallowed.
/etc/ftpusers	File listing users for whom FTP login privileges are disallowed. This use of this file is deprecated.
/var/log/xferlog	FTP Server transfer log file
/var/run/ftp.pids-*classname*	
/var/adm/wtmpx	Extended database files that contain the history of user access and accounting information for the wtmpx database.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu
Interface Stability	External

See Also csh(1), ftp(1), ftpcount(1), ftpwho(1), ls(1), svcs(1), ftpaddhost(1M), ftpconfig(1M), ftprestart(1M), ftpshut(1M), gkadmin(1M), inetadm(1M), inetd(1M), kadmin(1M), svcadm(1M), syslogd(1M), chroot(2), umask(2), getpwent(3C), getusershell(3C), syslog(3C), ftpaccess(4), ftpconversions(4), ftpgroups(4), ftphosts(4), ftpservers(4), ftpusers(4), group(4), passwd(4), services(4), xferlog(4), wtmpx(4), attributes(5), gss_auth_rules(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), smf(5), ip6(7P)

System Administration Guide: Security Services

Allman, M., Ostermann, S., and Metz, C. *RFC 2428, FTP Extensions for IPv6 and NATs*. The Internet Society. September 1998.

Piscitello, D. *RFC 1639, FTP Operation Over Big Address Records (FOOBAR)*. Network Working Group. June 1994.

Postel, Jon, and Joyce Reynolds. *RFC 959, File Transfer Protocol (FTP)*. Network Information Center. October 1985.

St. Johns, Mike. *RFC 931, Authentication Server*. Network Working Group. January 1985.

Linn, J., *Generic Security Service Application Program Interface Version 2, Update 1, RFC 2743*. The Internet Society, January 2000.

Horowitz, M., Lunt, S., *FTP Security Extensions, RFC 2228*. The Internet Society, October 1997.

Diagnostics `in.ftpd` logs various errors to `syslogd(1M)`, with a facility code of daemon.

Notes The anonymous FTP account is inherently dangerous and should be avoided when possible.

The FTP Server must perform certain tasks as the superuser, for example, the creation of sockets with privileged port numbers. It maintains an effective user ID of the logged in user, reverting to the superuser only when necessary.

The FTP Server no longer supports the `/etc/default/ftpd` file. Instead of using `UMASK=nnn` to set the umask, use the `defumask` capability in the `ftpaccess` file. The banner greeting text capability is also now set through the `ftpaccess` file by using the greeting text capability instead of by using `BANNER="..."`. However, unlike the `BANNER` string, the greeting text string is not passed to the shell for evaluation. See `ftpaccess(4)`.

The `pam_unix`(5) module is no longer supported. Similar functionality is provided by `pam_authtok_check(5)`, `pam_authtok_get(5)`, `pam_authtok_store(5)`, `pam_dhkeys(5)`, `pam_passwd_auth(5)`, `pam_unix_account(5)`, `pam_unix_auth(5)`, and `pam_unix_session(5)`.

The `in.ftpd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/ftp
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name in.iked – daemon for the Internet Key Exchange (IKE)

Synopsis

```
/usr/lib/inet/in.iked [-d] [-f filename] [-p level]

/usr/lib/inet/in.iked -c [-f filename]
```

Description `in.iked` performs automated key management for IPsec using the Internet Key Exchange (IKE) protocol.

`in.iked` implements the following:

- IKE authentication with either pre-shared keys, DSS signatures, RSA signatures, or RSA encryption.
- Diffie-Hellman key derivation using either `768`, `1024`, or `1536`-bit public key moduli.
- Authentication protection with cipher choices of AES, DES, Blowfish, or 3DES, and hash choices of either `HMAC-MD5` or `HMAC-SHA-1`. Encryption in `in.iked` is limited to the IKE authentication and key exchange. See `ipsecesp(7P)` for information regarding IPsec protection choices.

`in.iked` is managed by the following `smf(5)` service:

```
svc:/network/ipsec/ike
```

This service is delivered disabled because the configuration file needs to be created before the service can be enabled. See `ike.config(4)` for the format of this file.

See "Service Management Facility" for information on managing the `smf(5)` service.

`in.iked` listens for incoming IKE requests from the network and for requests for outbound traffic using the `PF_KEY` socket. See `pf_key(7P)`.

`in.iked` has two support programs that are used for IKE administration and diagnosis: `ikeadm(1M)` and `ikecert(1M)`.

The `ikeadm(1M)` command can read the `/etc/inet/ike/config` file as a `rule`, then pass the configuration information to the running `in.iked` daemon using a doors interface.

```
example# ikeadm read rule /etc/inet/ike/config
```

Refreshing the `ike` `smf(5)` service provided to manage the `in.iked` daemon sends a `SIGHUP` signal to the `in.iked` daemon, which will (re)read `/etc/inet/ike/config` and reload the certificate database.

The preceding two commands have the same effect, that is, to update the running IKE daemon with the latest configuration. See "Service Management Facility" for more details on managing the `in.iked` daemon.

Service Management Facility

The IKE daemon (`in.iked`) is managed by the service management facility, `smf(5)`. The following group of services manage the components of IPsec:

```
svc:/network/ipsec/ipsecalgs   (See ipsecalgs(1M))
svc:/network/ipsec/policy      (See ipsecconf(1M))
svc:/network/ipsec/manual-key  (See ipseckey(1M))
svc:/network/ipsec/ike         (see ike.config(4))
```

The manual-key and `ike` services are delivered `disabled` because the system administrator must create configuration files for each service, as described in the respective man pages listed above.

The correct administrative procedure is to create the configuration file for each service, then enable each service using `svcadm(1M)`.

The `ike` service has a dependency on the `ipsecalgs` and `policy` services. These services should be enabled before the `ike` service. Failure to do so results in the `ike` service entering maintenance mode.

If the configuration needs to be changed, edit the configuration file then refresh the service, as follows:

```
example# svcadm refresh ike
```

The following properties are defined for the `ike` service:

`config/admin_privilege`
: Defines the level that `ikeadm(1M)` invocations can change or observe the running `in.iked`. The acceptable values for this property are the same as those for the `-p` option. See the description of `-p` in `OPTIONS`.

`config/config_file`
: Defines the configuration file to use. The default value is `/etc/inet/ike/config`. See `ike.config(4)` for the format of this file. This property has the same effect as the `-f` flag. See the description of `-f` in `OPTIONS`.

`config/debug_level`
: Defines the amount of debug output that is written to the `debug_logfile` file, described below. The default value for this is `op` or `operator`. This property controls the recording of information on events such as re-reading the configuration file. Acceptable value for `debug_level` are listed in the `ikeadm(1M)` man page. The value `all` is equivalent to the `-d` flag. See the description of `-d` in `OPTIONS`.

`config/debug_logfile`
: Defines where debug output should be written. The messages written here are from debug code within `in.iked`. Startup error messages are recorded by the `smf(5)` framework and recorded in a service-specific log file. Use any of the following commands to examine the `logfile` property:

```
example# svcs -l ike
example# svcprop ike
example# svccfg -s ike listprop
```

The values for these log file properties might be different, in which case both files should be inspected for errors.

`config/ignore_errors`
A boolean value that controls `in.iked`'s behavior should the configuration file have syntax errors. The default value is `false`, which causes `in.iked` to enter maintenance mode if the configuration is invalid.

Setting this value to `true` causes the IKE service to stay online, but correct operation requires the administrator to configure the running daemon with `ikeadm(1M)`. This option is provided for compatibility with previous releases.

These properties can be modified using `svccfg(1M)` by users who have been assigned the following authorization:

```
solaris.smf.value.ipsec
```

PKCS#11 token objects can be unlocked or locked by using `ikeadm` token login and `ikeadm` token logout, respectively. Availability of private keying material stored on these PKCS#11 token objects can be observed with: `ikeadm dump certcache`. The following authorizations allow users to log into and out of PKCS#11 token objects:

```
solaris.network.ipsec.ike.token.login
solaris.network.ipsec.ike.token.logout
```

See `auths(1)`, `ikeadm(1M)`, `user_attr(4)`, `rbac(5)`.

The service needs to be refreshed using `svcadm(1M)` before a new property value is effective. General, non-modifiable properties can be viewed with the `svcprop(1)` command.

```
# svccfg -s ipsec/ike setprop config/config_file = \
/new/config_file
# svcadm refresh ike
```

Administrative actions on this service, such as enabling, disabling, refreshing, and requesting restart can be performed using `svcadm(1M)`. A user who has been assigned the authorization shown below can perform these actions:

```
solaris.smf.manage.ipsec
```

The service's status can be queried using the `svcs(1)` command.

The `in.iked` daemon is designed to be run under `smf(5)` management. While the `in.iked` command can be run from the command line, this is discouraged. If the `in.iked` command is to be run from the command line, the `ike` `smf(5)` service should be disabled first. See `svcadm(1M)`.

Options The following options are supported:

`-c` Check the syntax of a configuration file.

`-d` Use debug mode. The process stays attached to the controlling terminal and produces large amounts of debugging output. This option is deprecated. See "Service Management Facility" for more details.

`-f` *filename* Use *filename* instead of `/etc/inet/ike/config`. See `ike.config(4)` for the format of this file. This option is deprecated. See "Service Management Facility" for more details.

`-p` *level* Specify privilege level (*level*). This option sets how much `ikeadm(1M)` invocations can change or observe about the running `in.iked`.

Valid *levels* are:

0 Base level

1 Access to preshared key info

2 Access to keying material

If `-p` is not specified, *level* defaults to `0`.

This option is deprecated. See "Service Management Facility" for more details.

Security This program has sensitive private keying information in its image. Care should be taken with any core dumps or system dumps of a running `in.iked` daemon, as these files contain sensitive keying information. Use the `coreadm(1M)` command to limit any corefiles produced by `in.iked`.

Files

`/etc/inet/ike/config`	Default configuration file.
`/etc/inet/secret/ike.privatekeys/*`	Private keys. A private key *must* have a matching public-key certificate with the same filename in `/etc/inet/ike/publickeys/`.
`/etc/inet/ike/publickeys/*`	Public-key certificates. The names are only important with regard to matching private key names.
`/etc/inet/ike/crls/*`	Public key certificate revocation lists.
`/etc/inet/secret/ike.preshared`	IKE pre-shared secrets for Phase I authentication.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also svcs(1), coreadm(1M), ikeadm(1M), ikecert(1M), svccfg(1M), svcadm(1M), ike.config(4), attributes(5), smf(5), ipsecesp(7P), pf_key(7P)

Harkins, Dan and Carrel, Dave. *RFC 2409, Internet Key Exchange (IKE)*. Network Working Group. November 1998.

Maughan, Douglas, Schertler, M., Schneider, M., Turner, J. *RFC 2408, Internet Security Association and Key Management Protocol (ISAKMP)*. Network Working Group. November 1998.

Piper, Derrell, *RFC 2407, The Internet IP Security Domain of Interpretation for ISAKMP*. Network Working Group. November 1998.

Name init, telinit – process control initialization

Synopsis

```
/sbin/init [0123456abcQqSs]

/etc/telinit [0123456abcQqSs]
```

Description `init` is the default primordial user process. (Options given to the kernel during boot may result in the invocation of an alternative primordial user process, as described on `kernel(1M)`). `init` initiates the core components of the service management facility, `svc.configd(1M)` and `svc.startd(1M)`, and restarts these components if they fail. For backwards compatibility, `init` also starts and restarts general processes according to `/etc/inittab`, as desribed below.

The run levels and system booting descriptions given below are provided for compatibility purposes only, and otherwise made obsolete by the service management facility, `smf(5)`.

init Failure If `init` exits for any reason other than system shutdown, it will be restarted with process-ID 1.

Run Level Defined At any given time, the system is in one of eight possible run levels. A run level is a software configuration under which only a selected group of processes exists. Processes spawned by `init` for each of these run levels are defined in `/etc/inittab`. `init` can be in one of eight run levels, `0`–`6` and `S` or `s` (`S` and `s` are identical). The run level changes when a privileged user runs `/sbin/init`.

init and System Booting When the system is booted, `init` is invoked and the following occurs. First, it reads `/etc/default/init` to set environment variables. This is typically where `TZ` (time zone) and locale-related environments such as `LANG` or `LC_CTYPE` get set. (See the FILES section at the end of this page.) `init` then looks in `/etc/inittab` for the `initdefault` entry (see `inittab(4)`). If the `initdefault` entry:

exists — `init` usually uses the run level specified in that entry as the initial run level to enter only if the options/milestone property has not been specified for `svc.startd(1M)`.

does not exist — The service management facility, `smf(5)`, examines its configuration specified in `svc.startd(1M)`, and enters the milestone specified by the options/milestone property.

The `initdefault` entry in `/etc/inittab` corresponds to the following run levels:

`S` or `s` — `init` goes to the single-user state. In this state, the system console device (`/dev/console`) is opened for reading and writing and the command `/sbin/su`, (see `su(1M)`), is invoked. Use either `init` or `telinit` to change the run level of the system. Note that if the shell is terminated (using an end-of-file), `init` only re-initializes to the single-user state if `/etc/inittab` does not exist.

`0-6` — `init` enters the corresponding run level. Run levels `0`, `5`, and `6` are reserved states for shutting the system down. Run levels `2`, `3`, and `4` are available as multi-user operating states.

If this is the first time since power up that `init` has entered a run level other than single-user state, `init` first scans `/etc/inittab` for `boot` and `bootwait` entries (see `inittab(4)`). These entries are performed before any other processing of `/etc/inittab` takes place, providing that the run level entered matches that of the entry. In this way any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. `init` then scans `/etc/inittab` and executes all other entries that are to be processed for that run level.

To spawn each process in `/etc/inittab`, `init` reads each entry and for each entry that should be respawned, it forks a child process. After it has spawned all of the processes specified by `/etc/inittab`, `init` waits for one of its descendant processes to die, a `powerfail` signal, or a signal from another `init` or `telinit` process to change the system's run level. When one of these conditions occurs, `init` re-examines `/etc/inittab`.

inittab Additions

New entries can be added to `/etc/inittab` at any time; however, `init` still waits for one of the above three conditions to occur before re-examining `/etc/inittab`. To get around this, `init Q` or `init q` command wakes `init` to re-examine `/etc/inittab` immediately.

When `init` comes up at boot time and whenever the system changes from the single-user state to another run state, `init` sets the `ioctl(2)` states of the console to those modes saved in the file `/etc/ioctl.syscon`. `init` writes this file whenever the single-user state is entered.

Run Level Changes

When a run level change request is made, `init` or a designate sends the warning signal (`SIGTERM`) to all processes that are undefined in the target run level. A minimum interval of five seconds is observed before `init` or its designate forcibly terminates these processes by sending a kill signal (`SIGKILL`). Additionally, init informs `svc.startd(1M)` that the run level is changing. `svc.startd(1M)` then restricts the system to the set of services which the milestone corresponding to the run-level change depends on.

When `init` receives a signal telling it that a process it spawned has died, it records the fact and the reason it died in `/var/adm/utmpx` and `/var/adm/wtmpx` if it exists (see `who(1)`). A history of the processes spawned is kept in `/var/adm/wtmpx`.

If `init` receives a `powerfail` signal (`SIGPWR`) it scans `/etc/inittab` for special entries of the type `powerfail` and `powerwait`. These entries are invoked (if the run levels permit) before any further processing takes place. In this way `init` can perform various cleanup and recording functions during the powerdown of the operating system.

Environment Variables in /etc/default/init

You can set default values for environment variables, for such items as timezone and character formatting, in `/etc/default/init`. See the FILES section, below, for a list of these variables.

telinit

`telinit`, which is linked to `/sbin/init`, is used to direct the actions of `init`. It takes a one-character argument and signals `init` to take the appropriate action.

Security `init` uses pam(3PAM) for session management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the session management module to be used for `init`. Here is a partial `pam.conf` file with entries for `init` using the UNIX session management module.

```
init    session    required     pam_unix_session.so.1
```

If there are no entries for the `init` service, then the entries for the "other" service will be used.

Options

`0` Go into firmware.

`1` Put the system in system administrator mode. All local file systems are mounted. Only a small set of essential kernel processes are left running. This mode is for administrative tasks such as installing optional utility packages. All files are accessible and no users are logged in on the system.

This request corresponds to a request for `smf(5)` to restrict the system milestone to svc:/milestone/single-user:default.

`2` Put the system in multi-user mode. All multi-user environment terminal processes and daemons are spawned. This state is commonly referred to as the multi-user state.

This request corresponds to a request for `smf(5)` to restrict the system milestone to svc:/milestone/multi-user:default.

`3` Extend multi-user mode by making local resources available over the network.

This request corresponds to a request for `smf(5)` to restrict the system milestone to svc:/milestone/multi-user-server:default.

`4` Is available to be defined as an alternative multi-user environment configuration. It is not necessary for system operation and is usually not used.

`5` Shut the machine down so that it is safe to remove the power. Have the machine remove power, if possible.

`6` Stop the operating system and reboot to the state defined by the `initdefault` entry in `/etc/inittab`.

`a,b,c` Process only those `/etc/inittab` entries having the `a`, `b`, or `c` run level set. These are pseudo-states, which may be defined to run certain commands, but which do not cause the current run level to change.

`Q,q` Re-examine `/etc/inittab`.

`S, s` Enter single-user mode. This is the only run level that doesn't require the existence of a properly formatted `/etc/inittab` file. If this file does not exist, then by default, the only legal run level that `init` can enter is the single-user mode. When in single-user mode, the filesystems required for basic system operation will be mounted. When the system comes down to single-user mode, these file systems will remain mounted (even if provided by a remote file server), and any other local filesystems will also be

left mounted. During the transition down to single-user mode, all processes started by `init` or `init.d` scripts that should only be running in multi-user mode are killed. In addition, any process that has a `utmpx` entry will be killed. This last condition insures that all port monitors started by the SAC are killed and all services started by these port monitors, including `ttymon` login services, are killed.

This request corresponds to a request for `smf(5)` to restrict the system milestone to svc:/milestone/single-user:default.

Files

`/dev/console`	System console device
`/etc/default/init`	Contains environment variables and their default values. For example, for the timezone variable, `TZ`, you might specify `TZ=US/Pacific`. The variables are:
`TZ`	Either specifies the timezone information (see `ctime(3C)`) or the name of a timezone information file `/usr/share/lib/zoneinfo`. Refer to the `TIMEZONE(4)` man page before changing this setting.
`CMASK`	The mask (see `umask(1)`) that `init` uses and that every process inherits from the `init` process. If not set, `init` uses the mask it inherits from the kernel. Note that `init` always attempts to apply a `umask` of 022 before creating a file, regardless of the setting of `CMASK`
`LC_CTYPE`	Character characterization information
`LC_MESSAGES`	Message translation
`LC_MONETARY`	Monetary formatting information
`LC_NUMERIC`	Numeric formatting information
`LC_TIME`	Time formatting information
`LC_ALL`	If set, all other `LC_*` environmental variables take-on this value.
`LANG`	If `LC_ALL` is not set, and any particular `LC_*` is also not set, the value of `LANG` is used for that particular environmental variable.
/etc/initpipe	A named pipe used for internal communication
`/etc/inittab`	Controls process dispatching by `init`

/etc/ioctl.syscon	ioctl states of the console, as saved by init when single-user state is entered
/var/adm/utmpx	User access and administration information
/var/adm/wtmpx	History of user access and administration information
/var/run/init.state	init state necessary to recover from failure.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also login(1), sh(1), stty(1), who(1), kernel(1M), shutdown(1M), su(1M), svc.configd(1M), svc.startd(1M), ttymon(1M), ioctl(2), kill(2), ctime(3C), pam(3PAM), init.d(4), inittab(4), pam.conf(4), TIMEZONE(4), utmpx(4), attributes(5), pam_unix_session(5), smf(5), termio(7I)

Diagnostics If init finds that it is respawning an entry from /etc/inittab more than ten times in two minutes, it assumes that there is an error in the command string in the entry and generates an error message on the system console. It then refuses to respawn this entry until either five minutes has elapsed or it receives a signal from a user-spawned init or telinit command. This prevents init from eating up system resources when someone makes a typographical error in the inittab file, or a program is removed that is referenced in /etc/inittab.

Notes init and telinit can be run only by a privileged user.

The S or s state must not be used indiscriminately in /etc/inittab. When modifying this file, it is best to avoid adding this state to any line other than initdefault.

If a default state is not specified in the initdefault entry in /etc/inittab, state 6 is entered. Consequently, the system will loop by going to firmware and rebooting continuously.

If the utmpx file cannot be created when booting the system, the system will boot to state "s" regardless of the state specified in the initdefault entry in /etc/inittab. This can occur if the /var file system is not accessible.

When a system transitions down to the S or s state, the /etc/nologin file (see nologin(4)) is created. Upon subsequent transition to run level 2, this file is removed.

init uses /etc/initpipe, a named pipe, for internal communication.

The pam_unix(5) module is no longer supported. Similar functionality is provided by pam_unix_session(5).

Name init.sma – start and stop the snmpd daemon

Synopsis

```
/etc/init.d/init.sma start | stop | restart | status
```

Description The `init.sma` utility is run automatically during installation and each time the system is rebooted. This utility manages the `snmpd`. See `snmpd(1M)`.

Options The following options are supported:

`start` Starts the `snmpd` daemon.

`stop` Stops the `snmpd` daemon.

`restart` Stops then starts the `snmpd` daemon.

`status` Reports the `snmpd` daemon's status.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmagt
Interface Stability	Unstable

See Also `snmpd(1M)`, `attributes(5)`

Name init.wbem – start and stop the CIM Boot Manager

Synopsis

```
svccfg svc:/application/management/wbem start | stop
```

Description The `init.wbem` utility is run automatically during installation, each time the system is rebooted, and by means of the `svccfg(1M)` command. This method manipulates the CIM Object Manager (CIMOM) and the Solaris Management Console server, both of which run combined in a single process. `init.wbem` can be used to start, stop, or retrieve status from the server.

CIM Object Manager

The CIM Object Manager manages CIM objects on a WBEM-enabled system. A CIM object is a computer representation, or model, of a managed resource, such as a printer, disk drive, or CPU. CIM objects are stored internally as Java classes.

When a WBEM client application accesses information about a CIM object, the CIM Object Manager contacts either the appropriate provider for that object or the CIM Object Manager Repository. Providers are classes that communicate with managed objects to access data.

When a WBEM client application requests data from a managed resource that is not available from the CIM Object Manager Repository, the CIM Object Manager forwards the request to the provider for that managed resource. The provider dynamically retrieves the information.

At startup, the CIM Object Manager performs the following functions:

- Listens for RMI connections on RMI port 5987 and for XML/HTTP connections on HTTP port 5988.
- Sets up a connection to the CIM Object Manager Repository.
- Waits for incoming requests.

During normal operations, the CIM Object Manager performs the following functions:

- Performs security checks to authenticate user login and authorization to access namespaces.
- Performs syntactical and semantic checking of CIM data operations to ensure that they comply with the latest CIM Specification.
- Routes requests to the appropriate provider or to the CIM Object Manager Repository.
- Delivers data from providers and from the CIM Object Manager Repository to WBEM client applications.

A WBEM client application contacts the CIM Object Manager to establish a connection when it needs to perform WBEM operations, such as creating a CIM class or updating a CIM instance. When a WBEM client application connects to a CIM Object Manager, it gets a reference to the CIM Object Manager, which it then uses to request services and operations.

Solaris Management Console Server

The Solaris Management Console server is the back end to the front end console, smc(1M). It provides tools for the console to download and performs common services for the console and its tools to use, such as authentication, authorization, logging, messaging, and persistence.

System Booting

The `init.wbem` script is installed in the `/etc/init.d` directory.

Options

The `init.wbem` function is implemented as a service management facility (SMF) method and is controlled by means of the svcadm(1M) command using the fault management resource identifier (FMRI) `svc:/application/management/wbem`.

The following options are supported through svcadm(1M):

`start` Starts the CIMOM and Solaris Management Console server on the local host.

`stop` Stops the CIMOM and Solaris Management Console server on the local host.

`status` Gets the status of the CIMOM and Solaris Management Console server on the local host.

The SMF property `options/tcp_listen` is used to control whether CIMOM and the Solaris Management console server will respond to requests from remote systems.

The specification:

```
svc:/application/management/wbem/options/tcp_listen = true
```

...allows remote access and `false` disallows remote access. `false` is the default value.

Examples

EXAMPLE 1 Allowing Access to Remote Systems

The following commands enable CIMOM and the Solaris Management Console server to allow access from remote systems.

```
# svccfg -s \
svc:/application/management/wbem setprop options/tcp_listen = true
# svcadm refresh svc:/application/management/wbem
```

Notes

When the `init.wbem` script is invoked by SMF, it does not run the CIMOM and Solaris Management Console server directly. The server process is in Java and is too heavyweight to be run immediately at system boot time. Instead, three lightweight processes listen on three different ports that the CIMOM and the Solaris Management Console server normally use. This acts similarly to inetd(1M).

Because Java programs cannot inherit file descriptors as other programs can, there is a small time period from when the first connection is made until the server is fully operational where client connections may be dropped. WBEM clients are immune to this, as they will retry until the server comes online. Solaris Management Console clients are not immune, and it may be necessary to manually reconnect, though this should not happen in the common case.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbcor

See Also inetd(1M), mofcomp(1M), smc(1M), smcconf(1M), svccfg(1M), svcadm(1M), wbemadmin(1M), wbemlogviewer(1M), attributes(5), smf(5)

Name inityp2l – create NIS (YP) to LDAP configuration files

Synopsis

```
/usr/lib/netsvc/yp/inityp2l [-m mapping_file_name]
     [-c config_file_name]
```

Description The `inityp2l` utility assists with creation of the `NISLDAPmapping` and `ypserv` files. See `NISLDAPmapping(4)` and`ypserv(4)`. `inityp2l` examines the NIS maps on a system. and through a dialogue with the user, determines which NIS to (and from) LDAP mappings are required. A `NISLDAPmapping` file is then created based on this information. The utility asks users about their LDAP server configuration and a `ypserv` file is created based on this information.

The `inityp2l` utility handles mappings for standard NIS maps and the `auto.*` series of maps. If requested, it creates default mappings for custom maps, with each map entry represented as a single DIT string. `inityp2l` does not handle full custom mapping, but if requested, `inityp2l` will insert comments into the `NISLDAPmapping` file that indicate where these should be added.

To write to the `NISLDAPmapping` or `ypserv` files is potentially dangerous. `inityp2l` warns the user and asks for confirmation before:

1. it overwrites either file
2. it writes to the default `NISLDAPmapping` file location, if this file did not previously exist. This is important because the existence of a file in this location causes NIS components to work NIS to LDAP (N2L) mode when next restarted, rather than to traditional NIS mode.

`inityp2l` assists with rapid creation of a simple N2L configuration files. It is not a general purpose tool for the management of these files. An advanced user who would like to maintain the files or use custom mappings should examine the output of `inityp2l` and customize it by using a standard text editor.

Options `inityp2l` supports the following options:

`-c` Specify the name of the generated `ypserv` file. The default location is described in Files.

`-m` Specify the name of the generated `NISLDAPmapping` file. The default is described in Files.

Files

`/var/yp`	The directory to be searched for candidate domains (`/var/yp/*`) and NIS maps (`/var/yp/*/*`)
`/var/yp/NISLDAPmapping`	The default location for the generated `NISLDAPmapping` file
`/etc/default/ypserv`	The default location for the generated `ypserv` file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu
Interface Stability	Obsolete

See Also NISLDAPmapping(4), ypserv(4), attributes(5)

Name in.lpd – BSD print protocol adaptor

Synopsis

```
/usr/lib/print/in.lpd
```

Description `in.lpd` implements the network listening service for the BSD print protocol specified in RFC 1179. The BSD print protocol provides a remote interface for systems to interact with a local spooling system. The protocol defines five standard requests from the client to the server: starting queue processing, transferring print jobs, retrieving terse status, retrieving verbose status, and canceling print jobs.

The `in.lpd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/lp
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

`in.lpd` uses the `config/log_from_remote` property to allow or disallow remote access. The default value of this property, `localhost`, disallows remote access.

`inetd` waits for connections on TCP port 515. Upon receipt of a connect request, `in.lpd` is started to service the connection. Once the request has been filled, `in.lpd` closes the connection and exits.

Examples **EXAMPLE 1** Allowing Remote Access

The following command allows remote access to `in.lpd`.

```
# inetadm -m svc:/application/print/rfc1179:default bind_addr=""
```

Exit Status The following exit values are returned:

`0`	Successful completion.
non-zero	An error occurred.

Files

`/etc/printers.conf`	System printer configuration database.
`printers.conf.byname`	NIS version of `/etc/printers.conf`.
`printers.org_dir`	NIS+ version of `/etc/printers.conf`.
`/usr/lib/print/bsd-adaptor/bsd_*.so*`	Spooler translation modules.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

See Also svcs(1), inetadm(1M), inetd(1M), svcadm(1M), printers.conf(4), attributes(5), smf(5)

Name in.mpathd – daemon for network adapter (NIC) failure detection, recovery, automatic failover and failback

Synopsis `/usr/lib/inet/in.mpathd`

Description The `in.mpathd` daemon performs Network Interface Card (NIC) failure and repair detection. In the event of a NIC failure, it causes IP network access from the failed NIC to failover to a standby NIC, if available, or to any another operational NIC that has been configured as part of the same network multipathing group. Once the failed NIC is repaired, all network access is restored to the repaired NIC.

The `in.mpathd` daemon can detect NIC failure and repair through two methods: by monitoring the IFF_RUNNING flag for each NIC (link-based failure detection), and by sending and receiving ICMP echo requests and replies on each NIC (probe-based failure detection). Link-based failure detection requires no explicit configuration and thus is always enabled (provided the NIC driver supports the feature); probe-based failure detection must be enabled through the configuration of one or more test addresses (described below), but has the benefit of testing the entire NIC send and receive path.

If only link-based failure detection is enabled, then the health of the interface is determined solely from the state of the `IFF_RUNNING` flag. Otherwise, the interface is considered failed if either of the two methods indicate a failure, and repaired once both methods indicate the failure has been corrected. Not all interfaces in a group need to be configured with the same failure detection methods.

As mentioned above, in order to perform probe-based failure detection `in.mpathd` needs a special test address on each NIC for the purpose of sending and receiving probes on the NIC. Use the `ifconfig` command `-failover` option to configure these test addresses. See `ifconfig(1M)`. The test address must belong to a subnet that is known to the hosts and routers on the link.

The `in.mpathd` daemon can detect NIC failure and repair by two methods, by sending and receiving ICMP echo requests and replies on each NIC, and by monitoring the `IFF_RUNNING` flag for each NIC. The link state on some models of NIC is indicated by the `IFF_RUNNING` flag, allowing for faster failure detection when the link goes down. The `in.mpathd` daemon considers a NIC to have failed if either of the above two methods indicates failure. A NIC is considered to be repaired only if both methods indicate the NIC is repaired.

The `in.mpathd` daemon sends the ICMP echo request probes to on-link routers. If no routers are available, it sends the probes to neighboring hosts. Thus, for network failure detection and repair, there must be at least one neighbor on each link that responds to ICMP echo request probes.

`in.mpathd` works on both IPv4 and IPv6. If IPv4 is plumbed on a NIC, an IPv4 test address is configured on theNIC, and the NIC is configured as part of a network multipathing group, then `in.mpathd` will start sending ICMP probes on the NIC using IPv4.

In the case of IPv6, the link-local address must be configured as the test address. The `in.mpathd` daemon will not accept a non-link-local address as a test address. If the NIC is part of a multipathing group, and the test address has been configured, then `in.mpathd` will probe the NIC for failures using IPv6.

Even if both the IPv4 and IPv6 protocol streams are plumbed, it is sufficient to configure only one of the two, that is, either an IPv4 test address or an IPv6 test address on a NIC. If only an IPv4 test address is configured, it probes using only ICMPv4. If only an IPv6 test address is configured, it probes using only ICMPv6. If both type test addresses are configured, it probes using both ICMPv4 and ICMPv6.

The `in.mpathd` daemon accesses three variable values in `/etc/default/mpathd`: *FAILURE_DETECTION_TIME*, *FAILBACK* and *TRACK_INTERFACES_ONLY_WITH_GROUPS*.

The *FAILURE_DETECTION_TIME* variable specifies the `NIC` failure detection time for the ICMP echo request probe method of detecting NIC failure. The shorter the failure detection time, the greater the volume of probe traffic. The default value of *FAILURE_DETECTION_TIME* is 10 seconds. This means that NIC failure will be detected by `in.mpathd` within 10 seconds. NIC failures detected by the `IFF_RUNNING` flag being cleared are acted on as soon as the `in.mpathd` daemon notices the change in the flag. The NIC repair detection time cannot be configured; however, it is defined as double the value of *FAILURE_DETECTION_TIME*.

By default, `in.mpathd` does failure detection only on NICs that are configured as part of a multipathing group. You can set *TRACK_INTERFACES_ONLY_WITH_GROUPS* to `no` to enable failure detection by `in.mpathd` on all NICs, even if they are not part of a multipathing group. However, `in.mpathd` cannot do failover from a failed NIC if it is not part of a multipathing group.

The `in.mpathd` daemon will restore network traffic back to the previously failed NIC, after it has detected a NIC repair. To disable this, set the value of *FAILBACK* to `no` in `/etc/default/mpathd`.

Files `/etc/default/mpathd` Contains default values used by the `in.mpathd` daemon.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

See Also `ifconfig(1M)`, `attributes(5)`, `icmp(7P)`, `icmp6(7P)`,

System Administration Guide: IP Services

Diagnostics `Test address` *address* `is not unique; disabling probe based failure detection on` *interface_name*

Description: For `in.mpathd` to perform probe-based failure detection, each test address in the group must be unique. Since the IPv6 test address is a link-local address derived from the MAC address, each IP interface in the group must have a unique MAC address.

`NIC` *interface_name* `of group` *group_name* `is not plumbed for IPv[4|6] and may affect failover capability`

Description: All NICs in a multipathing group must be homogeneously plumbed. For example, if a NIC is plumbed for IPv4, then all NICs in the group must be plumbed for IPv4. The streams modules pushed on all NICs must be identical.

`No test address configured on interface` *interface_name* `disabling probe-based failure detection on it`

Description: In order for `in.mpathd` to perform probe-based failure detection on a NIC, it must be configured with a test address: IPv4, IPv6, or both.

`The link has come up on` *interface_name* `more than 2 times in the last minute; disabling failback until it stabilizes.`

Description: In order to prevent interfaces with intermittent hardware, such as a bad cable, from causing repeated failovers and failbacks, `in.mpathd` does not failback to interfaces with frequently fluctuating link states.

`Invalid failure detection` *time* `assuming default 10000`

Description: An invalid value was encountered for *FAILURE_DETECTION_TIME* in the `/etc/default/mpathd` file.

`Too small failure detection time of` *time* `assuming minimum 100`

Description: The minimum value that can be specified for *FAILURE_DETECTION_TIME* is currently 100 milliseconds.

`Invalid value for FAILBACK` *value*

Description: Valid values for the boolean variable *FAILBACK* are `yes` or `no`.

`Invalid value for TRACK_INTERFACES_ONLY_WITH_GROUPS` *value*

Description: Valid values for the boolean variable *TRACK_INTERFACES_ONLY_WITH_GROUPS* are `yes` or `no`.

`Cannot meet requested failure detection time of` *time* `ms on (inet[6]` *interface_name*`) new failure detection time for group` *group_name* `is` *time* `ms`

Description: The round trip time for ICMP probes is higher than necessary to maintain the current failure detection time. The network is probably congested or the probe targets are loaded. `in.mpathd` automatically increases the failure detection time to whatever it can achieve under these conditions.

`Improved failure detection time` *time* `ms on (inet[6]` *interface_name*`) for group` *group_name*

Description: The round trip time for ICMP probes has now decreased and `in.mpathd` has lowered the failure detection time correspondingly.

`NIC failure detected on` *interface_name*

Description: `in.mpathd` has detected NIC failure on *interface_name*, and has set the `IFF_FAILED` flag on NIC *interface_name*.

`Successfully failed over from NIC` *interface_name1* `to NIC` *interface_name2*

Description: `in.mpathd` has caused the network traffic to failover from NIC *interface_name1* to NIC *interface_name2*, which is part of the multipathing group.

`NIC repair detected on` *interface_name*

Description: `in.mpathd` has detected that NIC *interface_name* is repaired and operational. If the `IFF_FAILED` flag on the NIC was previously set, it will be reset.

`Successfully failed back to NIC` *interface_name*

Description: `in.mpathd` has restored network traffic back to NIC *interface_name*, which is now repaired and operational.

`The link has gone down on` *interface_name*

Description: `in.mpathd` has detected that the `IFF_RUNNING` flag for NIC *interface_name* has been cleared, indicating the link has gone down.

`The link has come up on` *interface_name*

Description: `in.mpathd` has detected that the `IFF_RUNNING` flag for NIC *interface_name* has been set, indicating the link has come up.

Name in.ndpd – daemon for IPv6 autoconfiguration

Synopsis

```
/usr/lib/inet/in.ndpd  [-adt] [-f config_file]
```

Description `in.ndpd` provides both the host and router autoconfiguration components of Neighbor Discovery for IPv6 and Stateless Address Autoconfiguration for IPv6. In particular, `in.ndpd` implements

- router discovery;
- prefix discovery;
- parameter discovery;
- address autoconfiguration; and
- privacy extensions for stateless address autoconfiguration.

Other aspects of Neighbor Discovery are implemented by `ip6(7P)`, including:

- address resolution;
- neighbor unreachability detection; and
- redirect.

The duplicate address detection function is implemented by `ifconfig(1M)`.

If the `/etc/inet/ndpd.conf` file does not exist or does not set the variable `AdvSendAdvertisements` to true for a network interface, then `in.ndpd` will make the node a host for that interface, that is, sending router solicitation messages and then using router advertisement messages it receives to autoconfigure the node. Note that `in.ndpd` only autoconfigures the addresses of global or site-local scope from the prefix advertisement.

If `AdvSendAdvertisements` is set to true for an interface, then `in.ndpd` will perform router functions on that interface, that is, sending router advertisement messages to autoconfigure the attached hosts, but not use any advertisements it receives for autoconfiguration. However, when sending advertisements, `in.ndpd` will use the advertisements it sends itself to autoconfigure its prefixes.

Stateless autoconfiguration requires no manual configuration of hosts, minimal (if any) configuration of routers, and no additional servers. The stateless mechanism enables a host to generate its own addresses and uses local information as well as non-local information that is advertised by routers to generate the addresses.

Temporary addresses that are autoconfigured for an interface can also be implemented. A temporary address token is enabled for one or more interfaces on a host. However, unlike standard, autoconfigured IPv6 addresses, a temporary address consists of the site prefix and a randomly generated 64 bit number. This random number becomes the interface ID segment of the IPv6 address. A link-local address is not generated with the temporary address as the interface ID.

Routers advertise all prefixes that have been assigned on the link. IPv6 hosts use Neighbor Discovery to obtain a subnet prefix from a local router. Hosts automatically create IPv6

addresses by combining the subnet prefix with an interface IDs that is generated from an interface's MAC address. In the absence of routers, a host can generate only link-local addresses. Link-local addresses can only be used for communication with nodes on the same link.

For information on how to enable IPv6 address autoconfiguration, see *System Administration Guide: IP Services*

Options

`-a` Turn off stateless address auto configuration. When set, the daemon does not autoconfigure any addresses and does not renumber any addresses. This option does the same thing as the following line in `ndpd.conf(4)`:

```
ifdefault StatelessAddrConf off
```

`-d` Turn on large amounts of debugging output on `stdout`. When set, the program runs in the foreground and stays attached to the controlling terminal.

`-f` *config_file* Use *config_file* for configuration information instead of the default `/etc/inet/ndpd.conf`.

`-t` Turn on tracing (printing) of all sent and received packets to`stdout`. When set, the program runs in the foreground and stays attached to the controlling terminal.

Files

`/etc/inet/ndpd.conf` Configuration file. This file is not necessary on a host, but it is required on a router to enable `in.ndpd` to advertise autoconfiguration information to the hosts.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`ifconfig(1M)`, `ndpd.conf(4)`, `attributes(5)`, `icmp6(7P)`,`ip6(7P)`, `attributes(5)`

System Administration Guide: IP Services

Narten, T., Nordmark, E., Simpson, W.*RFC 2461, Neighbor Discovery for IP Version 6 (IPv6)*. The Internet Society. December 1998.

Thomson, S., Narten, T. *RFC 2462, IPv6 Stateless Address Autoconfiguration*. The Internet Society. December 1998.

Narten, T., and Draves, R. RFC 3041, Privacy Extensions for Stateless Address Autoconfiguration in IPv6. The Internet Society. January 2001.

Diagnostics Receipt of a `SIGHUP` signal will make `in.ndpd` restart and reread `/etc/inet/ndpd.conf`.

Name in.rarpd, rarpd – DARPA Reverse Address Resolution Protocol server

Synopsis

```
/usr/sbin/in.rarpd [-d] -a

/usr/sbin/in.rarpd [-d] device unit
```

Description `in.rarpd` starts a daemon that responds to Reverse Address Resolution Protocol (RARP) requests. The daemon forks a copy of itself that runs in background. It must be run as root.

RARP is used by machines at boot time to discover their Internet Protocol (IP) address. The booting machine provides its Ethernet address in a RARP request message. Using the `ethers` and `hosts` databases, `in.rarpd` maps this Ethernet address into the corresponding IP address which it returns to the booting machine in an RARP reply message. The booting machine must be listed in both databases for `in.rarpd` to locate its IP address. `in.rarpd` issues no reply when it fails to locate an IP address.

`in.rarpd` uses the STREAMS-based Data Link Provider Interface (DLPI) message set to communicate directly with the datalink device driver.

Options The following options are supported:

`-a` Get the list of available network interfaces from IP using the SIOCGIFADDR ioctl and start a RARP daemon process on each interface returned.

`-d` Print assorted debugging messages while executing.

Examples

EXAMPLE 1 Starting An `in.rarpd` Daemon For Each Network Interface Name Returned From `/dev/ip`:

The following command starts an `in.rarpd` for each network interface name returned from `/dev/ip`:

```
example# /usr/sbin/in.rarpd -a
```

EXAMPLE 2 Starting An `in.rarpd` Daemon On The Device `/dev/le` With The Device Instance Number `0`

The following command starts one `in.rarpd` on the device `/dev/le` with the device instance number `0`.

```
example# /usr/sbin/in.rarpd le 0
```

Files

`/etc/ethers` File or other source, as specified by `nsswitch.conf(4)`.

`/etc/hosts` File or other source, as specified by `nsswitch.conf(4)`.

`/tftpboot`

`/dev/ip`

`/dev/arp`

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbsu

See Also svcs(1), boot(1M), ifconfig(1M), svcadm(1M), ethers(4), hosts(4), netconfig(4), nsswitch.conf(4),attributes(5), smf(5), dlpi(7P)

Finlayson, R., Mann, T., Mogul, J., and Theimer, M., *RFC 903, A Reverse Address Resolution Protocol*, Network Information Center, SRI International, June 1984.

Notes The `in.rarpd` service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/rarp
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name in.rdisc, rdisc – network router discovery daemon

Synopsis

```
/usr/sbin/in.rdisc  [-a] [-f] [-s] [send-address] [receive-address]

/usr/sbin/in.rdisc  -r [-p preference] [-T interval]
 [send-address] [receive-address]
```

Description `in.rdisc` remains part of the software distribution of the Solaris Operating Environment. It is, however, not used by default. `in.routed(1M)` includes the functionality provided by `in.rdisc`. See `routeadm(1M)` for details of how to specify the IPV4 routing daemon.

`in.rdisc` implements the ICMP router discovery protocol. The first form of the command is used on hosts and the second form is used on routers.

`in.rdisc` can be invoked in either the first form (host mode) or second form (router mode).

On a host, `in.rdisc` populates the network routing tables with default routes. On a router, advertises the router to all the hosts.

`in.rdisc` is managed by the service management facility (SMF), by means of the service identifier:

```
svc:/network/routing/rdisc:default
```

Host (First Form) On a host, `in.rdisc` listens on the `ALL_HOSTS` (224.0.0.1) multicast address for `ROUTER_ADVERTISE` messages from routers. The received messages are handled by first ignoring those listed router addresses with which the host does not share a network. Among the remaining addresses, the ones with the highest preference are selected as default routers and a default route is entered in the kernel routing table for each one of them.

Optionally, `in.rdisc` can avoid waiting for routers to announce themselves by sending out a few `ROUTER_SOLICITATION` messages to the `ALL_ROUTERS` (224.0.0.2) multicast address when it is started.

A timer is associated with each router address. The address will no longer be considered for inclusion in the routing tables if the timer expires before a new *advertise* message is received from the router. The address will also be excluded from consideration if the host receives an *advertise* message with the preference being maximally negative or with a lifetime of zero.

Router (Second Form) When `in.rdisc` is started on a router, it uses the `SIOCGIFCONF` `ioctl(2)` to find the interfaces configured into the system and it starts listening on the `ALL_ROUTERS` multicast address on all the interfaces that support multicast. It sends out *advertise* messages to the `ALL_HOSTS` multicast address advertising all its `IP` addresses. A few initial *advertise* messages are sent out during the first 30 seconds and after that it will transmit *advertise* messages approximately every 600 seconds.

When `in.rdisc` receives a *solicitation* message, it sends an *advertise* message to the host that sent the *solicitation* message.

When `in.rdisc` is terminated by a signal, it sends out an *advertise* message with the preference being maximally negative.

Options Supported options and equivalent SMF service properties are listed below. SMF service properties are set using a command of the form:

```
# routeadm -m rdisc:default key=value
```

`-a` Accept all routers independent of the preference they have in their *advertise* messages. Normally, `in.rdisc` only accepts (and enters in the kernel routing tables) the router or routers with the highest preference. Use of this option is equivalent to setting the `accept_all` property to true.

`-f` Run `in.rdisc` forever even if no routers are found. Normally, `in.rdisc` gives up if it has not received any *advertise* message after soliciting three times, in which case it exits with a non-zero exit code. If `-f` is not specified in the first form then `-s` must be specified. For SMF execution, this option is required.

`-r` Act as a router, rather than a host. Use of this option is equivalent to setting the `act_as_router` property to true.

`-s` Send three *solicitation* messages initially to quickly discover the routers when the system is booted. When `-s` is specified, `in.rdisc` exits with a non-zero exit code if it can not find any routers. This can be overridden with the `-f` option. This option is not compatible with SMF execution and is not supported for the `rdisc` service.

`-p` *preference* Set the preference transmitted in the *solicitation* messages. The default is zero. Use of this option is equivalent to setting the `preference` property.

`-T` *interval* Set the interval between transmitting the *advertise* messages. The default time is 600 seconds. Use of this option is equivalent to setting the `transmit_interval` property.

The `send-address` and `receive-address` daemon options can be set by means of the `send_address` and `receive_address` properties.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWroute

See Also `in.routed(1M)`, `routeadm(1M)`, `svcadm(1M)`, `ioctl(2)`, `gateways(4)`, `attributes(5)`, `smf(5)`, `icmp(7P)`, `inet(7P)`

Deering, S.E., editor, *ICMP Router Discovery Messages*, RFC 1256, Network Information Center, SRI International, Menlo Park, California, September 1991.

Name in.rexecd, rexecd – remote execution server

Synopsis

```
in.rexecd
```

Description `in.rexecd` is the server for the `rexec(3SOCKET)` routine. The server provides remote execution facilities with authentication based on user names and passwords. It is invoked automatically as needed by `inetd(1M)`, and then executes the following protocol:

1. The server reads characters from the socket up to a null (`\0`) byte. The resultant string is interpreted as an ASCII number, base 10.
2. If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the `stderr`. A second connection is then created to the specified port on the client's machine.
3. A null terminated user name of at most 16 characters is retrieved on the initial socket.
4. A null terminated password of at most 16 characters is retrieved on the initial socket.
5. A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
6. `rexecd` then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted and a diagnostic message is returned.
7. A null byte is returned on the connection associated with the `stderr` and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by `rexecd`.

Usage `in.rexecd` and `rexecd` are IPv6–enabled. See `ip6(7P)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also `svcs(1)`, `inetd(1M)`, `inetadm(1M)`, `svcadm(1M)`, `rexec(3SOCKET)`, `attributes(5)`, `smf(5)`, `ip6(7P)`

Diagnostics All diagnostic messages are returned on the connection associated with the `stderr`, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

`username too long` The name is longer than 16 characters.

`password too long` The password is longer than 16 characters.

`command too long`	The command line passed exceeds the size of the argument list (as configured into the system).
`Login incorrect.`	No password file entry for the user name existed.
`Password incorrect.`	The wrong password was supplied.
`No remote directory.`	The `chdir` command to the home directory failed.
`Try again.`	A `fork` by the server failed.
`/usr/bin/sh: ...`	The user's login shell could not be started.

Notes The `in.rexecd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rexec:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name in.ripngd – network routing daemon for IPv6

Synopsis
```
/usr/sbin/in.ripngd  [-s] [-q] [-t] [-p n] [-P] [-v ]
  [logfile]
```

Description `in.ripngd` is the IPv6 equivalent of `in.routed(1M)`. It is invoked at boot time to manage the network routing tables. The routing daemon uses the Routing Information Protocol for IPv6.

`in.ripngd` is managed by the service management facility (SMF), by means of the service identifier:

```
svc:/network/routing/ripng:default
```

In normal operation, `in.ripngd` listens on the `udp(7P)` socket port 521 for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When `in.ripngd` is started, it uses the `SIOCGLIFCONF` `ioctl(2)` to find those directly connected IPv6 interfaces configured into the system and marked "up"; the software loopback interface is ignored. If multiple interfaces are present, it is assumed the host will forward packets between networks. `in.ripngd` then multicasts a request packet on each IPv6 interface and enters a loop, listening for request and response packets from other hosts.

When a request packet is received, `in.ripngd` formulates a reply based on the information maintained in its internal tables. The response packet contains a list of known routes. With each route is a number specifying the number of bits in the prefix. The prefix is the number of bits in the high order part of an address that indicate the subnet or network that the route describes. Each route reported also has a "*hop count*" metric. A count of 16 or greater is considered "infinity." The metric associated with each route returned provides a metric relative to the sender.

The request packets received by `in.ripngd` are used to update the routing tables if one of the following conditions is satisfied:

- No routing table entry exists for the destination network or host, and the metric indicates the destination is "reachable", that is, the *hop count* is not infinite.
- The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
- The existing entry in the routing table has not been updated for a period of time, defined to be 90 seconds, and the route is at least as cost-effective as the current route.
- The new route describes a shorter route to the destination than the one currently stored in the routing tables; this is determined by comparing the metric of the new route against the one stored in the table.

When an update is applied, `in.ripngd` records the change in its internal tables and generates a response packet to all directly connected hosts and networks. To allow possible unstable situations to settle, `in.ripngd` waits a short period of time (no more than 30 seconds) before modifying the kernel's routing tables.

In addition to processing incoming packets, `in.ripngd` also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks.

Options `in.ripngd` supports the options listed below. Listed with the options are the equivalent SMF property values. These are set for the `ripng:default` service with a command of the form:

```
# routeadm -m ripng:default key=value
```

`-p` *n* Send and receive the routing packets from other routers using the UDP port number *n*. Use of this option is equivalent to setting the `udp_port` property.

`-P` Do not use poison reverse. Use of this option is equivalent to setting the `poison_reverse` property to false.

`-q` Do not supply routing information. Use of this option is equivalent to setting the `quiet_mode` property to true.

`-s` Force `in.ripngd` to supply routing information whether it is acting as an internetwork router or not. Use of this option is equivalent to setting the `supply_routes` property to true.

`-t` Print all packets sent or received to standard output. `in.ripngd` will not divorce itself from the controlling terminal. Accordingly, interrupts from the keyboard will kill the process. Not supported by the `ripng` service.

`-v` Print all changes made to the routing tables to standard output with a timestamp. Use of this option is equivalent to setting the `verbose` property to true.

Any other argument supplied to this option is interpreted as the name of the file in which the actions of `in.ripngd`, as specified by this option or by `-t`, should be logged instead of being sent to standard output.

The logfile can be specified for the `ripng` service by means of the `log_file` property.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWroute

See Also `in.routed(1M)`, `routeadm(1M)`, `svcadm(1M)`, `ioctl(2)`, `attributes(5)`, `smf(5)`, `udp(7P)`

G. Malkin, R. Minnear, *RFC 2080, RIPng for IPv6*, January 1997.

Notes The kernel's routing tables may not correspond to those of `in.ripngd` for short periods of time while processes that utilize existing routes exit; the only remedy for this is to place the routing process in the kernel.

`in.ripngd` currently does not support all of the functionality of `in.routed(1M)`. Future releases may support more if appropriate.

`in.ripngd` initially obtains a routing table by examining the interfaces configured on a machine. It then sends a request on all directly connected networks for more routing information. `in.ripngd` does not recognize or use any routing information already established on the machine prior to startup. With the exception of interface changes, `in.ripngd` does not see any routing table changes that have been done by other programs on the machine, for example, routes added, deleted or flushed by means of the `route(1M)` command. Therefore, these types of changes should not be done while `in.ripngd` is running. Rather, shut down `in.ripngd`, make the changes required, and then restart `in.ripngd`.

Name in.rlogind, rlogind – remote login server

Synopsis

```
/usr/sbin/in.rlogind [-k5eExXciPp] [-s tos] [-S keytab]
     [-M realm]
```

Description `in.rlogind` is the server for the `rlogin(1)` program. The server provides a remote login facility with authentication based on Kerberos V5 or privileged port numbers.

`in.rlogind` is invoked by `inetd(1M)` when a remote login connection is established. When Kerberos V5 authentication is required (see option `-k` below), the authentication sequence is as follows:

- Check Kerberos V5 authentication.
- Check authorization according to the rules in `krb5_auth_rules(5)`.
- Prompt for a password if any checks fail and `/etc/pam.conf` is configured to do so.

In order for Kerberos authentication to work, a `host/`<*FQDN*> Kerberos principal must exist for each Fully Qualified Domain Name associated with the `in.rlogind` server. Each of these `host/`<*FQDN*> principals must have a `keytab` entry in the `/etc/krb5/krb5.keytab` file on the `in.rlogind` server. An example principal might be:

```
host/bigmachine.eng.example.com
```

See `kadmin(1M)` or `gkadmin(1M)` for instructions on adding a principal to a `krb5.keytab` file. See *System Administration Guide: Security Services* for a discussion of Kerberos authentication.

If Kerberos V5 authentication is not enabled, then the authentication procedure follows the standard `rlogin` protocol:

- The server checks the client's source port. If the port is not in the range 512-1023, the server aborts the connection.
- The server checks the client's source address. If an entry for the client exists in both `/etc/hosts` and `/etc/hosts.equiv`, a user logging in from the client is not prompted for a password. If the address is associated with a host for which no corresponding entry exists in `/etc/hosts`, the user is prompted for a password, regardless of whether or not an entry for the client is present in `/etc/hosts.equiv`. See `hosts(4)` and `hosts.equiv(4)`.

Once the source port and address have been checked, `in.rlogind` allocates a pseudo-terminal and manipulates file descriptors so that the slave half of the pseudo-terminal becomes the `stdin`, `stdout`, and `stderr` for a login process. The login process is an instance of the `login(1)` program, invoked with the `-r`.

The login process then proceeds with the `pam(3PAM)` authentication process. See `SECURITY` below. If automatic authentication fails, it reprompts the user to login.

The parent of the login process manipulates the master side of the pseudo-terminal, operating as an intermediary between the login process and the client instance of the `rlogin` program. In normal operation, a packet protocol is invoked to provide Ctrl-S and Ctrl-Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, `TERM`.

Options The following options are supported:

`-5` Same as `-k`, for backwards compatibility.

`-c` Requires Kerberos V5 clients to present a cryptographic checksum of initial connection information like the name of the user that the client is trying to access in the initial authenticator. This checksum provides additionl security by preventing an attacker from changing the initial connection information. This option is mutually exclusive with the `-i` option.

`-e` Creates an encrypted session.

`-E` Same as `-e`, for backwards compatibility.

`-i` Ignores authenticator checksums if provided. This option ignores authenticator checksums presented by current Kerberos clients to protect initial connection information. Option `-i` is the opposite of option `-c`.

`-k` Allows Kerberos V5 authentication with the `.k5login` access control file to be trusted. If this authentication system is used by the client and the authorization check is passed, then the user is allowed to log in.

`-M` *realm* Uses the indicated Kerberos V5 realm. By default, the daemon will determine its realm from the settings in the `krb5.conf(4)` file.

`-p` Prompts for authentication only if other authentication checks fail.

`-P` Prompts for a password in addition to other authentication methods.

`-s` *tos* Sets the `IP TOS` option.

`-S` *keytab* Sets the `KRB5` keytab file to use. The `/etc/krb5/krb5.keytab` file is used by default.

`-x` Same as `-e`, for backwards compatibility.

`-X` Same as `-e`, for backwards compatibility.

Usage `rlogind` and `in.rlogind` are IPv6–enabled. See `ip6(7P)`. `IPv6` is not currently supported with Kerberos V5 authentication.

Typically, Kerberized `rlogin` service runs on port 543 (klogin) and Kerberized, encrypted `rlogin` service runs on port 2105 (eklogin). The corresponding FMRI entries are:

```
svc:/network/login:klogin (rlogin with kerberos)
svc:/network/login:eklogin (rlogin with kerberos and encryption)
```

Security `in.rlogind` uses pam(3PAM) for authentication, account management, and session management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the modules to be used for `in.rlogind`. Here is a partial `pam.conf` file with entries for the `rlogin` command using the "rhosts" and UNIX authentication modules, and the UNIX account, session management, and password management modules.

```
rlogin          auth sufficient     pam_rhosts_auth.so.1
rlogin          auth requisite      pam_authtok_get.so.1
rlogin          auth required       pam_dhkeys.so.1
rlogin          auth required       pam_unix_auth.so.1

rlogin          account required    pam_unix_roles.so.1
rlogin          account required    pam_unix_projects.so.1
rlogin          account required    pam_unix_account.so.1

rlogin          session required    pam_unix_session.so.1
```

With this configuration, the server checks the client's source address. If an entry for the client exists in both `/etc/hosts` and `/etc/hosts.equiv`, a user logging in from the client is not prompted for a password. If the address is associated with a host for which no corresponding entry exists in `/etc/hosts`, the user is prompted for a password, regardless of whether or not an entry for the client is present in `/etc/hosts.equiv`. See hosts(4) and hosts.equiv(4).

When running a Kerberized rlogin service (with or without the encryption option), the pam service name that should be used is "`krlogin`".

If there are no entries for the `rlogin` service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords. Removing the `pam_rhosts_auth.so.1` entry will disable the `/etc/hosts.equiv` and `~/.rhosts` authentication protocol and the user would always be forced to type the password. The *sufficient* flag indicates that authentication through the `pam_rhosts_auth.so.1` module is sufficient to authenticate the user. Only if this authentication fails is the next authentication module used.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also login(1), svcs(1), rlogin(1), gkadmin(1M), in.rshd(1M), inetadm(1M), inetd(1M), kadmin(1M), svcadm(1M), pam(3PAM), hosts(4), hosts.equiv(4), krb5.conf(4), pam.conf(4), attributes(5), environ(5), krb5_auth_rules(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), smf(5)

System Administration Guide: Security Services

Diagnostics All diagnostic messages are returned on the connection associated with the stderr, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

`Hostname for your address unknown.`	No entry in the host name database existed for the client's machine.
`Try again.`	A *fork* by the server failed.
`/usr/bin/sh:...`	The user's login shell could not be started.

Notes The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but it is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

The pam_unix(5) module is no longer supported. Similar functionality is provided by pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), and pam_unix_session(5).

The in.rlogind service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/login:rlogin (rlogin)
svc:/network/login:klogin (rlogin with kerberos)
svc:/network/login:eklogin (rlogin with kerberos and encryption)
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). Responsibility for initiating and restarting this service is delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

Name in.routed, routed – network routing daemon

Synopsis

```
/usr/sbin/in.routed [-AdghmnqsStVz] [-T tracefile [-v]]
 [-F net[/mask ][,metric]] [-P params]
```

Description The daemon `in.routed`, often referred to as `routed`, is invoked at boot time to manage the network routing tables. It uses Routing Information Protocol, RIPv1 (RFC 1058), RIPv2 (RFC 2453), and Internet Router Discovery Protocol (RFC 1256) to maintain the kernel routing table. The RIPv1 protocol is based on the reference 4.3BSD daemon.

`in.routed` is managed by means of the service management facility (SMF), using the fault management resource identifier (FMRI):

```
svc:/network/routing/route:default
```

The daemon listens on a `udp` socket for the `route` service (see `services(4)`) for Routing Information Protocol packets. It also sends and receives multicast Router Discovery ICMP messages. If the host is a router, `in.routed` periodically supplies copies of its routing tables to any directly connected hosts and networks. It also advertises or solicits default routes using Router Discovery ICMP messages.

When started (or when a network interface is later turned on), `in.routed` uses an `AF_ROUTE` address family facility to find those directly connected interfaces configured into the system and marked "up". It adds necessary routes for the interfaces to the kernel routing table. Soon after being first started, and provided there is at least one interface on which RIP has not been disabled, `in.routed` deletes all pre-existing non-static routes in the kernel table. Static routes in the kernel table are preserved and included in RIP responses if they have a valid RIP metric (see `route(1M)`).

If more than one interface is present (not counting the loopback interface), it is assumed that the host should forward packets among the connected networks. After transmitting a RIP request and Router Discovery Advertisements or Solicitations on a new interface, the daemon enters a loop, listening for RIP request and response and Router Discovery packets from other hosts.

When a request packet is received, `in.routed` formulates a reply based on the information maintained in its internal tables. The response packet generated contains a list of known routes, each marked with a "hop count" metric (a count of 16 or greater is considered "infinite"). Advertised metrics reflect the metric associated with an interface (see `ifconfig(1M)`), so setting the metric on an interface is an effective way to steer traffic.

Responses do not include routes with a first hop on the requesting network, to implement in part split-horizon. Requests from query programs such as `rtquery(1M)` are answered with the complete table.

The routing table maintained by the daemon includes space for several gateways for each destination to speed recovery from a failing router. RIP response packets received are used to

update the routing tables, provided they are from one of the several currently recognized gateways or advertise a better metric than at least one of the existing gateways.

When an update is applied, `in.routed` records the change in its own tables and updates the kernel routing table if the best route to the destination changes. The change in the kernel routing table is reflected in the next batch of response packets sent. If the next response is not scheduled for a while, a flash update response containing only recently changed routes is sent.

In addition to processing incoming packets, in.routed also periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed until the route has been advertised with an infnite metric to insure the invalidation is propagated throughout the local internet. This is a form of poison reverse.

Routes in the kernel table that are added or changed as a result of ICMP Redirect messages are deleted after a while to minimize black-holes. When a TCP connection suffers a timeout, the kernel tells `in.routed`, which deletes all redirected routes through the gateway involved, advances the age of all RIP routes through the gateway to allow an alternate to be chosen, and advances of the age of any relevant Router Discovery Protocol default routes.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. These RIP responses are sent to the broadcast address on nets that support broadcasting, to the destination address on point-to-point links, and to the router's own address on other networks. If RIPv2 is enabled, multicast packets are sent on interfaces that support multicasting.

If no response is received on a remote interface, if there are errors while sending responses, or if there are more errors than input or output (see `netstat(1M)`), then the cable or some other part of the interface is assumed to be disconnected or broken, and routes are adjusted appropriately.

The Internet Router Discovery Protocol is handled similarly. When the daemon is supplying RIP routes, it also listens for Router Discovery Solicitations and sends Advertisements. When it is quiet and listening to other RIP routers, it sends Solicitations and listens for Advertisements. If it receives a good Advertisement and it is not multi-homed, it stops listening for broadcast or multicast RIP responses. It tracks several advertising routers to speed recovery when the currently chosen router dies. If all discovered routers disappear, the daemon resumes listening to RIP responses. It continues listening to RIP while using Router Discovery if multi-homed to ensure all interfaces are used.

The Router Discovery standard requires that advertisements have a default “lifetime” of 30 minutes. That means should something happen, a client can be without a good route for 30 minutes. It is a good idea to reduce the default to 45 seconds using `-P rdisc_interval=45` on the command line or `rdisc_interval=45` in the `/etc/gateways` file. See `gateways(4)`.

While using Router Discovery (which happens by default when the system has a single network interface and a Router Discover Advertisement is received), there is a single default route and a variable number of redirected host routes in the kernel table. On a host with more than one network interface, this default route will be via only one of the interfaces. Thus, multi-homed hosts running with `-q` might need the `no_rdisc` argument described below.

To support "legacy" systems that can handle neither RIPv2 nor Router Discovery, you can use the `pm_rdisc` parameter in the `/etc/gateways`. See `gateways(4)`.

By default, neither Router Discovery advertisements nor solicitations are sent over point-to-point links (for example, PPP). The Solaris OE uses a netmask of all ones (255.255.255.255) on point-to-point links.

`in.routed` supports the notion of "distant" passive or active gateways. When the daemon is started, it reads the file `/etc/gateways` to find such distant gateways that cannot be located using only information from a routing socket, to discover if some of the local gateways are passive, and to obtain other parameters. Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange RIP packets. Routes through passive gateways are installed in the kernel's routing tables once upon startup and are not included in transmitted RIP responses.

Distant active gateways are treated like network interfaces. RIP responses are sent to the distant active gateway. If no responses are received, the associated route is deleted from the kernel table and RIP responses are advertised via other interfaces. If the distant gateway resumes sending RIP responses, the associated route is restored.

Distant active gateways can be useful on media that do not support broadcasts or multicasts but otherwise act like classic shared media, such as some ATM networks. One can list all RIP routers reachable on the HIPPI or ATM network in `/etc/gateways` with a series of "host" lines. Note that it is usually desirable to use RIPv2 in such situations to avoid generating lists of inferred host routes.

Gateways marked external are also passive, but are not placed in the kernel routing table, nor are they included in routing updates. The function of external entries is to indicate that another routing process will install such a route if necessary, and that other routes to that destination should not be installed by `in.routed`. Such entries are required only when both routers might learn of routes to the same destination.

Options Listed below are available options. Any other argument supplied is interpreted as the name of a file in which the actions of `in.routed` should be logged. It is better to use `-T` (described below) instead of appending the name of the trace file to the command. Associated SMF properties for these options are described, and can be set by means of a command of the form:

```
# routeadm -m route:default name=value
```

`-A`
Do not ignore RIPv2 authentication if we do not care about RIPv2 authentication. This option is required for conformance with RFC 2453. However, it makes no sense and breaks using RIP as a discovery protocol to ignore all RIPv2 packets that carry authentication when this machine does not care about authentication. This option is equivalent to setting the `ignore_auth` property value to false.

`-d`
Do not run in the background. This option is meant for interactive use and is not usable under the SMF.

`-F net[/`*mask*`][,`*metric*`]`
Minimize routes in transmissions via interfaces with addresses that match *net* (network number)/*mask* (netmask), and synthesizes a default route to this machine with the *metric*. The intent is to reduce RIP traffic on slow, point-to-point links, such as PPP links, by replacing many large UDP packets of RIP information with a single, small packet containing a "fake" default route. If *metric* is absent, a value of 14 is assumed to limit the spread of the "fake" default route. This is a dangerous feature that, when used carelessly, can cause routing loops. Notice also that more than one interface can match the specified network number and mask. See also `-g`. Use of this option is equivalent to setting the `minimize_routes` property.

`-g`
Used on internetwork routers to offer a route to the "default" destination. It is equivalent to `-F 0/0,1` and is present mostly for historical reasons. A better choice is `-P pm_rdisc` on the command line or `pm_rdisc` in the `/etc/gateways` file. A larger metric will be used with the latter alternatives, reducing the spread of the potentially dangerous default route. The `-g` (or `-P`) option is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers. Note that because a metric of 1 is used, this feature is dangerous. Its use more often creates chaos with a routing loop than solves problems. Use of this option is equivalent to setting the `offer_default_route` property to true.

`-h`
Causes host or point-to-point routes not to be advertised, provided there is a network route going the same direction. That is a limited kind of aggregation. This option is useful on gateways to LANs that have other gateway machines connected with point-to-point links such as SLIP. Use of this option is equivalent to setting the `advertise_host_routes` property to false.

`-m`
Cause the machine to advertise a host or point-to-point route to its primary interface. It is useful on multi-homed machines such as NFS servers. This option should not be used except when the cost of the host routes it generates is justified by the popularity of the server. It is effective only when the machine is supplying routing information, because there is more than one interface. The `-m` option overrides the `-q` option to the limited extent

of advertising the host route. Use of this option is equivalent to setting the `advertise_host_routes_primary` property to true.

`-n`
Do not install routes in kernel. By default, routes are installed in the kernel. Use of this option is equivalent to setting the `install_routes` property to false.

`-P` *params*
Equivalent to adding the parameter line *params* to the `/etc/gateways` file. Can also be set by means of the `parameters` property.

`-q`
Opposite of the `-s` option. This is the default when only one interface is present. With this explicit option, the daemon is always in "quiet mode" for RIP and does not supply routing information to other computers. Use of this option is equivalent to setting the `quiet_mode` property to true.

`-s`
Force `in.routed` to supply routing information. This is the default if multiple network interfaces are present on which RIP or Router Discovery have not been disabled, and if the `/dev/ip` `ndd` variable `ip_forwarding` is set to 1. Use of this option is equivalent to setting the `supply_routes` property to true.

`-S`
If `in.routed` is not acting as an internetwork router, instead of entering the whole routing table in the kernel, it enters only a default route for each internetwork router. This reduces the memory requirements without losing any routing reliability. This option is provided for compatibility with the previous, RIPv1–only `in.routed`. Use of this option is generally discouraged. Use of this option is equivalent to setting the `default_routes_only` property to true.

`-t`
Runs in the foreground (as with `-d`) and logs the contents of the packets received (as with `-zz`). This is for compatibility with prior versions of Solaris and has no SMF equivalent.

`-T` *tracefile*
Increases the debugging level to at least 1 and causes debugging information to be appended to the trace file. Because of security concerns, do not to run `in.routed` routinely with tracing directed to a file. Use of this option is equivalent to setting the `log_file` property to `trace file path`.

`-v`
Enables debug. Similar to `-z`, except, where `-z` increments `trace_level`, `-v` sets `trace_level` to 1. Also, `-v` requires the `-T` option. Use of this option is equivalent to setting the `debug` property to true.

`-V`
Displays the version of the daemon.

`-z`
Increase the debugging level, which causes more information to be logged on the tracefile specified with `-T` or stdout. The debugging level can be increased or decreased with the `SIGUSR1` or `SIGUSR2` signals or with the `rtquery(1M)` command.

Files

`/etc/defaultrouter`	If this file is present and contains the address of a default router, the system startup script does not run `in.routed`. See `defaultrouter(4)`.
`/etc/gateways`	List of distant gateways and general configuration options for `in.routed`. See `gateways(4)`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWroute

See Also

`route(1M)`, `routeadm(1M)`, `rtquery(1M)`, `svcadm(1M)`, `ioctl(2)`, `inet(3SOCKET)`, `defaultrouter(4)`, `gateways(4)`, `attributes(5)`, `icmp(7P)`, `inet(7P)`, `udp(7P)`

Internet Transport Protocols, XSIS 028112, Xerox System Integration Standard

Routing Information Protocol, v2 (RFC 2453, STD 0056, November 1998)

RIP-v2 MD5 Authentication (RFC 2082, January 1997)

Routing Information Protocol, v1 (RFC 1058, June 1988)

ICMP Router Discovery Messages (RFC 1256, September 1991)

Notes

In keeping with its intended design, this daemon deviates from RFC 2453 in two notable ways:

- By default, `in.routed` does not discard authenticated RIPv2 messages when RIP authentication is not configured. There is little to gain from dropping authenticated packets when RIPv1 listeners will gladly process them. Using the `-A` option causes `in.routed` to conform to the RFC in this case.
- Unauthenticated RIP requests are never discarded, even when RIP authentication is configured. Forwarding tables are not secret and can be inferred through other means such as test traffic. RIP is also the most common router-discovery protocol, and hosts need to send queries that will be answered.

`in.routed` does not always detect unidirectional failures in network interfaces, for example, when the output side fails.

Name in.rshd, rshd – remote shell server

Synopsis

```
in.rshd [-k5eciU] [-s tos] [-S keytab] [-M realm]
     [-L env_var] host.port
```

Description `in.rshd` is the server for the `rsh(1)` program. The server provides remote execution facilities with authentication based on Kerberos V5 or privileged port numbers.

`in.rshd` is invoked by `inetd(1M)` each time a shell service is requested.

When Kerberos V5 authentication is required (this can be set with Kerberos-specific options listed below), the following protocol is initiated:

1. Check Kerberos V5 authentication.
2. Check authorization according to rules in `krb5_auth_rules(5)`.
3. A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. (The `PATH` variable is set to `/usr/bin`.) The shell inherits the network connections established by `in.rshd`.

In order for Kerberos authentication to work, a `host/`<*FQDN*> Kerberos principal must exist for each Fully Qualified Domain Name associated with the `in.rshd` server. Each of these `host/`<*FQDN*> principals must have a `keytab` entry in the `/etc/krb5/krb5.keytab` file on the `in.rshd` server. An example principal might be:

```
host/bigmachine.eng.example.com
```

See `kadmin(1M)` or `gkadmin(1M)` for instructions on adding a principal to a `krb5.keytab` file. See *System Administration Guide: Security Services* for a discussion of Kerberos authentication.

If Kerberos V5 authentication is not enabled, then `in.rshd` executes the following protocol:

1. The server checks the client's source port. If the port is not in the range 512-1023, the server aborts the connection. The client's host address (in hex) and port number (in decimal) are the arguments passed to `in.rshd`.
2. The server reads characters from the socket up to a null (`\\0`) byte. The resultant string is interpreted as an ASCII number, base 10.
3. If the number received in step 2 is non-zero, it is interpreted as the port number of a secondary stream to be used for the `stderr`. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 512-1023.
4. A null-terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the *client*'s machine.
5. A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the *server*'s machine.

6. A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
7. `in.rshd` then validates the user according to the following steps. The remote user name is looked up in the password file and a `chdir` is performed to the user's home directory. If the lookup fails, the connection is terminated. If the `chdir` fails, it does a `chdir` to `/` (root). If the user is not the superuser, (user ID 0), and if the pam_rhosts_auth PAM module is configured for authentication, the file `/etc/hosts.equiv` is consulted for a list of hosts considered "equivalent". If the client's host name is present in this file, the authentication is considered successful. See the SECURITY section below for a discussion of PAM authentication.

   If the lookup fails, or the user is the superuser, then the file `.rhosts` in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine. If this lookup fails, the connection is terminated
8. A null byte is returned on the initial connection and the command line is passed to the normal login shell of the user. The `PATH` variable is set to `/usr/bin`. The shell inherits the network connections established by `in.rshd`.

Options The following options are supported:

`-5`	Same as `-k`, for backwards compatibility
`-c`	Requires Kerberos V5 clients to present a cryptographic checksum of initial connection information like the name of the user that the client is trying to access in the initial authenticator. This checksum provides additionl security by preventing an attacker from changing the initial connection information. This option is mutually exclusive with the `-i` option.
`-e`	Requires the client to encrypt the connection.
`-i`	Ignores authenticator checksums if provided. This option ignores authenticator checksums presented by current Kerberos clients to protect initial connection information. Option `-i` is the opposite of option `-c`.
`-k`	Allows Kerberos V5 authentication with the `.k5login` access control file to be trusted. If this authentication system is used by the client and the authorization check is passed, then the user is allowed to log in.
`-L` *env_var*	List of environment variables that need to be saved and passed along.
`-M` *realm*	Uses the indicated Kerberos V5 realm. By default, the daemon will determine its realm from the settings in the `krb5.conf(4)` file.
`-s` *tos*	Sets the `IP TOS` option.
`-S` *keytab*	Sets the `KRB5` keytab file to use. The`/etc/krb5/krb5.keytab` file is used by default.

`-U` Refuses connections that cannot be mapped to a name through the `getnameinfo(3SOCKET)` function.

Usage `rshd` and `in.rshd` are IPv6–enabled. See `ip6(7P)`. `IPv6` is not currently supported with Kerberos V5 authentication.

The Kerberized `rshd` service runs on port 544 (kshell). The corresponding FMRI entry is: :

```
svc:/network/shell:kshell (rshd with kerberos (ipv4 only))
```

Security `in.rshd` uses `pam(3PAM)` for authentication, account management, and session management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the modules to be used for `in.rshd`. Here is a partial `pam.conf` file with entries for the `rsh` command using rhosts authentication, UNIX account management, and session management module.

rsh	auth	required	pam_rhosts_auth.so.1
rsh	account	required	pam_unix_roles.so.1
rsh	session	required	pam_unix_projects.so.1
rsh	session	required	pam_unix_account.so.1
rsh	session	required	pam_unix_session.so.1

If there are no entries for the `rsh` service, then the entries for the "other" service are used. To maintain the authentication requirement for `in.rshd`, the rsh entry must always be configured with the `pam_rhosts_auth.so.1` module.

`in.rshd` can authenticate using Kerberos V5 authentication or `pam(3PAM)`. For Kerberized `rsh` service, the appropriate PAM service name is `krsh`.

Files `/etc/hosts.equiv`

`$HOME/.k5login` File containing Kerberos principals that are allowed access.

`/etc/krb5/krb5.conf` Kerberos configuration file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also rsh(1), svcs(1), gkadmin(1M), inetadm(1M), inetd(1M), kadmin(1M), svcadm(1M), pam(3PAM), getnameinfo(3SOCKET), hosts(4), krb5.conf(4), pam.conf(4), attributes(5), environ(5), krb5_auth_rules(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_rhosts_auth(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), smf(5), ip6(7P)

System Administration Guide: Security Services

Diagnostics The following diagnostic messages are returned on the connection associated with `stderr`, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 in step 8 above (`0` is returned above upon successful completion of all the steps prior to the command execution).

`locuser too long`	The name of the user on the client's machine is longer than 16 characters.
`remuser too long`	The name of the user on the remote machine is longer than 16 characters.
`command too long`	The command line passed exceeds the size of the argument list (as configured into the system).
`Hostname for your address unknown.`	No entry in the host name database existed for the client's machine.
`Login incorrect.`	No password file entry for the user name existed.
`Permission denied.`	The authentication procedure described above failed.
`Can't make pipe.`	The pipe needed for the `stderr` was not created.
`Try again.`	A *fork* by the server failed.

Notes The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but it is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

The `pam_unix`(5) module is no longer supported. Similar functionality is provided by pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), and pam_unix_session(5).

The `in.rshd` service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/shell:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). Responsibility for initiating and restarting this service is delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

Name in.rwhod, rwhod – system status server

Synopsis `/usr/sbin/in.rwhod [-m [ttl]]`

Description `in.rwhod` is the server which maintains the database used by the `rwho(1)` and `ruptime(1)` programs. Its operation is predicated on the ability to broadcast or multicast messages on a network.

`in.rwhod` operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast or multicast on a network. As a consumer of information, it listens for other `in.rwhod` servers' status messages, validating them, then recording them in a collection of files located in the directory `/var/spool/rwho`.

The `rwho` server transmits and receives messages at the port indicated in the `rwho` service specification, see `services(4)`. The messages sent and received are defined in `/usr/include/protocols/rwhod.h` and are of the form:

```
struct  outmp {
        char    out_line[8];    /* tty name */
        char    out_name[8];    /* user id */
        long    out_time;       /* time on */
};
struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int     we_idle;
    } wd_we[1024 / sizeof (struct whoent)];
 };
```

All fields are converted to network byte order prior to transmission. The load averages are as calculated by the `w(1)` program, and represent load averages over the 1, 5, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the `uname(2)` system call. The array at the end of the message contains information about the users who are logged in to the sending machine. This information includes the contents of the `utmpx(4)` entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the `rwho` server are discarded unless they originated at a `rwho` server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by `in.rwhod` are placed in files named `whod.hostname` in the directory `/var/spool/rwho`. These files contain only the most recent message, in the format described above.

Status messages are generated approximately once every 3 minutes.

Options The following options are supported:

`-m` [*ttl*] Use the rwho IP multicast address (224.0.1.3) when transmitting. Receive announcements both on this multicast address and on the IP broadcast address. If *ttl* is not specified `in.rwhod` multicasts on all interfaces but with the IP TimeToLive set to 1 (that is, packets are not forwarded by multicast routers.) If *ttl* is specified `in.rwhod` only transmits packets on one interface and setting the IP TimeToLive to the specified *ttl*.

Files `/var/spool/rwho/whod.*` information about other machines

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also `ruptime(1)`, `rwho(1)`, `w(1)`, `uname(2)`, `services(4)`, `utmpx(4)`, `attributes(5)`

Warnings This service can cause network performance problems when used by several hosts on the network. It is not run at most sites by default. If used, include the `-m` multicast option.

Notes This service takes up progressively more network bandwidth as the number of hosts on the local net increases. For large networks, the cost becomes prohibitive.

`in.rwhod` should relay status information between networks. People often interpret the server dying as a machine going down.

Name install – install commands

Synopsis

```
/usr/sbin/install -c dira [-m mode] [-u user] [-g group]
     [-o] [-s] file

/usr/sbin/install -f dirb [-m mode] [-u user] [-g group]
     [-o] [-s] file

/usr/sbin/install -n dirc [-m mode] [-u user] [-g group]
     [-o] [-s] file

/usr/sbin/install -d | -i [-m mode] [-u user] [-g group]
     [-o] [-s] dirx...

/usr/sbin/install [-m mode] [-u user] [-g group] [-o] [-s] file
     [dirx]...
```

Description `install` is most commonly used in "makefiles" (see `make(1S)`) to install a `file` in specific locations, or to create directories within a file system. Each `file` is installed by copying it into the appropriate directory.

`install` uses no special privileges to copy files from one place to another. The implications of this are:

- You must have permission to read the files to be installed.
- You must have permission to copy into the destination directory.
- You must have permission to change the modes on the final copy of the file if you want to use the `-m` option.
- You must be super-user if you want to specify the ownership of the installed file with the `-u` or `-g` options. If you are not the super-user, the installed file is owned by you, regardless of who owns the original.

Note that if the `ROOT` environment variable is set, each of the default directory paths are prefixed by its value (for example, `$ROOT/bin` and so on).

`install` prints messages telling the user exactly what files it is replacing or creating and where they are going.

If no options or directories (*dirx* . . .) are given, `install` searches a set of default directories (`/bin`, `/usr/bin`, `/etc`, `/lib`, and `/usr/lib`, in that order) for a file with the same name as `file`. When the first occurrence is found, `install` issues a message saying that it is overwriting that file with `file`, and proceeds to do so. If the file is not found, the program states this and exits.

If one or more directories (*dirx* . . .) are specified after `file`, those directories are searched before the default directories.

This version of `install` (`/usr/sbin/install`) is not compatible with the `install` binaries in many versions of Unix other than Solaris. For a higher degree of compatibility with other Unix versions, use `/usr/ucb/install`, which is described in the `install(1B)` man page.

Options The following options are supported:

`-c` *dira* Install `file` in the directory specified by *dira*, if `file` does not yet exist. If it is found, `install` issues a message saying that the file already exists, and exits without overwriting it.

`-f` *dirb* Force `file` to be installed in given directory, even if the file already exists. If the file being installed does not already exist, the mode and owner of the new file is set to `755` and `bin` , respectively. If the file already exists, the mode and owner is that of the already existing file.

`-n` *dirc* If `file` is not found in any of the searched directories, it is put in the directory specified in *dirc*. The mode and owner of the new file is set to `755` and `bin`, respectively.

`-d` Create a directory. Missing parent directories are created as required as in `mkdir -p`. If the directory already exists, the owner, group and mode is set to the values given on the command line.

`-i` Ignore default directory list, searching only through the given directories (*dirx* . . .).

`-m` *mode* The mode of the new file is set to *mode*. Set to `0755` by default.

`-u` *user* The owner of the new file is set to *user*. Only available to the super-user. Set to `bin` by default.

`-g` *group* The group id of the new file is set to *group*. Only available to the super-user. Set to `bin` by default.

`-o` If `file` is found, save the "found" file by copying it to `OLDfile` in the directory in which it was found. This option is useful when installing a frequently used file such as `/bin/sh` or `/lib/saf/ttymon`, where the existing file cannot be removed.

`-s` Suppress printing of messages other than error messages.

Usage See `largefile(5)` for the description of the behavior of `install` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `chgrp(1)`, `chmod(1)`, `chown(1)`, `cp(1)`, `install(1B)`, `make(1S)`, `mkdir(1)`, `attributes(5)`, `largefile(5)`

Name installboot – install bootblocks in a disk partition

Synopsis

```
installboot [-F zfs|ufs|hsfs] bootblk raw-disk-device
```

Description The boot(1M) program, `ufsboot`, is loaded from disk by the bootblock program which resides in the boot area of a disk partition. This program is filesystem-specific, and must match the type of filesystem on the disk to be booted.

The boot objects are platform-dependent and reside in the `/usr/platform/`*platform-name*`/lib/fs/`*file-system* directory. The platform name can be found using the `-i` option of uname(1). The filesystem type can be found using:

```
% fstyp raw-disk-device
```

See fstyp(1M).

The `installboot` utility is a SPARC only program. It is not supported on the x86 architecture. x86 users should use installgrub(1M) instead.

Options The following option is supported:

`-F zfs|ufs|hsfs`
Specifies the file system type of the boot block to be installed. Required if you wish to specify `zfs` or `hsfs`. The default is `ufs`.

Operands *bootblk*
The name of the bootblock code.

raw-disk-device
The name of the disk device onto which the bootblock code is to be installed; it must be a character device which is readable and writable. Naming conventions for a SCSI or IPI drive are `c?t?d?s?` and `c?d?s?` for an IDE drive.

Examples **EXAMPLE 1** Installing UFS Boot Block

To install a `ufs` boot block on slice `0` of target `0` on controller `1` of the platform where the command is being run, use:

```
# installboot /usr/platform/‘uname -i‘/lib/fs/ufs/bootblk \
    /dev/rdsk/c1t0d0s0
```

EXAMPLE 2 Installing ZFS Boot Block

To install a ZFS boot block on slice 0 of target 0 on controller 1 of the platform where the command is being run, use syntax such as the following:

```
# installboot -F zfs /usr/platform/‘uname -i‘/lib/fs/zfs/bootblk \
      /dev/rdsk/c0t1d0s0
```

Files `/usr/platform/`*platform-name*`/lib/fs/` Directory where boot objects reside.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `od(1)`, `uname(1)`, `boot(1M)`, `fstyp(1M)`, `init(1M)`, `kadb(1M)`, `kernel(1M)`, `monitor(1M)`, `reboot(1M)`, `rpc.bootparamd(1M)`, `init.d(4)`, `attributes(5)`

Solaris 10 Installation Guide: Basic Installations

Warnings The `installboot` utility fails if the *bootblk* or *openfirmware* files do not exist or if the raw disk device is not a character device.

Name installer – Solaris Web Start installer utility

Synopsis

```
installer [-locales list] [-nodisplay] [-noconsole]
     [-debug]
```

Description The `installer` utility invokes a Web Start install wizard sequence which will lead the user through a sequence of installation panels. This installer utility is found on many CDs that are shipped with Solaris and it will be found among the top level files of these CDs.

When the installer is on a CD being accessed from a desktop file manager, the installer can be double clicked to start the installation sequence. If the user is not currently the system's root user, the root user password will be requested.

The installer utility can also be run from other UNIX scripts. Usually, a script is used in conjunction with the utility's `-nodisplay` option. Add the `-noconsole` option for non-interactive scripts.

Options The following options are supported:

`-locales` *list* Selects product translations for install, corresponding to the specified list of locales if the specified locale translations are present on the installation media. Locales are supplied in a comma-separated list following the `-locales` option. An example *list* would appear as follows:

```
installer -locales fr,de,it
```

This would install products with translations for the French, German, and Italian locales.

`-nodisplay` Runs the install without a graphical user interface. Use the default product install unless it was modified by the `-locales` options.

`-noconsole` Run the install without any interactive text console device. Useful when paired with `-nodisplay` for non-interactive UNIX script use.

`-debug` Outputs extra information about what the install is doing. Mainly for install diagnostic purposes.

Files `/var/sadm/install/logs` location of installation log files

See Also `prodreg(1M)`

Name installf – add a file to the software installation database

Synopsis

```
installf [-c class] [ [-M] -R root_path] [-V fs_file] pkginst pathname
     [ftype [major minor] [mode owner group]]

installf [-c class] [ [-M] -R root_path] [-V fs_file] pkginst –

installf -f [-c class] [ [-M] -R root_path] [-V fs_file] pkginst
```

Description `installf` informs the system that a pathname not listed in the `pkgmap(4)` file is being created or modified. It should be invoked before any file modifications have occurred.

When the second synopsis is used, the pathname descriptions will be read from standard input. These descriptions are the same as would be given in the first synopsis but the information is given in the form of a list. The descriptions should be in the form:

pathname [*ftype* [*major minor*] [*mode owner group*]]

After all files have been appropriately created and/or modified, `installf` should be invoked with the `-f` synopsis to indicate that installation is final. Links will be created at this time and, if attribute information for a pathname was not specified during the original invocation of `installf`, or was not already stored on the system, the current attribute values for the pathname will be stored. Otherwise, `installf` verifies that attribute values match those given on the command line, making corrections as necessary. In all cases, the current content information is calculated and stored appropriately.

Package commands are `largefile(5)`-aware. They handle files larger than 2 GB in the same way they handle smaller files. In their current implementations, `pkgadd(1M)`, `pkgtrans(1)` and other package commands can process a datastream of up to 4 GB.

Options

`-c` *class* — Class to which installed objects should be associated. Default class is `none`.

`-f` — Indicates that installation is complete. This option is used with the final invocation of `installf` (for all files of a given class).

`-M` — Instruct `installf` not to use the `$`*root_path*`/etc/vfstab` file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

`-R` *root_path* — Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* can be specified when installing to a client from a server (for example, `/export/root/client1`).

`installf` inherits the value of the `PKG_INSTALL_ROOT` environment variable. (See ENVIRONMENT VARIABLES, below.) If `PKG_INSTALL_ROOT` is set, such as when the `-R` option is used with `pkgadd(1M)` or `pkgrm(1M)`

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-V` *fs_file* Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the `$`*root_path*`/etc/vfstab` file is non-existent or unreliable.

Operands

pkginst Name of package instance with which the pathname should be associated.

pathname Pathname that is being created or modified.

ftype A one-character field that indicates the file type. Possible file types include:

- `b` block special device
- `c` character special device
- `d` directory
- `e` a file to be edited upon installation or removal
- `f` a standard executable or data file
- `l` linked file
- `p` named pipe
- `s` symbolic link
- `v` volatile file (one whose contents are expected to change)
- `x` an exclusive directory

major The major device number. The field is only specified for block or character special devices.

minor The minor device number. The field is only specified for block or character special devices.

mode The octal mode of the file (for example, 0664). A question mark (`?`) indicates that the mode will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked or symbolically linked files.

owner The owner of the file (for example, `bin` or `root`). The field is limited to 14 characters in length. A question mark (`?`) indicates that the owner will be left unchanged, implying that the file already exists on the target machine. This field is not used for linked or symbolically linked files.

group The group to which the file belongs (for example, `bin` or `sys`). The field is limited to 14 characters in length. A question mark (`?`) indicates that the group will be left unchanged, implying that the file already exists on the target

machine. This field is not used for linked or symbolically linked files.

Examples

EXAMPLE 1 Basic Usage

The following example shows the use of `installf`, invoked from an optional pre-install or post-install script:

```
# create /dev/xt directory
# (needs to be done before drvinstall)
installf $PKGINST /dev/xt d 755 root sys ||
    exit 2
majno=‘/usr/sbin/drvinstall -m /etc/master.d/xt
     -d $BASEDIR/data/xt.o -v1.0‘ ||
    exit 2
i=00
while [ $i –lt $limit ]
do
    for j in 0 1 2 3 4 5 6 7
    do
        echo /dev/xt$i$j c $majno ‘expr $i ? 8 + $j‘
             644 root sys |
        echo /dev/xt$i$j=/dev/xt/$i$j
    done
    i=‘expr $i + 1‘
    [ $i –le 9 ] && i="0$i" #add leading zero
done | installf $PKGINST – || exit 2
# finalized installation, create links
installf -f $PKGINST || exit 2
```

Environment Variables

`installf` inherits the value of the following environment variable. This variable is set when `pkgadd(1M)` or `pkgrm(1M)` is invoked with the `-R` option.

`PKG_INSTALL_ROOT` — If present, defines the full path name of a directory to use as the system's `PKG_INSTALL_ROOT` path. All product and package information files are then looked for in the directory tree, starting with the specified `PKG_INSTALL_ROOT` path. If not present, the default system path of `/` is used.

Exit Status

`0` — Successful operation.

`>0` — An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also pkginfo(1), pkgmk(1), pkgparam(1), pkgproto(1), pkgtrans(1), pkgadd(1M), pkgask(1M), pkgchk(1M), pkgrm(1M), removef(1M), pkgmap(4), space(4), attributes(5), largefile(5)

Application Packaging Developer's Guide

Notes When *ftype* is specified, all applicable fields, as shown below, must be defined:

ftype	Required Fields
p, x, d, f, v, or e	`mode owner group`
c or b	`major minor mode owner group`

The `installf` command will create directories, named pipes and special devices on the original invocation. Links are created when `installf` is invoked with the `-f` option to indicate installation is complete.

Links should be specified as *path1=path2*. *path1* indicates the destination and *path2* indicates the source file.

Files installed with `installf` will be placed in the class `none`, unless a class is defined with the command. Subsequently, they will be removed when the associated package is deleted. If this file should not be deleted at the same time as the package, be certain to assign it to a class which is ignored at removal time. If special action is required for the file before removal, a class must be defined with the command and an appropriate class action script delivered with the package.

When classes are used, `installf` must be used in one of the following forms:

```
installf -c class1 . . .
installf -f -c class1 . . .
installf -c class2 . . .
installf -f -c class2 . . .
```

Name installgrub – install GRUB in a disk partition or a floppy

Synopsis `/sbin/installgrub [-fm]` *stage1 stage2 raw-device*

Description The `installgrub` command is an x86–only program. GRUB stands for GRand Unified Bootloader.

`installgrub` installs GRUB stage 1 and stage 2 files on the boot area of a disk partition. If you specify the `-m` option, `installgrub` installs the stage 1 file on the master boot sector of the disk.

Options The `installgrub` command accepts the following options:

`-f` Suppresses interaction when overwriting the master boot sector.

`-m` Installs GRUB *stage1* on the master boot sector interactively.

Operands The `installgrub` command accepts the following operands:

stage1 The name of the GRUB stage 1 file.

stage2 The name of the GRUB stage 2 file.

raw-device The name of the device onto which GRUB code is to be installed. It must be a character device that is readable and writable. For disk devices, specify the slice where the GRUB menu file is located. (For Solaris it is the root slice.) For a floppy disk, it is `/dev/rdiskette`.

Examples **EXAMPLE 1** Installing GRUB on a Hard Disk Slice

The following command installs GRUB on a system where the root slice is `c0d0s0`:

```
example# /sbin/installgrub /boot/grub/stage1 \
         /boot/grub/stage2 /dev/rdsk/c0d0s0
```

EXAMPLE 2 Installing GRUB on a Floppy

The following command installs GRUB on a formatted floppy:

```
example# mount -F pcfs /dev/diskette /mnt
# mkdir -p /mnt/boot/grub
# cp /boot/grub/* /mnt/boot/grub
# umount /mnt
# cd /boot/grub
# /sbin/installgrub stage1 stage2 /dev/rdiskette
```

Files `/boot/grub` Directory where GRUB files reside.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also boot(1M), fdisk(1M), fmthard(1M), kernel(1M), attributes(5)

Warnings Installing GRUB on the master boot sector (`-m` option) overrides any boot manager currently installed on the machine. The system will always boot the GRUB in the Solaris partition regardless of which `fdisk` partition is active.

Name install_scripts, add_install_client, add_to_install_server, rm_install_client, setup_install_server, check – scripts used to install the Solaris software

Synopsis

media-mnt-pt/Solaris_*XX*/Tools/add_install_client
[-i *IP_address*]
[-e *Ethernet_address*] [-s *server_name* : *path*]
[-c *server_name* : *path*]
[-n [*server*] : *name_service* [(*netmask*]]
[-p *server_name* : *path*]
[-t [*server*:]*install_boot_image_path* |
-t *server*:[*install_boot_image_path*]] *host_name platform_group*

media-mnt-pt/Solaris_*XX*/Tools/add_install_client -d
[-s *server_name*:*path*] [-f *boot_file_name*]
[-c *server_name*:*path*] [-p *server_name*:*path*]
[-t [*server*:]*install_boot_image_path* |
-t *server*:[*install_boot_image_path*]] *platform_name platform_group*

media-mnt-pt/Solaris_*XX*/Tools/add_install_client -d
[-s *server_name*:*path*] [-f *boot_file_name*]
[-c *server_name*:*path*] [-p *server_name*:*path*]
[-t [*server*:]*install_boot_image_path* |
-t *server*:[*install_boot_image_path*]] -e *Ethernet_address*
[-b *property*=*value*] *platform_group*

media-mnt-pt/Solaris_*XX*/Tools/add_to_install_server [-s]
[-p *product_image_path*] *install_server_path*

media-mnt-pt/Solaris_*XX*/Tools/jumpstart_sample/check
[-p *install_dir_path*]
[-r *rulesfile*]

media-mnt-pt/Solaris_*XX*/Tools/rm_install_client *host_name*

media-mnt-pt/Solaris_*XX*/Tools/rm_install_client *platform_name*

media-mnt-pt/Solaris_*XX*/Tools/rm_install_client -e *Ethernet_address*

media-mnt-pt/Solaris_*XX*/Tools/rm_install_client -f *boot_file_name*

media-mnt-pt/Solaris_*XX*/Tools/setup_install_server [-b]
[-t *install_boot_image_path*] [-w *wanboot_image_path*] *install_dir_path*

Description These commands are located on slice `0` of the Solaris Software and Solaris Installer CDs or DVDs. (The terms “CD” and “DVD” are hereafter referred to as “installation media”.) If the Solaris installation media has been copied to a local disk, `media_mnt_pt` is the path to the copied Solaris installation media. They can be used for a variety of installation tasks.

The *XX* in Solaris_*XX* is the version number of the Solaris release being used.

There are three versions of the `add_install_client` command. See `SYNOPSIS`.

Use the following version of the `add_install_client` command to add clients for network installation (these commands update the `bootparams(4)` file). The `add_install_client` command must be run from the install server's Solaris installation image (a mounted Solaris installation media or a Solaris installation media copied to disk) or the boot server's boot directory (if a boot server is required). The Solaris installation image or the boot directory must be the same Solaris release that you want installed on the client.

media-mnt-pt`/Solaris_`*XX*`/Tools/add_install_client`
`[-i` *IP_address*`]`
`[-e` *Ethernet_address*`] [-s` *server_name* `:` *path*`]`
`[-c` *server_name* `:` *path*`]`
`[-n [`*server*`] :` *name_service* `[(` *netmask*`]]`
`[-p` *server_name* `:` *path*`]`
`[-t [`*server*`:]`*install_boot_image_path* `|`
`-t` *server*`:[`*install_boot_image_path*`] ]`
host_name platform_group

Use the following version of the `add_install_client` command to add support for instances of a platform within a platform group to the install server. This group is booted and configured using DHCP. The script performs the necessary configuration steps on the server, and prints the data that the user needs to add to the DHCP server for the group.

media-mnt-pt`/Solaris_`*XX*`/Tools/add_install_client -d`
`[-s` *server*`:`*path*`]`
`[-c` *server*`:`*path*`] [-p` *server*`:`*path*`]`
`[-t [`*server*`:]`*install_boot_image_path* `|`
`-t` *server*`:[`*install_boot_image_path*`] ]`
`[-f` *boot file name*`]`
platform_name platform_group

Use the following version of the `add_install_client` command to add a single client to the install server. This client is booted and configured using DHCP. The script performs the necessary configuration steps on the server, and prints the data that the user needs to add to the DHCP server for the client. The `-f` flag used above needs to be added to the existing usage as well. `-f` allows the user to specify a boot file name to be used for a given client.

media-mnt-pt`/Solaris_`*XX*`/Tools/add_install_client -d`
`[-s` *server_name*`:`*path*`]`
`[-c` *server_name*`:`*path*`] [-p` *server_name*`:`*path*`]`
`[-t [`*server*`:]`*install_boot_image_path* `|`
`-t` *server*`:[`*install_boot_image_path*`] ]`
`[-f` *boot_file_name*`] -e` *Ethernet_address*
`[-b` *property*`=`*value*`]` *platform_group*

Always use the `-d` option when registering x86 Architecture Pre-boot eXecution Environment (PXE) clients. These clients use DHCP for their configuration.

Use `add_to_install_server` to merge other Solaris installation media with an existing image on a Net Install Server. Each installation media that can be merged (each OS CD or DVD, and

the Language CD or DVD) has its own `add_to_install_server` script. Do not use `add_to_install_server` scripts with installation media other than the ones with which they were delivered.

Use `check` to validate the rules in a `rules` file (this is only necessary if a custom JumpStart installation is being set up).

Use `rm_install_client` to remove clients for network installation (these commands update the `bootparams(4)` file).

Use `setup_install_server` to copy the Solaris installation media to a disk (to set up an install server), to build a WANboot miniroot image (to set up a WANboot install server), or to copy just the boot software of the Solaris installation media to a disk (to set up a boot server). An install server is required to install clients over the network. A boot server is also required for network installations if the install server and clients to be installed are on different subnets (the boot server must be located on the client's subnet).

Options The `add_install_client` supports the following options:

`-b` *property=value*
Set a property value in the client-specific `menu.lst` file located on the boot server's TFTP directory, `/tftpboot` by default. Use this option to set boot properties that are specific to this client.

Use this option only with an x86 client an only in conjunction with the `-d` and `-e` options.

`-c` *server_name*:*path*
This option is required only to specify a JumpStart directory for a custom JumpStart installation. *server_name* is the host name of the server with a JumpStart directory. *path* is the absolute path to the JumpStart directory.

`-d`
Specify as a DHCP client.

`-e` *Ethernet_address*
Specify the Ethernet address of the system to be installed.

`-f`
Specify the *boot_file_name* of the client to be installed.

`-i` *IP_address*
Specify the IP address of the client to be installed.

`-n` [*server*]`:` *name_service*[(*netmask*)]
This option specifies which name service should be used during system configuration. This sets the `ns` keyword in the `bootparams(4)` file.

name_service
Valid entries are `nis`, `nisplus`, and `none`.

netmask
A series of four numbers separated by periods, specifying which portion of an IP address is the network part, and which is the host part.

server
The name of the server or IP address of the specified name service. If the server specified is on a different subnet, then the *netmask* may be needed to enable the client to contact the server.

`-p` *server_name*`:` *path*
This option is used to specify the NFS or ZFS shared directory that contains the user-defined `sysidcfg` file. When a client is booted, it attempts to read a file specifically named `sysidcfg` in this directory to obtain answers to the system and network identification questions. *server_name* is either a valid host name or IP address. *path* is the absolute pathname of the directory on the file server that contains the `sysidcfg` file.

`-s` *server_name*`:`*path*
This option is required only when using `add_install_client` from a boot server. Specify the name of the server and the absolute path of the Solaris installation image that is used for this installation. *path* is either the path to a mounted Solaris installation media or a path to a directory with a copy of the Solaris installation media.

`-t` [*server*:]*path*
`-t` *server*:[*path*]

path
Allows you to specify an alternate miniroot.

server
The name or IP address of the boot server. This can be used on boot servers with multiple network interfaces to specify the particular IP address from which clients should be booted.

The `add_to_install_server` command supports the following options:

`-p`
Specifies the location of the installation media (containing the supplemental products) to be copied.

`-s`
Allows users to select from a list only the products needing installation.

The `check` command supports the following options:

`-p` *install_dir_path*
Validates the rules file by using the `check` script from a specified Solaris installation image, instead of the `check` script from the system you are using. `install_dir_path` is the path to a Solaris installation image on a local disk or a mounted Solaris installation media.

Use this option to run the most recent version of `check` if your system is running a previous version of Solaris.

`-r` *rulesfile*
Specifies a rules file other than the one named `rules`. Using this option, the validity of a rule can be tested before integrating it into the rules file. `check` reports whether or not the rule is valid, but it does not create the `rules.ok` file necessary for a custom JumpStart installation.

The `rm_install_client` command supports the following options:

`-e` *Ethernet_address*
Specify the Ethernet address of the system to be removed.

`-f`
Specify the *boot_file_name* of the client to be removed.

The `setup_install_server` command supports the following options:

`-b`
This option sets up the server only as a boot server.

`-t`
This option allows an alternate miniroot to be specified.

`-w`
This option builds a WANboot miniroot image.

Operands The `add_install_client` command supports the following operands:

host_name
This is the name of the client to be installed.

platform_group
Vendor-defined grouping of hardware platforms for the purpose of distributing specific software. Examples of valid platform groups are:

System	Platform Group
x86	`i86pc`
Sun Fire 4800	`sun4u`

Use the `uname(1)` command (with the `-m` option) to determine a system's platform group.

platform_name
Use the `uname(1)` command (with the `-i` option) to determine a system's platform name.

The following example shows the use of the `uname` command to determine the system platform name for an Ultra 10:

```
uname -i
```

The system responds with:

```
SUNW,Ultra-5_10
```

Therefore, the system's platform name is `SUNW,Ultra-5_10`.

The following command calls `add_install_client` for Ultra 10s:

```
add_install_client -d SUNW,Ultra-5_10 sun4u
```

For `IA32` platforms, the platform name is always `SUNW.i86pc`.

The following command calls `add_install_client` for `IA32` platforms:

```
add_install_client -d SUNW.i86pc i86pc
```

install_boot_image_path
Pathname of alternate miniroot, specified with `-t` option.

The `rm_install_client` command supports the following operands:

host_name
Name of the client to be removed.

platform_name
The platform name of the client to be removed. See the description of this operand above.

Ethernet_address
Ethernet address of the client to be removed.

boot_file_name
Name of the boot file to be removed.

The `setup_install_server` command supports the following operands:

install_dir_path
The absolute path of the directory in which the Solaris software is to be copied. The directory must be empty.

wanboot_image_path
The absolute path of the directory in which the file containing the WANboot miniroot image is to be created.

install_boot_image_path
Pathname of alternate miniroot, specified with `-t` option.

Examples

EXAMPLE 1 Using `add_install_client`

The following `add_install_client` commands add clients for network installation from a mounted Solaris installation media:

```
example# cd /cdrom/cdrom0/s0/Solaris_10/Tools
example# ./add_install_client system_2/sun4u
```

EXAMPLE 2 Using `add_install_client`

The following `add_install_client` commands add clients for network installation from a mounted Solaris installation media on an install server. The `-c` option specifies a server and path to a JumpStart directory that has a rules file and a profile file for performing a custom JumpStart installation. Also, the Solaris installation media has been copied to the `/export/install` directory:

```
example# cd /export/install/Solaris_10/Tools
example# /add_install_client
           -c install_server:/jumpstart system_1 i86pc\
example# ./add_install_client -c install_server:/jumpstart\
           system_2 i86pc
```

EXAMPLE 3 Using `add_install_client`

The following `add_install_client` command adds support for a specific `sun4u` platform machine (`8:0:20:99:88:77`) using the boot file: `sun4u.solaris10`.

```
example# add_install_client -d -f sun4u.solaris10\
           -e 8:0:20:99:88:77 sun4u
```

EXAMPLE 4 Using `add_install_client`

The following `add_install_client` command adds x86 clients that use the PXE standard for network booting:

```
example# add_install_client -d -s svrname:/mnt/export/root\
           SUNW.i86pc i86p
```

EXAMPLE 5 Using `add_to_install_server`

The following `add_to_install_server` command copies the packages in all the installation media's products directories to an existing install server:

```
example# cd /cdrom/cdrom0/s0
example# ./add_to_install_server /export/Solaris_10
```

EXAMPLE 6 Using `check`

The following `check` command validates the syntax of the rules file used for a custom JumpStart installation:

```
example# cd jumpstart_dir_path
example# ./check -p /cdrom/cdrom0/s0
```

EXAMPLE 7 Using `rm_install_client`

The following `rm_install_client` commands remove clients for network installation:

```
example# cd /export/install/Solaris_10/Tools
example# ./rm_install_client holmes
example# ./rm_install_client watson
```

EXAMPLE 8 Using `setup_install_server`

The following `setup_install_server` command copies the mounted Solaris installation media to a directory named `/export/install` on the local disk:

```
example# cd /cdrom/cdrom0/s0/Solaris_10/Tools
example# ./setup_install_server /export/install
```

EXAMPLE 9 Using `setup_install_server`

The following `setup_install_server` command copies the boot software of a mounted Solaris installation media to a directory named `/boot_dir` on a system that is going to be a boot server for a subnet:

```
example# cd /cdrom/cdrom0/s0/Solaris_10/Tools
example# ./setup_install_server -b /boot_dir
```

EXAMPLE 10 Using `setup_install_server`

By default, `setup_install_server` looks for an installation boot directory at the `Solaris ../Tools/Boot` location of the mount Solaris distribution disc.

If an alternate boot directory is required, such as one saved on a network boot server by way of an earlier `./setup_install_server -b /boot_dir` command, the `-t` option can be used.

```
example# cd /cdrom/cdrom0/s0/Solaris_10/Tools
example# ./setup_install_server -t /boot_dir /export/install
```

EXAMPLE 11 Using `setup_install_server` with WANboot Option

The following `setup_install_server` command creates an image of the WANboot miniroot file systemand stores it in the file `/wanboot_dir/miniroot`.

```
example# cd /cdrom/cdrom0/s0/Solaris_10/Tools
example# ./setup_install_server -w /wanboot_dir /export/install
```

EXAMPLE 12 x86: Specifying a Serial Console to Use During a Network Installation (from Installation Media)

The following example illustrates how to add an x86 install client to an install server and specify a serial console to use during the installation. This example sets up the install client in the following manner:

- The `-d` option indicates that the client is set up to use DHCP to set installation parameters.
- The `-e` option indicates that this installation occurs only on the client with the ethernet address `00:07:e9:04:4a:bf`.
- The first and second uses of the `-b` option instruct the installation program to use the serial port `ttya` as an input and an output device.

```
install server# cd /export/boot/Solaris_10/Tools
install server# ./add_install_client -d -e "00:07:e9:04:4a:bf" \
```

EXAMPLE 12 x86: Specifying a Serial Console to Use During a Network Installation (from Installation Media) *(Continued)*

```
     -b "input-device=ttya" -b "output-device=ttya" \
     i86pc
```

For a complete description of the boot property variables and values you can use with the `-b` option, see `eeprom(1M)`.

EXAMPLE 13 Specifying a Boot Device to Use During a Network Installation (from Installation Media)

The following example illustrates how to add an x86 install client to an install server and specify a boot device to use during the installation. If you specify the boot device when you set up the install client, you are not prompted for this information by the Device Configuration Assistant during the installation.

This example sets up the install client in the following manner:

- The `-d` option indicates that the client is set up to use DHCP to set installation parameters
- The `-e` option indicates that this installation occurs only on the client with the ethernet address `00:07:e9:04:4a:bf`.
- The first and second uses of the `-b` option instruct the installation program to use the serial port `ttya` as an input and an output device.
- The third use of the `-b` option instructs the installation program to use a specific boot device during the installation.
- The value of the boot device path varies based on your hardware..
- The i86pc platform name indicates that the client is an x86-based system.

```
install server# cd /export/boot/Solaris_10/Tools
install server# ./add_install_client -d -e "00:07:e9:04:4a:bf" \
     -b "input-device=ttya" -b "output-device=ttya" \
     -b "bootpath=/pci@0,0/pci108e,16a8@8" i86pc
```

For a complete description of the boot property variables and values you can use with the `-b` option, see `eeprom(1M)`.

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`1`
: An error has occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	Solaris CD (Installation Media)

See Also `uname`(1), `eeprom`(1M), `bootparams`(4), `attributes`(5)

Solaris 10 Installation Guide: Basic Installations

Name install-solaris – install the Solaris operating system

Synopsis

```
install-solaris
```

Description `install-solaris` invokes the Solaris Install program. Depending on graphical capability and available memory at the time of invocation, `install-solaris` invokes either a text-based installer or a graphical installer.

The following minimum requirements for physical memory dictate which features are available during installation:

For SPARC machines:

128 MB Minimum physical memory for all installation types

228 MB Minimum physical memory required to display a window

512 MB Minimum physical memory required for graphical-based installation

For x86 machines:

256 MB Minimum physical memory for all installation types

256 MB Minimum physical memory required for windowing system

512 MB Minimum physical memory required for graphical-based installation

In some cases, even if the minimum physical memory is present, available virtual memory after system startup can limit the number of features available.

`install-solaris` exists only on the Solaris installation media (CD or DVD) and should be invoked only from there. Refer to the for more details.

`install-solaris` allows installation of the operating system onto any standalone system. `install-solaris` loads the software available on the installation media. Refer to the *Solaris 10 Installation Guide: Basic Installations* for disk space requirements.

Usage Refer to the *Solaris 10 Installation Guide: Basic Installations* for more information on the various menus and selections.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcdrom (Solaris installation media)
Interface Stability	Evolving

See Also `pkginfo(1)`, `install(1M)`, `pkgadd(1M)`, `attributes(5)`

Solaris 10 Installation Guide: Basic Installations

Notes It is advisable to exit `install-solaris` by means of the exit options in the `install-solaris` menus.

Name in.stdiscover – Service Tag Discovery Daemon

Synopsis

```
/usr/lib/inet/in.stdiscover
```

Description The `in.stdiscover` daemon allows a mechanism for discovering the location of the Service Tag Listener. By default, the `in.stdiscover` daemon listens for discovery probes (using a minimal built-in protocol) on UDP port 6481.

The daemon is under control of the service management facility, `smf(5)`, under its `inetd` framework. It only runs upon demand and exits when no longer in use.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWservicetagu
Interface Stability	Private

See Also `in.stlisten(1M)`, `stclient(1M)`

Name in.stlisten – Service Tag Listener

Synopsis

```
/usr/lib/inet/in.stlisten
```

Description The `in.stlisten` daemon allows a mechanism for discovering the location of the Service Tag. By default, the `in.stlisten` daemon listens for discovery probes (using a minimal built-in protocol) on TCP port 6481.

The daemon is under control of the service management facility, `smf(5)`, under its `inetd` framework. It only runs upon demand and exits when no longer in use.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWservicetagu
Interface Stability	Private

See Also `in.stdiscover(1M)`, `stclient(1M)`, `svccfg(1M)`, `attributes(5)`, `environ(5)`, `smf(5)`

Notes In open networks where the participants may not always be trusted, it is recommended that you deploy this daemon with the `passphrase-encryption` option. In `smf(5)` environments, the following command can be used to set the *passphrase*:

```
< prepare a text file "passfile" containing the passphrase >
# chown svctag:daemon passfile
# chmod 600 passfile
svccfg -s svc:/network/stlisten \
        setprop servicetag/passphrase=passfile
```

where *passfile* is the path of a file containing the intended *passphrase*.

This *passphrase* can be subsequently cleared as follows:

```
svccfg -s svc:/network/stlisten \
        setprop servicetag/passphrase=""
```

Name in.talkd, talkd – server for talk program

Synopsis

```
in.talkd
```

Description `talkd` is a server used by the `talk(1)` program. It listens at the UDP port indicated in the "talk" service description; see `services(4)`. The actual conversation takes place on a TCP connection that is established by negotiation between the two machines involved.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also `svcs(1)`, `talk(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `services(4)`, `attributes(5)`, `smf(5)`

Notes The protocol is architecture dependent.

The `in.talkd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/talk
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name in.telnetd, telnetd – DARPA TELNET protocol server

Synopsis

```
/usr/sbin/in.telnetd [-a authmode] [-EXUh] [-s tos]
     [-S keytab] [-M realm]
```

Description `in.telnetd` is a server that supports the DARPA standard TELNET virtual terminal protocol. `in.telnetd` is normally invoked in the internet server (see `inetd(1M)`), for requests to connect to the TELNET port as indicated by the `/etc/services` file (see `services(4)`).

`in.telnetd` operates by allocating a pseudo-terminal device for a client, then creating a login process which has the slave side of the pseudo-terminal as its standard input, output, and error. `in.telnetd` manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

When a TELNET session starts up, `in.telnetd` sends TELNET options to the client side indicating a willingness to do *remote* `echo` of characters, and to *suppress go ahead*. The pseudo-terminal allocated to the client is configured to operate in "cooked" mode, and with XTABS, ICRNL and ONLCR enabled. See `termio(7I)`.

`in.telnetd` is willing to do: *echo*, *binary*, *suppress go ahead*, and *timing mark*. `in.telnetd` is willing to have the remote client do: *binary*, *terminal type*, *terminal size*, *logout option*, and *suppress go ahead*.

`in.telnetd` also allows environment variables to be passed, provided that the client negotiates this during the initial option negotiation. The DISPLAY environment variable may be sent this way, either by the TELNET general environment passing methods, or by means of the XDISPLOC TELNET option. DISPLAY can be passed in the environment option during the same negotiation where XDISPLOC is used. Note that if you use both methods, use the same value for both. Otherwise, the results may be unpredictable.

These options are specified in Internet standards *RFC 1096*, *RFC 1408*, *RFC 1510*, *RFC 1571*, *RFC 2941*, *RFC 2942*, *RFC 2946*, and *RFC 1572*. The following Informational draft is also supported: *RFC 2952*.

The banner printed by `in.telnetd` is configurable. The default is (more or less) equivalent to ‘`uname -sr`‘ and will be used if no banner is set in `/etc/default/telnetd`. To set the banner, add a line of the form

```
BANNER="..."
```

to `/etc/default/telnetd`. Nonempty banner strings are fed to shells for evaluation. The default banner may be obtained by

```
BANNER="\\r\\n\\r\\n‘uname -s‘ ‘uname -r‘\\r\\n\\r\\n"
```

and no banner will be printed if `/etc/default/telnetd` contains

```
BANNER=""
```

Options The following options are supported:

`-a` *authmode* This option may be used for specifying what mode should be used for authentication. There are several valid values for *authmode*:

- `valid` Only allows connections when the remote user can provide valid authentication information to identify the remote user, and is allowed access to the specified account without providing a password.
- `user` Only allows connections when the remote user can provide valid authentication information to identify the remote user. The `login(1)` command will provide any additional user verification needed if the remote user is not allowed automatic access to the specified account.
- `none` This is the default state. Authentication information is not required. If no or insufficient authentication information is provided, then the `login`(1) program provides the necessary user verification.
- `off` This disables the authentication code. All user verification happens through the `login`(1) program.

`-E` Disables encryption support negotiation.

`-h` Disables displaying host specific information before login has been completed.

`-M` *realm* Uses the indicated Kerberos V5 realm. By default, the daemon will determine its realm from the settings in the `krb5.conf(4)` file.

`-s` *tos* Sets the `IP TOS` option.

`-S` *keytab* Sets the `KRB5` keytab file to use. The `/etc/krb5/krb5.keytab` file is used by default.

`-U` Refuses connections that cannot be mapped to a name through the `getnameinfo(3SOCKET)` function.

`-X` Disables Kerberos V5 authentication support negotiation.

Usage `telnetd` and `in.telnetd` are IPv6–enabled. See `ip6(7P)`.

Security `in.telnetd` can authenticate using Kerberos V5 authentication, `pam(3PAM)`, or both. By default, the telnet server will accept valid Kerberos V5 authentication credentials from a `telnet` client that supports Kerberos. `in.telnetd` can also support an encrypted session from such a client if the client requests it.

The telnet protocol only uses single DES for session protection—clients request service tickets with single DES session keys. The KDC must know that host service principals that offer the telnet service support single DES, which, in practice, means that such principals must have single DES keys in the KDC database.

In order for Kerberos authentication to work, a host/<*FQDN*> Kerberos principal must exist for each Fully Qualified Domain Name associated with the telnetd server. Each of these host/<*FQDN*> principals must have a keytab entry in the /etc/krb5/krb5.keytab file on the telnetd server. An example principal might be:

```
host/bigmachine.eng.example.com
```

See kadmin(1M) or gkadmin(1M) for instructions on adding a principal to a krb5.keytab file. See *System Administration Guide: Security Services* for a discussion of Kerberos authentication.

in.telnetd uses pam(3PAM) for authentication, account management, session management, and password management. The PAM configuration policy, listed through /etc/pam.conf, specifies the modules to be used for in.telnetd. Here is a partial pam.conf file with entries for the telnet command using the UNIX authentication, account management, session management, and password management modules.

```
telnet  auth requisite          pam_authtok_get.so.1
telent  auth required           pam_dhkeys.so.1
telent  auth required           pam_unix_auth.so.1

telnet  account requisite       pam_roles.so.1
telnet  account required        pam_projects.so.1
telnet  account required        pam_unix_account.so.1

telnet  session required        pam_unix_session.so.1

telnet  password required       pam_dhkeys.so.1
telent  password requisite      pam_authtok_get.so.1
telnet  password requisite      pam_authtok_check.so.1
telnet  password required       pam_authtok_store.so.1
```

If there are no entries for the telnet service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords.

For a Kerberized telnet service, the correct PAM service name is ktelnet.

Files /etc/default/telnetd

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtnetd

See Also `login(1)`, `svcs(1)`, `telnet(1)`, `gkadmin(1M)`, `inetadm(1M)`, `inetd(1M)`, `kadmin(1M)`, `svcadm(1M)`, `pam(3PAM)`, `getnameinfo(3SOCKET)`, `issue(4)`, `krb5.conf(4)`, `pam.conf(4)`, `services(4)`, `attributes(5)`, `pam_authtok_check(5)`, `pam_authtok_get(5)`, `pam_authtok_store(5)`, `pam_dhkeys(5)`, `pam_passwd_auth(5)`, `pam_unix_account(5)`, `pam_unix_auth(5)`, `pam_unix_session(5)`, `smf(5)`, `ip6(7P)`, `termio(7I)`

System Administration Guide: Security Services

Alexander, S. *RFC 1572, TELNET Environment Option*. Network Information Center, SRI International, Menlo Park, Calif., January 1994.

Borman, Dave. *RFC 1408, TELNET Environment Option*. Network Information Center, SRI International, Menlo Park, Calif., January 1993.

Borman, Dave. *RFC 1571, TELNET Environment Option Interoperability Issues*. Network Information Center, SRI International, Menlo Park, Calif., January 1994.

Crispin, Mark. *RFC 727, TELNET Logout Option*. Network Information Center, SRI International, Menlo Park, Calif., April 1977.

Marcy, G. *RFC 1096, TELNET X Display Location Option*. Network Information Center, SRI International, Menlo Park, Calif., March 1989.

Postel, Jon, and Joyce Reynolds. *RFC 854, TELNET Protocol Specification*. Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Waitzman, D. *RFC 1073, TELNET Window Size Option*. Network Information Center, SRI International, Menlo Park, Calif., October 1988.

Kohl, J., Neuman, C., *The Kerberos Network Authentication Service (V5), RFC 1510*. September 1993.

Ts'o, T. and J. Altman, *Telnet Authentication Option, RFC 2941*. September 2000.

Ts'o, T., *Telnet Authentication: Kerberos Version 5, RFC 2942*. September 2000.

Ts'o, T., *Telnet Data Encryption Option, RFC 2946*. September 2000.

Ts'o, T., *Telnet Encryption: DES 64 bit Cipher Feedback, RFC 2952*. September 2000.

Notes Some TELNET commands are only partially implemented.

Binary mode has no common interpretation except between similar operating systems.

The terminal type name received from the remote client is converted to lower case.

The *packet* interface to the pseudo-terminal should be used for more intelligent flushing of input and output queues.

`in.telnetd` never sends TELNET *go ahead* commands.

The `pam_unix`(5) module is no longer supported.. Similar functionality is provided by `pam_authtok_check`(5), `pam_authtok_get`(5), `pam_authtok_store`(5), `pam_dhkeys`(5), `pam_passwd_auth`(5), `pam_unix_account`(5), `pam_unix_auth`(5), and `pam_unix_session`(5).

The `in.telnetd` service is managed by the service management facility, `smf`(5), under the service identifier:

```
svc:/network/telnet
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm`(1M). Responsibility for initiating and restarting this service is delegated to `inetd`(1M). Use `inetadm`(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs`(1) command.

Name in.tftpd, tftpd – Internet Trivial File Transfer Protocol server

Synopsis

```
in.tftpd [-d] [-T rexmtval] [-s] [homedir]
```

Description `tftpd` is a server that supports the Internet Trivial File Transfer Protocol (TFTP).

Before responding to a request, the server attempts to change its current directory to *homedir*; the default directory is `/tftpboot`.

The use of `tftp` does not require an account or password on the remote system. Due to the lack of authentication information, `in.tftpd` will allow only publicly readable files to be accessed. Files may be written only if they already exist and are publicly writable. Note that this extends the concept of "public" to include all users on all hosts that can be reached through the network. This may not be appropriate on all systems, and its implications should be considered before enabling this service.

`in.tftpd` runs with the user ID and group ID set to `[GU]ID_NOBODY` under the assumption that no files exist with that owner or group. However, nothing checks this assumption or enforces this restriction.

Options

`-d` Debug. When specified it sets the `SO_DEBUG` socket option.

`-s` Secure. When specified, the directory change to *homedir* must succeed. The daemon also changes its root directory to *homedir*.

`-T` *rexmtval* Specifies the value of the retransmission timeout in seconds. This also affects the maximum session timeout in that the latter is set to five times the retransmission timeout value.

Usage The `in.tftpd` server is IPv6–enabled. See `ip6(7P)`.

`in.tftpd` supports transfers of greater than 32 MB, per RFC 2348.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtftp

See Also `svcs(1)`, `tftp(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `netconfig(4)`, `attributes(5)`, `smf(5)`, `ip6(7P)`

Malkin, G. and Harkin, A. *RFC 2347, TFTP Option Extension*. The Internet Society. May 1998

Malkin, G. and Harkin, A. *RFC 2348, TFTP Blocksize Option*. The Internet Society. May 1998

Malkin, G. and Harkin, A. *RFC 2349, TFTP Timeout Interval and Transfer Size Options*. The Internet Society. May 1998

Sollins, K.R. *RFC 1350, The TFTP Protocol (Revision 2)*. Network Working Group. July 1992.

Notes The `tftpd` server only acknowledges the transfer size option that is sent with a `read` request when the octet transfer mode is specified.

The `in.tftpd.1m` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/tftp/udp6:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Unlike most `smf(5)` services, a manifest for the `tftp` service is not included in the system. To create one and enable this service, the administrator should:

1. Edit `/etc/inet/inetd.conf` and uncomment the `tftp` entry.
2. Run `/usr/sbin/inetconv`.

After you run `inetconv`, the `svc:/network/tftp/udp6:default` service is created and enabled.

Name in.timed – UDP or TCP time protocol service daemon

Synopsis

```
in.timed

FMRI svc:/internet/time:default
```

Description FMRI stands for Fault Management Resource Identifier. It is used to identify resources managed by the Fault Manager. See `fmd(1M)` and `smf(5)`.

The `in.timed` service provides the server-side of the time protocol. The time server sends to requestors the time in seconds since midnight, January 1, 1900. The time protocol is available on both TCP and UDP transports through port 37.

The in.timed service is an `inetd(1M)` `smf(5)` delegated service. The `in.timed` detects which transport is requested by examining the socket it is passed by the `inetd` daemon.

TCP-based service — Once a connection is established, the `in.timed` sends the time as a 32-bit binary number and closes the connection. Any received data is ignored.

UDP-based service — The `in.timed` listens for UDP datagrams. When a datagram is received, the server generates a UDP datagram containing the time as a 32–bit binary number and sends it to the client. Any received data is ignored.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcnsu
Interface Stability	Evolving

See Also `inetd(1M)`, `attributes(5)`, `smf(5)`

RFC 868

Name in.tnamed, tnamed – DARPA trivial name server

Synopsis

```
/usr/sbin/in.tnamed [-v]
```

Description `in.tnamed` is a server that supports the DARPA Name Server Protocol. The name server operates at the port indicated in the "name" service description (see `services(4)`), and is invoked by `inetd(1M)` when a request is made to the name server.

Options

`-v` Invoke the daemon in verbose mode.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtnamd

See Also `svcs(1)`, `uucp(1C)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `services(4)`, `attributes(5)`, `smf(5)`

Postel, Jon, *Internet Name Server*, IEN 116, SRI International, Menlo Park, California, August 1979.

Notes The protocol implemented by this program is obsolete. Its use should be phased out in favor of the Internet Domain Name Service (DNS) protocol.

The `in.tnamed` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/tname
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name intrstat – report interrupt statistics

Synopsis

```
/usr/sbin/intrstat [-c cpulist | -C processor_set_id]
     [interval [count]]
```

Description The `intrstat` utility gathers and displays run-time interrupt statistics. The output is a table of device names and CPU IDs, where each row of the table denotes a device, and each column of the table denotes a CPU. Each cell in the table contains both the raw number of interrupts for the given device on the given CPU, and the percentage of absolute time spent in that device's interrupt handler on that CPU.

The device name is given in the form of {*name*}#{*instance*}. The name is the normalized driver name, and typically corresponds to the name of the module implementing the driver. See `ddi_driver_name(9F)`. Many Sun-delivered drivers have their own manual pages. See `Intro(7)`.

If standard output is a terminal, the table contains as many columns of data as can fit within the terminal width. If standard output is not a terminal, the table contains at most four columns of data. By default, data is gathered and displayed for all CPUs. If the data cannot fit in a single table, it is printed across multiple tables. The set of CPUs for which data is displayed can be optionally specified with the `-c` or `-C` option.

By default, `intrstat` displays data once per second and runs indefinitely. Both of these behaviors can be optionally controlled with the interval and count parameters, respectively. See `OPERANDS`.

Because `intrstat` uses dynamic discovery, it reports only on devices that raise interrupts while the command is running. Any devices that are silent while `intrstat` is running are not displayed.

`intrstat` induces a small system-wide performance degradation. As a result, only the super-user can run `intrstat` by default. The *Solaris Dynamic Tracing Guide* explains how administrators can grant privileges to other users to permit them to run `intrstat`.

Options The following options are supported:

`-c` *cpulist*
Displays data for the CPUs specified by *cpulist*.

cpulist can be a single processor ID (for example, `4`), a range of processor IDs (for example, `4-6`), or a comma separated list of processor IDs or processor ID ranges (for example, `4,5,6` or `4,6-8`).

`-C` *processor_set_id*
Displays data for the CPUs in the processor set specified by *processor_set_id*.

`intrstat` modifies its output to always reflect the CPUs in the specified processor set. If a CPU is added to the set, `intrstat` modifies its output to include the added CPU. If a CPU is removed from the set, `intrstat` modifies its output to exclude the removed CPU. At

most one processor set can be specified.

Operands The following operands are supported:

count
Indicates the number of intervals to execute before exiting.

interval
Indicates the number of seconds to be executed before exiting.

Examples **EXAMPLE 1** Using `intrstat` Without Options

Without options, `intrstat` displays a table of trap types and CPUs. At most, four columns can fit in the default terminal width. If there are more than four CPUs, multiple tables are displayed.

The following example runs `intrstat` on a uniprocessor Intel IA/32-based laptop:

```
example# intrstat
               device |      cpu0 %tim
     -----------------+---------------
                ata#0 |       166  0.4
                ata#1 |         0  0.0
           audioi810#0 |         6  0.0
               i8042#0 |       281  0.7
                iprb#0 |         6  0.0
                uhci#1 |         6  0.0
                uhci#2 |         6  0.0

               device |      cpu0 %tim
     -----------------+---------------
                ata#0 |       161  0.5
                ata#1 |         0  0.0
           audioi810#0 |         6  0.0
               i8042#0 |       303  0.6
                iprb#0 |         6  0.0
                uhci#1 |         6  0.0
                uhci#2 |         6  0.0
     ...
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdtrc
Interface Stability	See below.

The command-line syntax is Evolving. The human-readable output is Unstable.

See Also dtrace(1M), trapstat(1M), attributes(5), Intro(7), ddi_driver_name(9F)

Solaris Dynamic Tracing Guide

Name in.uucpd, uucpd – UUCP server

Synopsis

```
/usr/sbin/in.uucpd [-n]
```

Description `in.uucpd` is the server for supporting UUCP connections over networks.

`in.uucpd` is invoked by `inetd(1M)` when a UUCP connection is established, that is, a connection to the port indicated in the "uucp" service specification, and executes the following protocol. See `services(4)`:

1. The server prompts with `login:`. The `uucico(1M)` process at the other end must supply a username.
2. Unless the username refers to an account without a password, the server then prompts with `Password:`. The `uucico` process at the other end must supply the password for that account.

If the username is not valid, or is valid but refers to an account that does not have `/usr/lib/uucp/uucico` as its login shell, or if the password is not the correct password for that account, the connection is dropped. Otherwise, `uucico` is run, with the user ID, group ID, group set, and home directory for that account, with the environment variables `USER` and `LOGNAME` set to the specified username, and with a `-u` flag specifying the username. Unless the `-n` flag is specified, entries are made in `/var/adm/utmpx`, `/var/adm/wtmpx`, and `/var/adm/lastlog` for the username. `in.uucpd` must be invoked by a user with appropriate privilege (usually root) in order to be able to verify that the password is correct.

Security `in.uucpd` uses `pam(3PAM)` for authentication, account management, and session management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the modules to be used for `in.uucpd`. Here is a partial `pam.conf` file with entries for `uucp` using the UNIX authentication, account management, and session management module.

```
uucp    auth requisite          pam_authtok_get.so.1
uucp    auth required           pam_dhkeys.so.1
uucp    auth required           pam_unix_auth.so.1

uucp    account requisite       pam_roles.so.1
uucp    account required        pam_projects.so.1
uucp    account required        pam_unix_account.so.1

uucp      session required      pam_unix_session.so.1
```

If there are no entries for the `uucp` service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the peer may be prompted for multiple passwords.

Files

`/var/adm/utmpx`	accounting
`/var/adm/wtmpx`	accounting
`/var/adm/lastlog`	time of last login

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

See Also svcs(1), inetadm(1M), inetd(1M), svcadm(1M), uucico(1M), pam(3PAM), pam.conf(4), services(4), attributes(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), smf(5)

Diagnostics All diagnostic messages are returned on the connection, after which the connection is closed.

`user read` An error occurred while reading the username.

`passwd read` An error occurred while reading the password.

`Login incorrect.` The username is invalid or refers to an account with a login shell other than `/usr/lib/uucp/uucico`, or the password is not the correct password for the account.

Notes The `in.uucpd` service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/uucp
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). Responsibility for initiating and restarting this service is delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

The pam_unix(5) module is no longer supported. Similar functionality is provided by pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), and pam_unix_session(5).

Name iostat – report I/O statistics

Synopsis

```
/usr/bin/iostat  [-cCdDeEiImMnpPrstxXYz] [-l n] [-T u | d]
 [disk]... [interval [count]]
```

Description The `iostat` utility iteratively reports terminal, disk, and tape I/O activity, as well as CPU utilization. The first line of output is for all time since boot; each subsequent line is for the prior interval only.

To compute this information, the kernel maintains a number of counters. For each disk, the kernel counts reads, writes, bytes read, and bytes written. The kernel also takes hi-res time stamps at queue entry and exit points, which allows it to keep track of the residence time and cumulative residence-length product for each queue. Using these values, `iostat` produces highly accurate measures of throughput, utilization, queue lengths, transaction rates and service time. For terminals collectively, the kernel simply counts the number of input and output characters.

During execution of the kernel status command, the `state` of the system can change. If relevant, a state change message is included in the `iostat` output, in one of the following forms:

```
<<device added: sd0>>
<<device removed: sd0>>
<<partition added: sd0,a>>
<<partition removed: sd0,a>>
<<NFS mounted: nfs1>>
<<NFS unmounted: nfs1>>
<<multi-path added: ssd4>>
<<multi-path removed: ssd4>>
<<controller added: c1>>
<<controller removed: c1>>
<<processors added: 1, 3>>
<<processors removed: 1, 3>>
```

Note that the `names` printed in these state change messages are affected by the `-n` and `-m` options as appropriate.

For more general system statistics, use `sar(1)`, `sar(1M)`, or `vmstat(1M)`.

Output The output of the `iostat` utility includes the following information.

`device` name of the disk

`r/s` reads per second

`w/s` writes per second

`kr/s` kilobytes read per second

The average I/O size during the interval can be computed from `kr/s` divided by `r/s`.

`kw/s` kilobytes written per second

The average I/O size during the interval can be computed from `kw/s` divided by `w/s`.

`wait` average number of transactions waiting for service (queue length)

This is the number of I/O operations held in the device driver queue waiting for acceptance by the device.

`actv` average number of transactions actively being serviced (removed from the queue but not yet completed)

This is the number of I/O operations accepted, but not yet serviced, by the device.

`svc_t` average response time of transactions, in milliseconds

The `svc_t` output reports the overall *response* time, rather than the *service* time, of a device. The overall time includes the time that transactions are in queue and the time that transactions are being serviced. The time spent in queue is shown with the `-x` option in the `wsvc_t` output column. The time spent servicing transactions is the true service time. Service time is also shown with the `-x` option and appears in the `asvc_t` output column of the same report.

`%w` percent of time there are transactions waiting for service (queue non-empty)

`%b` percent of time the disk is busy (transactions in progress)

`wsvc_t` average service time in wait queue, in milliseconds

`asvc_t` average service time of active transactions, in milliseconds

`wt` the I/O wait time is no longer calculated as a percentage of CPU time, and this statistic will always return zero.

Options The following options are supported:

`-c` Report the percentage of time the system has spent in user mode, in system mode, waiting for I/O, and idling. See the NOTES section for more information.

`-C` When the `-x` option is also selected, report extended disk statistics aggregated by *controller id*.

`-d` For each disk, report the number of kilobytes transferred per second, the number of transfers per second, and the average service time in milliseconds.

`-D` For each disk, report the reads per second, writes per second, and percentage disk utilization.

`-e` Display device error summary statistics. The total errors, hard errors, soft errors, and transport errors are displayed.

`-E` Display all device error statistics.

`-i` In `-E` output, display the `Device ID` instead of the `Serial No`. The `Device Id` is a unique identifier registered by a driver through `ddi_devid_register(9F)`.

`-I` Report the counts in each interval, rather than rates (where applicable).

`-l` *n* Limit the number of disks included in the report to *n*; the disk limit defaults to 4 for `-d` and `-D`, and unlimited for `-x`. Note: disks explicitly requested (see *disk* below) are not subject to this disk limit.

`-m` Report file system mount points. This option is most useful if the `-P` or `-p` option is also specified or used in conjunction with `-Xn` or `-en`. The `-m` option is useful only if the mount point is actually listed in the output. This option can only be used in conjunction with the `-n` option.

`-M` Display data throughput in MB/sec instead of KB/sec.

`-n` Display names in descriptive format. For example, `cXtYdZ`, `rmt/N`, `server:/export/path`.

By default, disks are identified by instance names such as `ssd23` or `md301`. Combining the `-n` option with the `-x` option causes disk names to display in the `cXtYdZsN` format which is more easily associated with physical hardware characteristics. The `cXtYdZsN` format is particularly useful in FibreChannel (FC) environments where the FC World Wide Name appears in the `t` field.

`-p` For each disk, report per-partition statistics in addition to per-device statistics.

`-P` For each disk, report per-partition statistics only, no per-device statistics.

`-r` Display data in a comma-separated format.

`-s` Suppress messages related to `state changes`.

`-t` Report the number of characters read and written to terminals per second.

`-T u | d` Display a time stamp.

Specify `u` for a printed representation of the internal representation of time. See `time(2)`. Specify `d` for standard date format. See `ctime(3C)`.

`-x` Report extended disk statistics. By default, disks are identified by instance names such as `ssd23` or `md301`. Combining the `x` option with the `-n` option causes disk names to display in the `cXtYdZsN` format, more easily associated with physical hardware characteristics. Using the `cXtYdZsN` format is particularly helpful in the FibreChannel environments where the FC World Wide Name appears in the `t` field.

If no output display is requested (no `-x`, `-e`, `-E`), `-x` is implied.

`-Y` For disks under `scsi_vhci(7D)` control, in addition to disk *lun* statistics, also report statistics for *lun.targetport* and *lun.targetport.controller*.

In `-n` (descriptive) mode the *targetport* is shown in using the `target-port` property of the path. Without `-n` the *targetport* is shown using the shorter `port-id`. All target ports with the same `target-port` property value share the same `port-id`. The `target-port-to-port-id` association does not persist across reboot.

If no output display is requested (no `-x`, `-e`, `-E`), `-x` is implied.

`-z` Do not print lines whose underlying data values are all zeros.

The option set `-xcnCXTdz` *interval* is particularly useful for determining whether disk I/O problems exist and for identifying problems.

Operands

The following operands are supported:

count Display only *count* reports.

disk Explicitly specify the disks to be reported; in addition to any explicit disks, any active disks up to the disk limit (see `-l` above) will also be reported.

interval Report once each *interval* seconds.

Examples

EXAMPLE 1 Using `iostat` to Generate User and System Operation Statistics

The following command displays two reports of extended device statistics, aggregated by *controller id*, for user (`us`) and system (`sy`) operations. Because the `-n` option is used with the `-x` option, devices are identified by controller names.

```
example% iostat -xcnCXTdz 5


Mon Nov 24 14:58:36 2003
    cpu
 us sy wt id
 14 31  0 20
                    extended device statistics
  r/s    w/s    kr/s      kw wait  actv wsvc_t asvc_t  %w  %b device
  3.8   29.9   145.8    44.0  0.0   0.2    0.1    6.4   0   5     c0
666.3  814.8 12577.6 17591.1 91.3  82.3   61.6   55.6   0   2    c12
180.0  234.6  4401.1  5712.6  0.0 147.7    0.0  356.3   0  98    d10

Mon Nov 24 14:58:41 2003
    cpu
 us sy wt id
 11 31  0 22
```

EXAMPLE 1 Using `iostat` to Generate User and System Operation Statistics *(Continued)*

```
                  extended device statistics
  r/s    w/s    kr/s      kw wait  actv wsvc_t asvc_t  %w  %b device
  0.8   41.0     5.2    20.5 0.0    0.2    0.2    4.4   0   6     c0
565.3  581.7  8573.2 10458.9 0.0   26.6    0.0   23.2   0   3    c12
106.5   81.3  3393.2  1948.6 0.0    5.7    0.0   30.1   0  99    d10
```

EXAMPLE 2 Using `iostat` to Generate TTY Statistics

The following command displays two reports on the activity of five disks in different modes of operation. Because the `-x` option is used, disks are identified by instance names.

```
example% iostat -x tc 5 2
```

```
                  extended device statistics        tty         cpu
device r/s  w/s kr/s  kw/s wait actv svc_t %w  %b  tin tout  us sy wt id
sd0    0.4  0.3 10.4   8.0  0.0  0.0  36.9  0   1    0   10   0  0  0 99
sd1    0.0  0.0  0.3   0.4  0.0  0.0  35.0  0   0
sd6    0.0  0.0  0.0   0.0  0.0  0.0   0.0  0   0
nfs1   0.0  0.0  0.0   0.0  0.0  0.0   0.0  0   0
nfs2   0.0  0.0  0.0   0.1  0.0  0.0  35.6  0   0
            extended device statistics              tty         cpu
device r/s  w/s  kr/s  kw/s wait actv svc_t %w  %b tin tout  us sy wt id
sd0    0.0  0.0  0.0   0.0  0.0  0.0  0.0   0   0   0  155   0  0  0 100
sd1    0.0  0.0  0.0   0.0  0.0  0.0  0.0   0   0
sd6    0.0  0.0  0.0   0.0  0.0  0.0  0.0   0   0
nfs1   0.0  0.0  0.0   0.0  0.0  0.0  0.0   0   0
nfs2   0.0  0.0  0.0   0.0  0.0  0.0  0.0   0   0
```

EXAMPLE 3 Using `iostat` to Generate Partition and Device Statistics

The following command generates partition and device statistics for each disk. Because the `-n` option is used with the `-x` option, disks are identified by controller names.

```
example% iostat -xnp
```

```
                extended device statistics
r/s  w/s  kr/s kw/s wait actv wsvc_t asvc_t  %w %b device
0.4  0.3  10.4  7.9  0.0  0.0    0.0   36.9   0  1 c0t0d0
0.3  0.3   9.0  7.3  0.0  0.0    0.0   37.2   0  1 c0t0d0s0
0.0  0.0   0.1  0.5  0.0  0.0    0.0   34.0   0  0 c0t0d0s1
0.0  0.0   0.0  0.1  0.0  0.0    0.6   35.0   0  0 fuji:/export/home/user3
```

EXAMPLE 4 Show Translation from Instance Name to Descriptive Name

The following example illustrates the use of `iostat` to translate a specific instance name to a descriptive name.

```
example% iostat -xn sd1
                        extended device statistics
r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 c8t1d0
```

EXAMPLE 5 Show Target Port and Controller Activity for a Specific Disk

In the following example, there are four controllers, all connected to the same target port.

```
# iostat -Y ssd22
                    extended device statistics
device          r/s    w/s   kr/s   kw/s wait actv  svc_t  %w  %b
ssd22           0.2    0.0    1.5    0.0  0.0  0.0    0.7   0   0
ssd22.t2        0.2    0.0    1.5    0.0  0.0  0.0    0.0   0   0
ssd22.t2.fp0    0.0    0.0    0.4    0.0  0.0  0.0    0.0   0   0
ssd22.t2.fp1    0.0    0.0    0.4    0.0  0.0  0.0    0.0   0   0
ssd22.t2.fp2    0.0    0.0    0.4    0.0  0.0  0.0    0.0   0   0
ssd22.t2.fp3    0.0    0.0    0.4    0.0  0.0  0.0    0.0   0   0
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	See below.

Invocation is evolving. Human readable output is unstable.

See Also

`sar(1)`, `sar(1M)`, `mpstat(1M)`, `vmstat(1M)`, `time(2)`, `ctime(3C)`, `attributes(5)`, `scsi_vhci(7D)`

System Administration Guide: Basic Administration

Notes

The sum of CPU utilization might vary slightly from 100 because of rounding errors in the production of a percentage figure.

The `svc_t` response time is not particularly significant when the `I/0` (`r/s+w/s`) rates are under 0.5 per second. Harmless spikes are fairly normal in such cases.

The `mpstat` utility reports the same `wt`, `usr`, and `sys` statistics. See `mpstat(1M)` for more information.

When executed in a `zone` and if the pools facility is active, `iostat`(1M) will only provide information for those processors in the processor set of the pool to which the `zone` is bound.

Name ipaddrsel – configure IPv6 default address selection

Synopsis

```
/usr/sbin/ipaddrsel

/usr/sbin/ipaddrsel  -f file

/usr/sbin/ipaddrsel  -d
```

Description Use the `ipaddrsel` utility to configure the IPv6 default address selection policy table. The policy table is a longest-matching-prefix lookup table that is used for IPv6 source address selection and for destination address ordering when resolving names to `AF_INET6` addresses. For a description of how the policy table is used for source address selection, see `inet6(7P)`. For a description of how the policy table is used for destination address ordering, see `getaddrinfo(3SOCKET)`.

The unmodified policy table is valid for all typical IPv6 deployments. Modify the table only if a circumstance exists for which the default behavior of the IPv6 source address selection or destination address ordering mechanism is unsatisfactory. See the Examples section for examples of such circumstances. You should carefully consider your addressing strategy before you change the table from the provided default.

When the `ipaddrsel` command is issued without any arguments, the address selection policy currently in use is printed. The format of the output is compatible with the format of the configuration file that the `-f` option accepts.

Note – If the `usesrc` subcommand to `ifconfig(1M)` is applied to a particular physical interface, the selection policy specified by `usesrc` overrides the source address selection policies specified by `ipaddrsel`. This is true for packets that are locally generated and for applications that do not choose a non-zero source address using `bind(3SOCKET)`.

The Configuration File

The configuration file that the `-f` option accepts can contain either comment lines or policy entries. Comment lines have a '`#`' character as the first non-blank character. and they are ignored by the `ipaddrsel` utility. Policy entry lines have the following format:

prefix/prefix_length precedence label `[#` *comment* `]`

The *prefix* must be an IPv6 prefix in a format consistent with `inet(3SOCKET)`. The *prefix_length* is an integer ranging from 0 to 128. The IPv6 source address selection and destination address ordering algorithms determine the precedence or label of an address by doing a longest-prefix-match lookup using the prefixes in this table, much like next-hop determination for a destination is done by doing a longest-prefix-match lookup using an IP routing table.

The precedence is a non-negative integer that represents how the destination address ordering mechanism will sort addresses returned from name lookups. In general, addresses with a higher precedence will be in front of addresses with a lower precedence. Other factors, such as destinations with undesirable source addresses can, however, override these precedence values.

The label is a string of at most fifteen characters, not including the NULL terminator. The label allows particular source address prefixes to be used with destination prefixes of the same label. Specifically, for a particular destination address, the IPv6 source address selection algorithm prefers source addresses whose label is equal that of the destination.

The label may be followed by an optional comment.

The file must contain a default policy entry, which is an entry with "`::0/0`" as its *prefix* and *prefix_length*. This is to ensure that all possible addresses match a policy.

Options The `ippadrsel` utility supports the following options:

`-f` *file* — Replace the address selection policy table with the policy specified in the *file*.

`-d` — Revert the kernel's address selection policy table back to the default table. Invoking `ipaddrsel` in this way only changes the currently running kernel's policy table, and does not alter the configuration file `/etc/inet/ipaddrsel.conf`. To revert the configuration file back to its default settings, use `ipaddrsel -d`, then dump the contents of the table to the configuration file by redirecting the output of `ipaddrsel` to `/etc/inet/ipaddrsel.conf`.

```
example# ipaddrsel -d
example# ipaddrsel > /etc/inet/ipaddrsel.conf
```

Examples **EXAMPLE 1** The Default Policy in `/etc/inet/ipaddrsel.conf`

The following example is the default policy that is located in `/etc/inet/ipaddrsel.conf`:

```
# Prefix                  Precedence Label
::1/128                          50 Loopback
::/0                             40 Default
2002::/16                        30 6to4
::/96                            20 IPv4_Compatible
::ffff:0.0.0.0/96                10 IPv4
```

EXAMPLE 2 Assigning a Lower Precedence to Link-local and Site-local Addresses

By default, the destination address ordering rules sort addresses of smaller scope before those of larger scope. For example, if a name resolves to a global and a site-local address, the site local address would be ordered before the global address. An administrator can override this ordering rule by assigning a lower precedence to addresses of smaller scope, as the following table demonstrates.

```
# Prefix                  Precedence Label
::1/128                          50 Loopback
::/0                             40 Default
2002::/16                        30 6to4
fec0::/10                        27 Site-Local
```

EXAMPLE 2 Assigning a Lower Precedence to Link-local and Site-local Addresses *(Continued)*

```
fe80::/10                       23 Link-Local
::/96                           20 IPv4_Compatible
::ffff:0.0.0.0/96               10 IPv4
```

EXAMPLE 3 Assigning Higher Precedence to IPv4 Destinations

By default, IPv6 addresses are ordered in front of IPv4 addresses in name lookups. `::ffff:0.0.0.0/96` has the lowest precedence in the default table. In the following example, IPv4 addresses are assigned higher precedence and are ordered in front of IPv6 destinations:

```
# Prefix                Precedence Label
::1/128                         50 Loopback
::/0                            40 Default
2002::/16                       30 6to4
::/96                           20 IPv4_Compatible
::ffff:0.0.0.0/96               60 IPv4
```

EXAMPLE 4 Ensuring that a particular source address is only used when communicating with destinations in a particular network.

The following policy table assigns a label of 5 to a particular source address on the local system, `2001:1111:1111::1`. The table assigns the same label to a network, `2001:2222:2222::/48`. The result of this policy is that the `2001:1111:1111::1` source address will only be used when communicating with destinations contained in the `2001:2222:2222::/48` network. For this example, this network is the "`ClientNet`", which could represent a particular client's network.

```
# Prefix                Precedence Label
::1/128                         50 Loopback
2001:1111:1111::1/128           40 ClientNet
2001:2222:2222::/48             40 ClientNet
::/0                            40 Default
2002::/16                       30 6to4
::/96                           20 IPv4_Compatible
::ffff:0.0.0.0/96               10 IPv4
```

This example assumes that the local system has one physical interface, and that all global prefixes are assigned to that physical interface.

Exit Status

`ipaddrsel` returns the following exit values:

`0` `ipaddrsel` successfully completed.

`>0` An error occurred. If a failure is encountered, the kernel's current policy table is unchanged.

Files `/etc/inet/ipaddrsel.conf` The file that contains the IPv6 default address selection policy to be installed at boot time. This file is loaded before any Internet services are started.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also `nscd(1M)`, `inet(3SOCKET)`, `getaddrinfo(3SOCKET)`, `ipaddrsel.conf(4)`, `attributes(5)`, `inet6(7P)`

Notes The ipnodes cache kept by `nscd(1M)` contains addresses that are ordered using the destination address ordering algorithm, which is one of the reasons why `ipaddrsel` is called before `nscd` in the boot sequence. If `ipaddrsel` is used to change the address selection policy after `nscd` has started, you should invalidate the `nscd` ipnodes cache invalidated by invoking the following command:

```
example# /usr/sbin/nscd -i ipnodes
```

Name ipf – alter packet filtering lists for IP packet input and output

Synopsis

```
ipf [-6AdDEInoPRrsvVyzZ] [-l block | pass | nomatch]
     [-T optionlist] [-F i | o | a | s | S] -f filename
     [-f filename...]
```

Description The `ipf` utility is part of a suite of commands associated with the Solaris IP Filter feature. See `ipfilter(5)`.

The `ipf` utility opens the filenames listed (treating a hyphen (`-`) as stdin) and parses the file for a set of rules which are to be added or removed from the packet filter rule set.

If there are no parsing problems, each rule processed by `ipf` is added to the kernel's internal lists. Rules are added to the end of the internal lists, matching the order in which they appear when given to `ipf`.

`ipf`'s use is restricted through access to `/dev/ipauth`, `/dev/ipl`, and `/dev/ipstate`. The default permissions of these files require `ipf` to be run as root for all operations.

Enabling Solaris IP Filter Feature

Solaris IP Filter is installed with the Solaris operating system. However, packet filtering is not enabled by default. Use the following procedure to activate the Solaris IP Filter feature.

1. Assume a role that includes the IP Filter Management rights profile (see `rbac(5)`) or become superuser.
2. Create a packet filtering rule set. See `ipf(4)`.
3. (Optional) Create a network address translation (NAT) configuration file. See `ipnat.conf(4)`.
4. (Optional) Create an address pool configuration file. See `ippool(4)`.

   Create an `ipool.conf` file if you want to refer to a group of addresses as a single address pool. If you want the address pool configuration file to be loaded at boot time, create a file called `/etc/ipf/ippool.conf` in which to put the address pool. If you do not want the address pool configuration file to be loaded at boot time, put the `ippool.conf` file in a location other than `/etc/ipf` and manually activate the rules.
5. Enable Solaris IP Filter, as follows:

   ```
   # svcadm enable network/ipfilter
   ```

To re-enable packet filtering after it has been temporarily disabled either reboot the machine or perform the following series of commands:

1. Enable Solaris IP Filter:

   ```
   # ipf -E
   ```
2. Activate packet filtering:

   # ipf -f *<ipf configuration file>*
3. (Optional) Activate NAT:

ipnat -f *<IPNAT configuration file>*

See ipnat(1M).

Note – If you reboot your system, the packet filtering rules in the `/etc/ipf/ipf.conf` file and the `/etc/ipf/ipnat.conf` file are activated.

Options The following options are supported:

`-6`
This option is required to parse IPv6 rules and to have them loaded. Loading of IPv6 rules is subject to change in the future.

`-A`
Set the list to make changes to the active list (default).

`-d`
Turn debug mode on. Causes a hex dump of filter rules to be generated as it processes each one.

`-D`
Disable the filter (if enabled). Not effective for loadable kernel versions.

`-E`
Enable the filter (if disabled). Not effective for loadable kernel versions.

`-F i | o | a`
Specifies which filter list to flush. The parameter should either be `i` (input), `o` (output) or `a` (remove all filter rules). Either a single letter or an entire word starting with the appropriate letter can be used. This option can be before or after any other, with the order on the command line determining that used to execute options.

`-F s | S`
To flush entries from the state table, use the `-F` option in conjuction with either `s` (removes state information about any non-fully established connections) or `S` (deletes the entire state table). You can specify only one of these two options. A fully established connection will show up in `ipfstat -s` output as `4/4`, with deviations either way indicating the connection is not fully established.

`-f` *filename*
Specifies which files `ipf` should use to get input from for modifying the packet filter rule lists.

`-I`
Set the list to make changes to the inactive list.

`-l pass | block | nomatch`
Toggles default logging of packets. Valid arguments to this option are `pass`, `block` and `nomatch`. When an option is set, any packet which exits filtering and matches the set category is logged. This is most useful for causing all packets that do not match any of the loaded rules to be logged.

`-n`
Prevents `ipf` from making any ioctl calls or doing anything which would alter the currently running kernel.

`-o`
Force rules by default to be added/deleted to/from the output list, rather than the (default) input list.

`-P`
Add rules as temporary entries in the authentication rule table.

`-R`
Disable both IP address-to-hostname resolution and port number-to-service name resolution.

`-r`
Remove matching filter rules rather than add them to the internal lists.

`-s`
Swap the currently active filter list to be an alternative list.

`-T` *optionlist*
Allows run-time changing of IPFilter kernel variables. To allow for changing, some variables require IPFilter to be in a disabled state (`-D`), others do not. The *optionlist* parameter is a comma-separated list of tuning commands. A tuning command is one of the following:

`list`
Retrieve a list of all variables in the kernel, their maximum, minimum, and current value.

single variable name
Retrieve its current value.

variable name with a following assignment
To set a new value.

Examples follow:

```
# Print out all IPFilter kernel tunable parameters
ipf -T list

# Display the current TCP idle timeout and then set it to 3600
ipf -D -T fr_tcpidletimeout,fr_tcpidletimeout=3600 -E

# Display current values for fr_pass and fr_chksrc, then set
# fr_chksrc to 1.
ipf -T fr_pass,fr_chksrc,fr_chksrc=1
```

`-v`
Turn verbose mode on. Displays information relating to rule processing.

`-V`
Show version information. This will display the version information compiled into the `ipf` binary and retrieve it from the kernel code (if running or present). If it is present in the kernel, information about its current state will be displayed; for example, whether logging is active, default filtering, and so forth).

`-y`
Manually resync the in-kernel interface list maintained by IP Filter with the current interface status list.

`-z`
For each rule in the input file, reset the statistics for it to zero and display the statistics prior to them being zeroed.

`-Z`
Zero global statistics held in the kernel for filtering only. This does not affect fragment or state statistics.

Files

`/dev/ipauth`
`/dev/ipl`
`/dev/ipstate`
Links to IP Filter pseudo devices.

`/etc/ipf/ipf.conf`
Location of `ipf` startup configuration file. See `ipf(4)`.

`/usr/share/ipfilter/examples/`
Contains numerous IP Filter examples.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWipfu
Interface Stability	Evolving

See Also `ipfstat(1M)`, `ipmon(1M)`, `ipnat(1M)`, `svcadm(1M)`, `ipf(4)`, `ipnat.conf(4)`, `ippool(4)`, `attributes(5)`, `ipfilter(5)`

System Administration Guide: IP Services

Diagnostics Needs to be run as root for the packet filtering lists to actually be affected inside the kernel.

Notes To view license terms, attribution, and copyright for IP Filter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.

Name ipfs – saves and restores information for NAT and state tables

Synopsis

```
ipfs [-nv] -l

ipfs [-nv] -u

ipfs [-nv] [-d dirname] -R

ipfs [-nv] [-d dirname] -W

ipfs [-nNSv] [-f filename] -r

ipfs [-nNSv] [-f filename] -w

ipfs [-nNSv] -f filename -i <if1>,<if2>
```

Description The `ipfs` utility enables the saving of state information across reboots. Specifically, the utility allows state information created for NAT entries and rules using "keep state" to be locked (modification prevented) and then saved to disk. Then, after a reboot, that information is restored. The result of this state-saving is that connections are not interrupted.

Options The following options are supported:

`-d` Change the default directory used with `-R` and `-W` options for saving state information.

`-n` Do not take any action that would affect information stored in the kernel or on disk.

`-v` Provides a verbose description of `ipfs` activities.

`-N` Operate on NAT information.

`-S` Operate on filtering state information.

`-u` Unlock state tables in the kernel.

`-l` Lock state tables in the kernel.

`-r` Read information in from the specified file and load it into the kernel. This requires the state tables to have already been locked and does not change the lock once complete.

`-w` Write information out to the specified file and from the kernel. This requires the state tables to have already been locked and does not change the lock once complete.

`-R` Restores all saved state information, if any, from two files, `ipstate.ipf` and `ipnat.ipf`, stored in the `/var/db/ipf` directory. This directory can be changed with the `-d` option. The state tables are locked at the beginning of this operation and unlocked once complete.

`-W` Saves in-kernel state information, if any, out to two files, `ipstate.ipf` and `ipnat.ipf`, stored in the `/var/db/ipf` directory. This directory can be changed with the `-d` option. The state tables are locked at the beginning of this operation and unlocked once complete.

Files

- `/var/db/ipf/ipstate.ipf`
- `/var/db/ipf/ipnat.ipf`
- `/dev/ipl`
- `/dev/ipstate`
- `/dev/ipnat`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWipfu
Interface Stability	Evolving

See Also `ipf(1M)`, `ipmon(1M)`, `ipnat(1M)`, `attributes(5)`

Notes To view license terms, attribution, and copyright for IP Filter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.

Diagnostics Arguably, the `-W` and `-R` operations should set the locking and, rather than undo it, restore it to what it was previously.

Fragment table information is currently not saved.

Name ipfstat – reports on packet filter statistics and filter list

Synopsis

```
ipfstat [-6aACdfghIilnoRstv]

ipfstat [-C] [-D addrport] [-P protocol] [-S addrport]
     [-T refreshtime]
```

Description The `ipfstat` command is part of a suite of commands associated with the Solaris IP Filter feature. See `ipfilter(5)`.

The `ipfstat` command examines `/dev/kmem` using the symbols `_fr_flags`, `_frstats`, `_filterin`, and `_filterout`. To run and work, it needs to be able to read both `/dev/kmem` and the kernel itself.

The default behavior of `ipfstat` is to retrieve and display the statistics which have been accumulated over time as the kernel has put packets through the filter.

The role of `ipfstat` is to display current kernel statistics gathered as a result of applying the filters in place (if any) to packets going in and out of the kernel. This is the default operation when no command line parameters are present. When supplied with either `-i` or `-o`, `ipfstat` will retrieve and display the appropriate list of filter rules currently installed and in use by the kernel.

`ipfstat` uses kernel device files to obtain information. The default permissions of these files require `ipfstat` to be run as root for all operations.

The `ipfstat` command supports the `kstat(3KSTAT)` kernel facility. Because of this support, as an alternative to `ipfstat`, you can use `kstat(1M)`. For example:

```
# kstat -m ipf
```

Using the `ipfstat -t` option causes `ipfstat` to enter the state top mode. In this mode the state table is displayed similarly to the way the Unix `top` utility displays the process table. The `-C`, `-D`, `-P`, `-S` and `-T` command line options can be used to restrict the state entries that will be shown and to specify the frequency of display updates.

In state top mode, use the following keys to influence the displayed information:

`d` Select information to display.

`l` Redraw the screen.

`q` Quit the program.

`s` Switch between different sorting criteria.

`r` Reverse the sorting criteria.

States can be sorted by protocol number, by number of IP packets, by number of bytes, and by time-to-live of the state entry. The default is to sort by the number of bytes. States are sorted in descending order, but you can use the `r` key to sort them in ascending order.

It is not possible to interactively change the source, destination, and protocol filters or the refresh frequency. This must be done from the command line.

The screen must have at least 80 columns for correct display. However, `ipfstat` does not check the screen width.

Only the first *X*-5 entries that match the sort and filter criteria are displayed (where *X* is the number of rows on the display). There is no way to see additional entries.

Options The following options are supported:

`-6` Display filter lists and states for IPv6, if available. This option might change in the future.

`-a` Display the accounting filter list and show bytes counted against each rule.

`-A` Display packet authentication statistics.

`-C` Valid only in combination with `-t`. Display "closed" states as well in the top. Normally, a TCP connection is not displayed when it reaches the `CLOSE_WAIT` protocol state. With this option enabled, all state entries are displayed.

`-d` Produce debugging output when displaying data.

`-D` *addrport* Valid only in combination with `-t`. Limit the state top display to show only state entries whose destination IP address and port match the *addrport* argument. The *addrport* specification is of the form *ipaddress*[,*port*]. The *ipaddress* and *port* should be either numerical or the string `any` (specifying any IP address and any port, in that order). If the `-D` option is not specified, it defaults to `-D any,any`.

`-f` Show fragment state information (statistics) and held state information (in the kernel) if any is present.

`-g` Show groups currently configured (both active and inactive).

`-h` Show per-rule the number of times each one scores a "hit". For use in combination with `-i`.

`-i` Display the filter list used for the input side of the kernel IP processing.

`-I` Swap between retrieving `inactive/active` filter list details. For use in combination with `-i`.

`-l` When used with `-s`, show a list of active state entries (no statistics).

`-n` Show the rule number for each rule as it is printed.

`-o` Display the filter list used for the output side of the kernel IP processing.

`-P` *protocol* — Valid only in combination with `-t`. Limit the state top display to show only state entries that match a specific protocol. The argument can be a protocol name (as defined in `/etc/protocols`) or a protocol number. If this option is not specified, state entries for any protocol are specified.

`-R` — Disable both IP address-to-hostname resolution and port number-to-service name resolution.

`-S` *addrport* — Valid only in combination with `-t`. Limit the state top display to show only state entries whose source IP address and port match the *addrport* argument. The *addrport* specification is of the form *ipaddress*[,*port*]. The *ipaddress* and *port* should be either numerical or the string `any` (specifying any IP address and any port, in that order). If the `-S` option is not specified, it defaults to `-S any,any`.

`-s` — Show packet/flow state information (statistics only).

`-T` *refreshtime* — Valid only in combination with `-t`. Specifies how often the state `top` display should be updated. The refresh time is the number of seconds between an update. Any positive integer can be used. The default (and minimal update time) is 1.

`-t` — Show the state table in a way similar to the way the Unix utility, `top`, shows the process table. States can be sorted in a number of different ways.

`-v` — Turn verbose mode on. Displays additional debugging information.

Files

- `/dev/kmem`
- `/dev/ksyms`
- `/dev/ipl`
- `/dev/ipstate`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWipfu
Interface Stability	Evolving

See Also

`ipf(1M)`, `kstat(1M)`, `kstat(3KSTAT)`, `attributes(5)`, `ipfilter(5)`

System Administration Guide: IP Services

Notes

To view license terms, attribution, and copyright for IP Filter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.

Name ipmon – monitors /dev/ipl for logged packets

Synopsis

```
ipmon [-abDFhnpstvxX] [-N device] [ [o] [NSI]] [-O [NSI]]
      [-P pidfile] [-S device] [-f device] [filename]
```

Description The `ipmon` command is part of a suite of commands associated with the Solaris IP Filter feature. See `ipfilter(5)`.

The `ipmon` command opens `/dev/ipl` for reading and awaits data to be saved from the packet filter. The binary data read from the device is reprinted in human readable form. However, IP addresses are not mapped back to hostnames, nor are ports mapped back to service names. The output goes to standard output, by default, or a filename, if specified on the command line. Should the `-s` option be used, output is sent instead to `syslogd(1M)`. Messages sent by means of `syslog` have the day, month, and year removed from the message, but the time (including microseconds), as recorded in the log, is still included.

Messages generated by `ipmon` consist of whitespace-separated fields. Fields common to all messages are:

- The date of packet receipt. This is suppressed when the message is sent to `syslog`.
- The time of packet receipt. This is in the form *HH*:*MM*:*SS*.*F*, for hours, minutes, seconds, and fractions of a second (which can be several digits long).
- The name of the interface on which the packet was processed, for example, `ib1`.
- The group and rule number of the rule, for example, `@0:17`. These can be viewed with `ipfstat -in` for input rules or `ipfstat -in` for output rules. See `ipfstat(1M)`.
- The action: `p` for passed, `b` for blocked, `s` for a short packet, `n` did not match any rules, or `L` for a log rule.
- The addresses. This is actually three fields: the source address and port (separated by a comma), the symbol →, and the destination address and port. For example: `209.53.17.22,80 → 198.73.220.17,1722`.
- `PR` followed by the protocol name or number, for example, `PR tcp`.
- `len` followed by the header length and total length of the packet, for example, `len 20 40`.

If the packet is a TCP packet, there will be an additional field starting with a hyphen followed by letters corresponding to any flags that were set. See `ipf.conf(4)` for a list of letters and their flags.

If the packet is an ICMP packet, there will be two fields at the end, the first always being `icmp`, the next being the ICMP message and submessage type, separated by a slash. For example, `icmp 3/3` for a port unreachable message.

Options The following options are supported:

`-a`

Open all of the device logfiles for reading log entries. All entries are displayed to the same output device (stderr or syslog).

`-b`
For rules which log the body of a packet, generate hex output representing the packet contents after the headers.

`-D`
Cause `ipmon` to turn itself into a daemon. Using subshells or backgrounding of `ipmon` is not required to turn it into an orphan so it can run indefinitely.

`-f` *device*
Specify an alternative device/file from which to read the log information for normal IP Filter log records.

`-F`
Flush the current packet log buffer. The number of bytes flushed is displayed, even if the result is zero.

`-h`
Displays usage information.

`-n`
IP addresses and port numbers will be mapped, where possible, back into hostnames and service names.

`-N` *device*
Set the logfile to be opened for reading NAT log records from or to *device*.

`-o` *letter*
Specify which log files from which to actually read data. `N`, NAT logfile; `S`, state logfile; `I`, normal IP Filter logfile. The `-a` option is equivalent to using `-o NSI`.

`-O` *letter*
Specify which log files you do not wish to read from. This is most commonly used in conjunction with the `-a`. Letters available as parameters are the same as for `-o`.

`-p`
Cause the port number in log messages always to be printed as a number and never attempt to look it up.

`-P` *pidfile*
Write the PD of the `ipmon` process to a file. By default this is `/var/run/ipmon.pid`.

`-s`
Packet information read in will be sent through `syslogd` rather than saved to a file. The default facility when compiled and installed is `local0`. The following levels are used:

`LOG_INFO`
Packets logged using the `log` keyword as the action rather than `pass` or `block`.

`LOG_NOTICE`
Packets logged that are also passed.

`LOG_WARNING`
Packets logged that are also blocked.

`LOG_ERR`
Packets that have been logged and that can be considered “short”.

`-S` *device*
Set the logfile to be opened for reading state log records from or to *device*.

`-t`
Read the input file/device in the way performed by `tail(1)`.

`-v`
Show TCP `window`, `ack`, and `sequence` fields

`-x`
Show the packet data in hex.

`-X`
Show the log header record data in hex.

Files
- `/dev/ipl`
- `/dev/ipnat`
- `/dev/ipstate`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWipfu
Interface Stability	Evolving

See Also `ipf(1M)`, `ipfstat(1M)`, `ipnat(1M)`, `attributes(5)`, `ipfilter(5)`

System Administration Guide: IP Services

Diagnostics `ipmon` expects data that it reads to be consistent with how it should be saved and aborts if it fails an assertion which detects an anomaly in the recorded data.

Notes To view license terms, attribution, and copyright for IP Filter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.

Name ipnat – user interface to the NAT subsystem

Synopsis

```
ipnat [-CdFhlnRrsv] -f filename
```

Description The `ipnat` utility opens a specified file (treating - as stdin) and parses it for a set of rules that are to be added or removed from the IP NAT.

If there are no parsing problems, each rule processed by `ipnat` is added to the kernel's internal lists. Rules are appended to the internal lists, matching the order in which they appear when given to `ipnat`.

`ipnat`'s use is restricted through access to `/dev/ipauth`, `/dev/ipl`, and `/dev/ipstate`. The default permissions of these files require `ipnat` to be run as root for all operations.

`ipnat`'s use is restricted through access to `/dev/ipnat`. The default permissions of `/dev/ipnat` require `ipnat` to be run as root for all operations.

Options The following options are supported:

`-C`	Delete all entries in the current NAT rule listing (NAT rules).
`-d`	Turn debug mode on. Causes a hex dump of filter rules to be generated as it processes each one.
`-F`	Delete all active entries in the current NAT translation table (currently active NAT mappings).
`-f` *filename*	Parse specified file for rules to be added or removed from the IP NAT. *filename* can be stdin.
`-h`	Print number of hits for each MAP/Redirect filter.
`-l`	Show the list of current NAT table entry mappings.
`-n`	Prevents `ipf` from doing anything, such as making ioctl calls, which might alter the currently running kernel.
`-R`	Disable both IP address-to-hostname resolution and port number-to-service name resolution.
`-r`	Remove matching NAT rules rather than add them to the internal lists.
`-s`	Retrieve and display NAT statistics.
`-v`	Turn verbose mode on. Displays information relating to rule processing and active rules/table entries.

Files

`/dev/ipnat`	Link to IP Filter pseudo device.
`/dev/kmem`	Special file that provides access to virtual address space.
`/etc/ipf/ipnat.conf`	Location of `ipnat` startup configuration file.

`/usr/share/ipfilter/examples/` Contains numerous IP Filter examples.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWipfu
Interface Stability	Evolving

See Also `ipf(1M)`, `ipfstat(1M)`, `ipnat(4)`, `attributes(5)`

Notes To view license terms, attribution, and copyright for IP Filter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.

Name ippool – user interface to the IP Filter pools

Synopsis

```
ippool -a [-dnv] [-m num] [-o role] -i ipaddr [/netmask]
ippool -A [-dnv] [-m num] [-o role] [-S seed] [-t type]
ippool -f file [-dnuv]
ippool -F [-dv] [-o role] [-t type]
ippool -l [-dv] [-m num] [-t type]
ippool -r [-dnv] [-m num] [-o role] -i ipaddr [/netmask]
ippool -R [-dnv] [-m num] [-o role] [-t type]
ippool -s [-dtv] [-M core] [-N namelist]
```

Description The `ippool` utility is used to manage information stored in the IP pools subsystem of IP Filter software. Configuration file information can be parsed and loaded into the kernel and currently configured pools can be removed, changed, or inspected.

`ippool`'s use is restricted through access to `/dev/ippool`. The default permissions of `/dev/ippool` require `ippool` to be run as root for all operations.

The command line options used are divided into two sections: the global options and the instance-specific options.

`ippool`'s use is restricted through access to `/dev/ipauth`, `/dev/ipl`, and `/dev/ipstate`. The default permissions of these files require `ippool` to be run as root for all operations.

Options `ippool` supports the option categories described below.

Global Options The following global options are supported:

`-d`
Toggle debugging of processing the configuration file.

`-n`
Prevents `ippool` from doing anything, such as making ioctl calls, that would alter the currently running kernel.

`-v`
Turn verbose mode on.

Instance-Specific Options The following instance-specific options are supported:

`-a`
Add a new data node to an existing pool in the kernel.

`-A`
Add a new (empty) pool to the kernel.

`-f` *file*
Read in IP pool configuration information from *file* and load it into the kernel.

`-F`
Flush loaded pools from the kernel.

`-l`
Display a list of pools currently loaded into the kernel.

`-r`
Remove an existing data node from a pool in the kernel.

`-R`
Remove an existing pool from within the kernel.

`-s`
Display IP pool statistical information.

Other Options The following, additional options are supported:

`-i` *ipaddr*[/*netmask*]
Sets the IP address for the operation being undertaken with an all-one's mask or, optionally, a specific netmask, given in either dotted-quad notation or as a single integer.

`-m` *poolname*
Sets the pool name for the current operation.

`-M` *core*
Specify an alternative path to `/dev/kmem` from which to retrieve statistical information.

`-N` *namelist*
Specify an alternative path to lookup symbol name information when retrieving statistical information.

`-o` *role*
Sets the role with which this pool is to be used. Currently only `ipf`, `auth`, and `count` are accepted as arguments to this option.

`-S` *seed*
Sets the hashing seed to the number specified. For use with hash-type pools only.

`-t` *type*
Sets the type of pool being defined. Must be one of `pool`, `hash`, or `group-map`.

`-u`
When parsing a configuration file, rather than load new pool data into the kernel, unload it.

Files `/dev/ippool`
Link to IP Filter pseudo device.

`/dev/kmem`
Special file that provides access to virtual address space.

`/etc/ipf/ippool.conf`
Location of `ippool` startup configuration file.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWipfu
Interface Stability	Evolving

See Also ipf(1M), ipfstat(1M), ippool(4), attributes(5)

Notes To view license terms, attribution, and copyright for IP Filter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.

Name ipqosconf – configure the IPQoS facility

Synopsis

```
/usr/sbin/ipqosconf

/usr/sbin/ipqosconf -a conf_file [-vs]

/usr/sbin/ipqosconf -c

/usr/sbin/ipqosconf -f

/usr/sbin/ipqosconf -l

/usr/sbin/ipqosconf -L
```

Description The `ipqosconf` utility configures the Quality of Service facility of the Internet Protocol (IP). Only superusers can use this command.

Without arguments, `ipqosconf` displays the actual `IPQoS` configuration.

Configuration is not preserved across reboot. You must apply the configuration every time that the machine reboots. To apply the configuration early in the boot phase, you can populate the `/etc/inet/ipqosinit.conf` file, which is then read from the `svc:/network/initial:default` service.

Options The following options are supported:

`-a` *conf_file* Apply the configuration in *conf_file*. If the *conf_file* is –, `ipqosconf` reads from standard input.

`-c` Populate the boot file with the current configuration.

`-f` Flush the configuration.

`-l` List the current applied configuration.

`-L` List the current configuration in verbose mode.

In addition to the information that the `-l` option provides, the `-L` option provides filters and classes configured through other means than the `iqposconf` command. This option also provides the full set of filters that were created by `ipqosconf` by representing a multi-homed host in a configuration file

`-s` Log messages to `syslog` during an `-a` operation.

`-v` Toggle verbose mode during an `-a` operation.

The `-v` option causes all messages to go to the console in addition to their normal destination. Messages intended to go to `syslog`, because the `-s` flag is set or because it is a log message, still go to `syslog` as well as the console.

Configuration File The configuration file is composed of a format version and a succession of configuration (action) blocks. There are different configuration blocks for each type of action that is being configured.

Format Version The first line of the configuration file specifies the format version contained in the configuration file.

The following entry specifies the format version:

```
fmt_version x.x
```

where *x.x* is the format version. `1.0` is the only supported version.

Configuration Blocks Following the format version, are a succession of configuration (action) blocks that are different for each type of action being configured. A configuration block always has the following structure :

```
action {
      name action_name
      module module_name
      params_clause | ""
      cf_clauses
}

action_name      ::= string
module_name      ::= ipgpc | dlcosmk | dscpmk | flowacct | tswtclmt |
                     tokenmt

params_clause    ::= params {
                        parameters
                        params_stats | ""
                     }

parameters       ::= prm_name_value parameters | ""

prm_name_value   ::= param_name  param_value
```

Modules The *param_name* and the types of *param_value* are specific to a given module.

```
params_stats     ::= global_stats boolean

cf_clauses       ::= class_clause cf_clauses |
                     filter_clause cf_clauses | ""

class_clause     ::= class {
                         name class_name
                         next_action next_action_name
                         class_stats | ""
                     }
```

```
class_name       ::= string
next_action_name ::= string
class_stats      ::= enable_stats boolean
boolean          ::= TRUE | FALSE

filter_clause    ::= filter {
                        name filter_name
                        class class_name
                        parameters
                     }

filter_name      ::= string
```

There must be exactly one configuration block belonging to module `ipgpc`. The action must be named `ipgpc.classify`. All other actions should be reachable from `ipgpc` by way of parameters of type action or the next_action of a class.

The set of types that are used for parameters of the different modules are:

```
action        ::=     string
protocol      ::=     1..255
port          ::=     1..65535
uint8         ::=     0..255
uint32        ::=     0..4294967296
int32         ::=     -2147483648..2147483648
address       ::=    <see the description section>
ifname        ::=    <interface name recognized by SIOGLIFINDEX ioctl>
enum          ::=     string | { string_list }
boolean       ::=     TRUE | FALSE
integer_array ::=     { range_value_list }
map_index     ::=     uint32
address       ::=     ip_address | ip_node_name
user          ::=     uid | username
uid           ::=     0..65535
username      ::=     string
string_list          ::=   string sl_entrys
sl_entrys            ::=   ',' string sl_entrys | ""
range_value_list     ::=   range_value_entry range_value_entrys
range_value_entry    ::=   range ':' integer_array_value
range                ::=   uint32 '-' uint32
integer_array_value  ::=   string | integer_array_number
integer_array_number ::=   uint8 | uint32
range_value_entrys   ::=   ';' range_value_entry range_value_entrys | ""
ip_node_name         ::=   string
ip_address           ::=   v4_address | v6_address
v4_address           ::=   v4_ip_address / v4_cidr_mask |
v4_ip_address
```

```
v4_cidr_mask         ::=   1-32
v6_address           ::=   v6_ip_address / v6_cidr_mask |
v6_ip_address
v6_cidr_mask         ::=   1-128
```

METER module tokenmt configuration syntax :

```
red_action_name         action
yellow_action_name      action
green_action_name       action
committed_rate          uint32
committed_burst         uint32
peak_rate               uint32
<if present this signifies that this will be a two rate meter, not
   a single rate meter>
peak_burst              uint32
<this is the 'peak' burst size for a two rate meter, but
   the 'excess' burst size for a single rate meter>
color_aware             boolean
color_map               integer_array
global_stats            boolean
```

METER module tswtclmt configuration syntax :

```
red_action_name         action
yellow_action_name      action
green_action_name       action
committed_rate          uint32
peak_rate               uint32
window                  uint32
global_stats            boolean
```

MARKER module dscpmk configuration syntax :

```
next_action         action
dscp_map            int_array
dscp_detailed_stats boolean
global_stats       boolean
```

MARKER module dlcosmk configuration syntax :

```
next_action         action
cos                 map_index
global_stats        boolean
```

CLASSIFIER module ipgpc configuration syntax :

```
if_grpname         string
user               user
projid             int32
```

```
if_name             ifname
direction           enum {
                    LOCAL_IN,
                    LOCAL_OUT,
                    FWD_IN,
                    FWD_OUT}
protocol            protocol
dsfield             uint8
dsfield_mask        uint8
saddr               address
daddr               address
sport               port
dport               port
priority            uint32
precedence          uint32
ip_version          enum {
                    V4,
                    V6 }
global_stats        boolean
```

ACCOUNTING module flowacct configuration syntax

```
next_action       action
timer             uint32
timeout           uint32
max_limit         uint32
```

Types

action A string of characters with a matching action definition. The character string can be up to twenty three characters in length. To allow for spaces the string needs to be enclosed in quotes and cannot span lines. Two special actions are pre-defined and can not have an explicit action definition. The two pre-defined actions are `continue` and `drop`. continue causes the packet that is passed to it to continue normal processing. `drop` causes the packet that is passed to it to be dropped.

address A machine name or address recognized by `getipnodebyname`(3SOCKET). If a machine name is specified, and `ip_version` has been defined, the query is done using that address family. If a machine name is not specified and `ip_version` has not been defined, the query is done using the `AI_DEFAULT` flag to `getipnodebyname()(..AF_INET6..)`. `CIDR` address masks following an IP address are allowed. Specify the `CIDR` address masks as `1-32` (for `v4`) or `1-128` (for `v6`). `CIDR` addresses are disallowed for node names.

enum Either one of the supported values or comma delimited list of support values, enclosed in curly braces.

ifname A non-`NULL`, existing interface name recognized by the `SIOGLIFINDEX` socket ioctl.

integer_array	A comma delimited set of *range*/*value* pairs , enclosed in curly braces. Specify *range* in the format *x*-*y*, where *x* and *y* are integers that denote the range of array indexes to which the value applies. The minimum value for both *x* and *y* is `0`. The maximum value for *x* is particular to the parameter. Any array indexes not referred to in the set of ranges are left at their previous value.
map_index	A non-negative integer used as an index into any maps associated with a parameter of this type. The maximum value of this type is dictated by the number of entries in the associated maps. The index starts at `0`.
port	Either a service name recognized by `getservbyname(3SOCKET)` or an integer `1-65535`.
protocol	Either a protocol name recognized by `getprotobyname(3SOCKET)` or an integer `1-255`.
string	A character string. Enclose *string* in quotes. *string* cannot span multiple lines.
user	Either a valid user ID or username for the system that is being configured.

Parameters

The configuration file can contain the following parameters

color_aware	A value of `TRUE` or `FALSE`, indicating whether or not the configured action takes account of the previous packet coloring when classifying.
color_map	An integer array that defines which values of the `dscp` field correspond with which colors for when the `color_aware` parameter is set to `TRUE`.
committed_burst	The committed burst size in bits.
committed_rate	The committed rate in bits per second.
cos	The value used to determine the underlying driver level priority applied to the packet which is defined in `802.1D`.
daddr	The destination address of the datagram.
direction	The value used to build a filter matching only part of the traffic. This parameter is of type `enum` with valid values of `LOCAL_IN` (local bound traffic), `LOCAL_OUT` (local sourced traffic), `FWD_IN` (forwarded traffic entering the system), and `FWD_OUT` (forwarded traffic exiting the system).
dport	The destination port of the datagram.

dscp_detailed_stats	A value of `TRUE` or `FALSE` that determines whether detailed statistics are switched on for this `dscp` action. Specify `TRUE` to switch on or `FALSE` to switch off.
dscp_map	The *integer_array* that supplies the values that IP packets with a given `dscp` value have their dscp re-marked with. The existing value is used to index into the array where the new value is taken from. The array is of size `64`, meaning valid indexes are `0-63` and valid values are also `0-63`.
dsfield	The `DS` field of the `IP` datagram header. This is an 8–bit value, with each bit position corresponding with the same one in the header; this enables matches to be done on the CU bits. If you specify this parameter, you must also specify the `dsfield_mask` parameter.
`dsfield_mask`	The mask applied to the `dsfield` parameter to determine the bits against which to match. This is an 8-bit value, with each bit position corresponding with the same one in the `dsfield` parameter.
global_stats	A value of `TRUE` or `FALSE` to enable or disable the statistic collection for this action.
green_action_name	The action to be executed for packets that are deemed to be green.
if_grpname	The interface group name.
if_name	The name of an interface recognized by the `SIOGLIFINDEX` ioctl. This parameter is of type `ifname`.
ip_version	This parameter is of type `enum` and has valid values of `V4` and `V6`. If it is set to `V4` only then only `ipv4`addresses are requested for a specified hostname. If it is set to `V6`, only `ipv6` addresses are returned if there are any, otherwise `v4` mapped `v6` addresses are returned. If both `V4` and `V6` are specified, or if `ip_version` is not specified, then both `ipv4` and `ipv6` addresses are requested for a specified hostname.
max_limit	The maximum number of flow entries present at one time in the `flowacct` actions in the memory resident table.
next_action	The action to be executed when the current action is complete. This value can be either the name of an action defined in the configuration file, or one of the two special action types: `drop` and `continue`. See Types for additional information on special action types.

peak_burst	The peak burst size, for a two rate meter, or excess burst size, for a single rate meter, in bits.
peak_rate	The peak rate in bits per second.
precedence	An integer that is used to order filters. If there are two matching filters that have the same priority value, the one with the lower precedence value is the one matched. This parameter should be used because the order of the filters in a configuration file has no influence on their relative precedence.
priority	An integer that represents the relative priority of a filter. If there are two matching filters, the one with the higher priority value is the one matched. Multiple filters can have the same priority.
projid	The project ID of the process sending the data. This value is always `-1` for received traffic.
protocol	The Upper Layer Protocol against which this entry is matched.
red_action_name	The action to be executed for packets that are determined to be red.
saddr	The source address of the datagram.
sport	The source port of the datagram.
timeout	The timeout in milliseconds after which flows are written to the accounting file.
timer	The period in milliseconds at which timed-out flows are checked for.
user	The user ID or username of the process sending the data. This value is always `-1` for received traffic.
window	The window size in ms.
yellow_action_name	The action to be executed for packets that are determined to be yellow.

Security None.

Examples **EXAMPLE 1** Sending All Traffic From eng to the AF 1 Class of Service

This example sends all traffic from `eng` to the `AF 1` class of service. It is documented in four separate steps:

The following step creates a `tokenmt` action with three outcomes:

```
#meter for class 1.
action {
        name AF_CL1
        module tokenmt
```

EXAMPLE 1 Sending All Traffic From eng to the AF 1 Class of Service *(Continued)*

```
        params{
                committed_rate 64
                committed_burst 75
                peak_burst 150
                global_stats TRUE
                red_action_name drop
                yellow_action_name markAF12
                green_action_name markAF11
        }
}
```

The following step creates two `dscpmk` actions:

```
#class 1, low drop precedence.
action {
        name markAF11
        module dscpmk
        params{
             dscp_map {0-63:28}
             dscp_detailed_stats TRUE
                global_stats TRUE
                next_action acct1
        }
}
#class 1, medium drop precedence.
action {
        name markAF12
        module dscpmk
        params {
                dscp_map {0-63:30}
             dscp_detailed_stats TRUE
                global_stats TRUE
                next_action acct1
        }
}
```

The following step creates an accounting action:

```
#billing for transmitted class 1 traffic.
action {
        name acct1
        module flowacct
        params {
                timer 10
                timeout 30
                global_stats TRUE
```

EXAMPLE 1 Sending All Traffic From eng to the AF 1 Class of Service *(Continued)*

```
max_limit 1024
next_action continue
        }
}
```

The following step creates an `ipgpc` action:

```
#traffic from eng sent, traffic from ebay dropped.
action {
        name ipgpc.classify
        module ipgpc
        class {
                name from_eng
                enable_stats TRUE
                next_action AF_CL1
        }
        class {
                name from_ebay
                enable_stats TRUE
                next_action drop
        }

        filter {
                name from_eng
                saddr eng-subnet
                class from_eng
        }
        filter {
                name from_ebay
                saddr ebay-subnet
                class from_ebay
        }
}
```

Files

`/etc/inet/ipqosinit.conf`	Contains the `IPQoS` configuration loaded at boot time. If this file exists, it is read from the `network/initial:default` service.
`/etc/inet/ipqosconf.1.sample`	Sample configuration file for an application server
`/etc/inet/ipqosconf.2.sample`	Sample configuration file that meters the traffic for a specified application
`/etc/inet/ipqosconf.3.sample`	Sample configuration file that marks the ethernet headers of web traffic with a given user priority

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWqosu
Interface Stability	Evolving

See Also syslog(3C), getipnodebyname(3SOCKET), getprotobyname(3SOCKET), getservbyname(3SOCKET), attributes(5), dlcosmk(7ipp), dscpmk(7ipp), flowacct(7ipp), ipgpc(7ipp), ipqos(7ipp), tokenmt(7ipp), tswtclmt(7ipp)

Diagnostics ipqosconf sends messages to syslog of facility user, severity notice when any changes are made to the IPQoS configuration.

Errors that occur during an ipqosconf operation send an error message to the console by default. For the application of a new configuration if the -s option is set then these messages are sent to syslog as facility user, severity error instead. If the -v option is present during an application then all error and change notificationmessages are sent to the console as well as their default destination.

Name ipsecalgs – configure the IPsec protocols and algorithms table

Synopsis

```
ipsecalgs

ipsecalgs -l

ipsecalgs -s

ipsecalgs -a [-P protocol-number | -p protocol-name] -k keylen-list
     [-i inc] [-K default-keylen] -b blocklen-list -n alg-names
     -N alg-number -m mech-name [-f] [-s]

ipsecalgs -P protocol-number -p protocol-name
     [-e exec-mode] [-f] [-s]

ipsecalgs -r -p protocol-name [] -n alg-name [-s]

ipsecalgs -r -p protocol-name [] -N alg-number [-s]

ipsecalgs -R -P protocol-number [-s]

ipsecalgs -R -p protocol-name [-s]

ipsecalgs -e exec-mode -P protocol-number [-s]

ipsecalgs -e exec-mode -p protocol-name [-s]
```

Description Use the `ipsecalgs` command to query and modify the IPsec protocol and algorithms stored in `/etc/inet/ipsecalgs`. You can use the `ipsecalgs` command to do the following:

- list the currently defined IPsec protocols and algorithms
- modify IPsec protocols definitions
- modify IPsec algorithms definitions

Never edit the `/etc/inet/ipsecalgs` file manually. The valid IPsec protocols and algorithms are described by the ISAKMP DOI. See *RFC 2407*. In the general sense, a Domain of Interpretation (DOI) defines data formats, network traffic exchange types, and conventions for naming security-relevant information such as security policies or cryptographic algorithms and modes. For `ipsecalgs`, the DOI defines naming and numbering conventions for algorithms and the protocols they belong to. These numbers are defined by the Internet Assigned Numbers Authority (IANA). Each algorithm belongs to a protocol. Algorithm information includes supported key lengths, block or MAC length, and the name of the cryptographic mechanism corresponding to that algorithm. This information is used by the IPsec modules, `ipsecesp(7P)` and `ipsecah(7P)`, to determine the authentication and encryption algorithms that can be applied to IPsec traffic.

The following protocols are predefined:

`IPSEC_PROTO_ESP`	Defines the encryption algorithms (transforms) that can be used by IPsec to provide data confidentiality.
`IPSEC_PROTO_AH`	Defines the authentication algorithms (transforms) that can be used by IPsec to provide authentication.

The mechanism name specified by an algorithm entry must correspond to a valid Solaris Cryptographic Framework mechanism. You can obtain the list of available mechanisms by using the cryptoadm(1M) command.

Applications can retrieve the supported algorithms and their associated protocols by using the functions getipsecalgbyname(3NSL), getipsecalgbynum(3NSL), getipsecprotobyname(3NSL) and getipsecprotobynum(3NSL).

Modifications to the protocols and algorithm by default update only the contents of the /etc/inet/ipsecalgs configuration file. In order for the new definitions to be used for IPsec processing, the changes must be communicated to the kernel using the -s option. See NOTES for a description of how the ipsecalgs configuration is synchronized with the kernel at system restart.

When invoked without arguments, ipsecalgs displays the list of mappings that are currently defined in /etc/inet/ipsecalgs. You can obtain the corresponding kernel table of protocols and algorithms by using the -l option.

Options ipsecalgs supports the following options:

-a Adds an algorithm of the protocol specified by the -P option. The algorithm name(s) are specified with the -n option. The supported key lengths and block sizes are specified with the -k, -i, and -b options.

-b Specifies the block or MAC lengths of an algorithm, in bytes. Set more than one block length by separating the values with commas.

-e Designates the execution mode of cryptographic requests for the specified protocol in the absence of cryptographic hardware provider. See cryptoadm(1M). *exec-mode* can be one of the following values:

sync Cryptographic requests are processed synchronously in the absence of a cryptographic hardware provider. This execution mode leads to better latency when no cryptographic hardware providers are available

async Cryptographic requests are always processed asynchronously in the absence of cryptographic hardware provider. This execution can improve the resource utilization on a multi-CPU system, but can lead to higher latency when no cryptographic hardware providers are available.

This option can be specified when defining a new protocol or to modify the execution mode of an existing protocol. By default, the sync execution mode is used in the absence of a cryptographic hardware provider.

-f Used with the -a option to force the addition of an algorithm or protocol if an entry with the same name or number already exists.

`-i` Specifies the valid key length increments in bits. This option must be used when the valid key lengths for an algorithm are specified by a range with the `-k` option.

`-K` Specifies the default key lengths for an algorithm, in bits. If the `-K` option is not specified, the minimum key length will be determined as follows:

- If the supported key lengths are specified by range, the default key length will be the minimum key length.
- If the supported key lengths are specified by enumeration, the default key length will be the first listed key length.

`-k` Specifies the supported key lengths for an algorithm, in bits. You can designate the supported key lengths by enumeration or by range.

Without the `-i` option, `-k` specifies the supported key lengths by enumeration. In this case, *keylen-list* consists of a list of one or more key lengths separated by commas, for example:

```
128,192,256
```

The listed key lengths need not be increasing, and the first listed key length will be used as the default key length for that algorithm unless the `-K` option is used.

With the `-i` option, `-k` specifies the range of supported key lengths for the algorithm. The minimum and maximum key lengths must be separated by a dash ('`-`') character, for example:

```
32-448
```

`-l` Displays the kernel algorithm tables.

`-m` Specifies the name of the cryptographic framework mechanism name corresponding to the algorithm. Cryptographic framework mechanisms are described in the `cryptoadm(1M)` man page.

`-N` Specifies an algorithm number. The algorithm number for a protocol must be unique. IANA manages the algorithm numbers. See *RFC 2407*.

`-n` Specifies one or more names for an algorithm. When adding an algorithm with the `-a` option, *alg-names* contains a string or a comma-separated list of strings, for example:

```
des-cbs,des
```

When used with the `-r` option to remove an algorithm, *alg-names* contains one of the valid algorithm names.

`-P` Adds a protocol of the number specified by *protocol-number* with the name specified by the `-p` option. This option is also used to specify an IPsec protocol when used with the `-a` and the `-R` options. Protocol numbers are managed by the IANA. See *RFC 2407*.

`-p` Specifies the name of the IPsec protocol.

`-R` Removes and IPsec protocol from the algorithm table. The protocol can be specified by number by using the `-P` option or by name by using the `-p` option. The algorithms associated with the protocol are removed as well.

`-r` Removes the mapping for an algorithm The algorithm can be specified by algorithm number using the `-N` option or by algorithm name using the `-A` option.

`-s` Synchronizes the kernel with the contents of `/etc/inet/ipsecalgs`. The contents of `/etc/inet/ipsecalgs` are always updated, but new information is not passed on to the kernel unless the `-s` is used. See `NOTES` for a description of how the `ipsecalgs` configuration is synchronized with the kernel at system restart.

Examples

EXAMPLE 1 Adding a Protocol for IPsec Encryption

The following example shows how to add a protocol for IPsec encryption:

```
example# ipsecalgs -P 3 -p "IPSEC_PROTO_ESP"
```

EXAMPLE 2 Adding the Blowfish Algorithm

The following example shows how to add the Blowfish algorithm:

```
example# ipsecalgs -a -P 3 -k 32-488 -K 128 -i 8 -n "blowfish" \
  -b 8 -N 7 -m CKM_BF_CBC
```

EXAMPLE 3 Updating the Kernel Algorithm Table

The following example updates the kernel algorithm table with the currently defined protocol and algorithm definitions:

```
example# svcadm refresh ipsecalgs
```

Files

`/etc/inet/ipsecalgs`
File that contains the configured IPsec protocols and algorithm definitions. Never edit this file manually.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also

`cryptoadm(1M)`, `ipsecconf(1M)`, `ipseckey(1M)`, `svcadm(1M)`, `getipsecalgbyname(3NSL)`, `getipsecprotobyname(3NSL)`, `ike.config(4)`, `attributes(5)`, `smf(5)`, `ipsecah(7P)`, `ipsecesp(7P)`

Piper, Derrell, *RFC 2407, The Internet IP Security Domain of Interpretation for ISAKMP*. Network Working Group. November 1998.

Notes When protocols or algorithm definitions that are removed or altered, services that rely upon these definitions can become unavailable. For example, if the `IPSEC_PROTO_ESP` protocol is removed, then IPsec cannot encrypt and decrypt packets.

Synchronization of the `ipsecalgs` configuration with the kernel at system startup is provided by the following `smf(5)` service:

```
svc:/network/ipsec/ipsecalgs:default
```

The IPsec services are delivered as follows:

```
svc:/network/ipsec/policy:default (enabled)
svc:/network/ipsec/ipsecalgs:default (enabled)
svc:/network/ipsec/manual-key:default (disabled)
svc:/network/ipsec/ike:default (disabled)
```

Services that are delivered disabled are delivered that way because the system administrator must create configuration files for those services before enabling them. See `ipseckey(1M)` and `ike.config(4)`. The default policy for the `policy` service is to allow all traffic to pass without IPsec protection. See `ipsecconf(1M)`.

The correct administrative procedure is to create the configuration file for each service, then enable each service using `svcadm(1M)`, as shown in the following example:

```
example# svcadm enable ipsecalgs
```

The service's status can be queried using the `svcs(1)` command.

If the `ipsecalgs` configuration is modified, the new configuration should be resynchronized as follows:

```
example# svcadm refresh ipsecalgs
```

Administrative actions on this service, such as enabling, disabling, refreshing, and requesting restart can be performed using `svcadm(1M)`. A user who has been assigned the authorization shown below can perform these actions:

```
solaris.smf.manage.ipsec
```

See `auths(1)`, `user_attr(4)`, `rbac(5)`.

The `ipsecalgs` `smf(5)` service does not have any user-configurable properties.

The `smf(5)` framework records any errors in the service-specific log file. Use any of the following commands to examine the `logfile` property:

```
example# svcs -l ipsecalgs
example# svcprop ipsecalgs
example# svccfg -s ipsecalgs listprop
```

This command requires `sys_ip_config` privilege to operate and thus can run in the global zone and in exclusive-IP zones. All shared-IP zones share the same available set of algorithms; however, you can use `ipsecconf(1M)` to set up system policy that uses differing algorithms for various shared-IP zones. All exclusive-IP zones have their own set of algorithms.

Name ipsecconf – configure system wide IPsec policy

Synopsis

```
/usr/sbin/ipsecconf

/usr/sbin/ipsecconf -a file [-q]

/usr/sbin/ipsecconf -c file

/usr/sbin/ipsecconf  -d [-i tunnel-name] {index, tunnel-name, index}

/usr/sbin/ipsecconf  -f  [-i tunnel-name]

/usr/sbin/ipsecconf  -F

/usr/sbin/ipsecconf  -l  [-i tunnel-name] [-n]

/usr/sbin/ipsecconf  -L  [-n]
```

Description The `ipsecconf` utility configures the IPsec policy for a host or for one of its tunnels. Once the policy is configured, all outbound and inbound datagrams are subject to policy checks as they exit and enter the host or tunnel. For the host policy, if no entry is found, no policy checks will be completed, and all the traffic will pass through. For a tunnel, if no entry is found and there is at least one entry for the tunnel, the traffic will automatically drop. The difference in behavior is because of the assumptions about IPsec tunnels made in many implementations. Datagrams that are being forwarded will not be subjected to policy checks that are added using this command. See `ifconfig(1M)` and `tun(7M)` for information on how to protect forwarded packets. Depending upon the match of the policy entry, a specific action will be taken.

This command can be run only by superuser.

Each entry can protect traffic in either one direction (requiring a pair of entries) or by a single policy entry which installs the needed symmetric `sadb` rules.

When the command is issued without any arguments, the list of file policy entries loaded are shown. To display the (`spd p.e.s`) use the `-l` option. Both will display the index number for the entry. To specify a single tunnel's SPD, use the `-i` option in combination with `-l`. To specify all SPDs, both host and for all tunnels, use `-L`.

Note, since one file policy entry (FPE) can generate multiple SPD pol entries (SPEs), the list of FPEs may not show all the actual entries. However, it is still useful in determining what what rules have been added to get the spd into its current state.

You can use the `-d` option with the index to delete a given policy in the system. If the `-d` option removes an FPE entry that produces multiple SPEs, only then SPD with the same policy index as the FPE will be removed. This can produce a situation where there may be SPEs when there are no FPEs.

As with `-l`, `-d` can use the `-i` flag to indicate a tunnel. An alternate syntax is to specify a tunnel name, followed by a comma (`,`), followed by an index. For example, `ip.tun0,1`.

With no options, the entries are displayed in the order that they were added, which is not necessarily the order in which the traffic match takes place.

To view the order in which the traffic match will take place, use the `-l` option. The rules are ordered such that all bypass rules are checked first, then ESP rules, then AH rules. After that, they are checked in the order entered.

Policy entries are not preserved across system restarts. Permanent policy entries should be added to `/etc/inet/ipsecinit.conf`. This file is read by the following `smf(5)` service:

```
svc:/network/ipsec/policy
```

See `NOTES` for more information on managing IPsec security policy and `SECURITY` for issues in securing `/etc/inet/ipsecinit.conf`.

Options `ipsecconf` supports the following options:

`-a` *file*
Add the IPsec policy to the system as specified by each entry in the file. An IPsec configuration file contains one or more entries that specify the configuration. Once the policy is added, all outbound and inbound datagrams are subject to policy checks.

Entries in the files are described in the Operands section below. Examples can be found in the Examples section below.

Policy is latched for TCP/UDP sockets on which a `connect(3SOCKET)` or `accept(3SOCKET)` is issued. So, the addition of new policy entries may not affect such endpoints or sockets. However, the policy will be latched for a socket with an existing non-null policy. Thus, make sure that there are no preexisting connections that will be subject to checks by the new policy entries.

The feature of policy latching explained above may change in the future. It is not advisable to depend upon this feature.

`-c` *file*
Check the syntax of the configuration file and report any errors without making any changes to the policy. This option is useful when debugging configurations and when `smf(5)` reports a configuration error. See `SECURITY`.

`-d` *index*
Delete the host policy denoted by the index. The index is obtained by invoking `ipsecconf` without any arguments, or with the `-l` option. See DESCRIPTION for more information. Once the entry is deleted, all outbound and inbound datagrams affected by this policy entry will not be subjected to policy checks. Be advised that with connections for which the policy has been latched, packets will continue to go out with the same policy, even if it has been deleted. It is advisable to use the `-l` option to find the correct policy index.

`-d` *name*,*index*
Delete the policy entry denoted by *index* on a tunnel denoted by *name*. Since tunnels affect traffic that might originate off-node, latching does not apply as it does in the host policy case. Equivalent to: `-d` *index* `-i` *name*.

`-f`
Flush all the policies in the system. Constraints are similar to the `-d` option with respect to latching and host versus per-tunnel behavior.

`-F`
Flush all policies on all tunnels and also flush all host policies.

`-i` *name*
Specify a tunnel interface name for use with the `-d`, `-f`, or `-l` flags.

`-l`
Listing of a single policy table, defaulting to the host policy. When `ipsecconf` is invoked without any arguments, a complete list of policy entries with indexes added by the user since boot is displayed. The current table can differ from the previous one if, for example, a multi-homed entry was added or policy reordering occurred, or if a single rule entry generates two `spd` rules In the case of a multi-homed entry, all the addresses are listed explicitly. If a mask was not specified earlier but was instead inferred from the address, it will be explicitly listed here. This option is used to view policy entries in the correct order. The outbound and inbound policy entries are listed separately.

`-L`
Lists all policy tables, including host policy and all tunnel instances (including configured but unplumbed).

If `-i` is specified, `-L` lists the policy table for a specific tunnel interface.

`-n`
Show network addresses, ports, protocols in numbers. The `-n` option may only be used with the `-l` option.

`-q`
Quiet mode. Suppresses the warning message generated when adding policies.

Operands Each policy entry contains three parts specified as follows:

```
{pattern} action {properties}
```

or

```
{pattern} action {properties} ["or" action {properties}]*
```

Every policy entry begins on a new line and can span multiple lines. If an entry exceeds the length of a line, you should split it only within a “braced” section or immediately before the first (left-hand) brace of a braced section. Avoid using the backslash character (\). See EXAMPLES.

The *pattern* section, as shown in the syntax above, specifies the traffic pattern that should be matched against the outbound and inbound datagrams. If there is a match, a specific *action* determined by the second argument will be taken, depending upon the *properties* of the policy entry.

If there is an `or` in the rule (multiple action-properties for a given pattern), a transmitter will use the first action-property pair that works, while a receiver will use any that are acceptable.

pattern and *properties* are name-value pairs where name and value are separated by a <space>, <tab> or <newline>. Multiple name-value pairs should be separated by <space>, <tab> or <newline>. The beginning and end of the pattern and properties are marked by { and } respectively.

Files can contain multiple policy entries. An unspecified name-value pair in the *pattern* will be considered as a wildcard. Wildcard entries match any corresponding entry in the datagram.

One thing to remember is that UDP port 500 is always bypassed regardless of any policy entries. This is a requirement for `in.iked(1M)` to work.

File can be commented by using a # as the first character. Comments may be inserted either at the beginning or the end of a line.

The complete syntax of a policy entry is:

```
policy ::= { <pattern1> } <action1> { <properties1> } |
     { <pattern2> } <action2> { <properties2> }
     [ 'or' <action2> { <properties2>} ]*

     pattern1 ::=  <pattern_name_value_pair1>*

     pattern2 ::=  <pattern_name_value_pair2>*

     action1 ::= apply | permit | bypass | pass
     action2 ::=  bypass | pass | drop | ipsec

     properties1 ::=   {<prop_name_value_pair1>}
     properties2 ::=   {<prop_name_value_pair2>}


     pattern_name_value_pair1 ::=
        saddr <address>/<prefix> |
        src  <address>/<prefix> |
        srcaddr <address>/<prefix> |
        smask <mask> |
        sport <port> |
        daddr <address>/<prefix> |
        dst <address>/<prefix> |
        dstaddr <address>/<prefix> |
        dmask <mask> |
        dport <port> |
        ulp <protocol> |
        proto <protocol> |
        type <icmp-type> |
        type <number>-<number> |
```

```
   code <icmp-code>
   code <number>-<number>
   tunnel <interface-name> |
   negotiate <tunnel,transport>

pattern_name_value_pair2 ::=
   raddr <address>/<prefix> |
   remote <address>/<prefix> |
   rport <port> |
   laddr <address>/<prefix> |
   local <address>/<prefix> |
   lport <port> |
   ulp <protocol> |
   type <icmp-type> |
   type <number>-<number> |
   code <icmp-code> |
   code <number>-<number>
   proto <protocol>  |
   tunnel <interface-name> |
   negotiate <tunnel,transport> |
   dir <dir_val2>

address ::=  <IPv4 dot notation> | <IPv6 colon notation> |
             <String recognized by gethostbyname>|
             <String recognized by getnetbyname>

prefix ::=  <number>

mask ::= <0xhexdigit[hexdigit]> | <0Xhexdigit[hexdigit]> |
         <IPv4 dot notation>

port ::= <number>| <String recognized by getservbyname>

protocol ::=  <number>| <String recognized by getprotobyname>

prop_name_value_pair1 ::=
     auth_algs <auth_alg> |
     encr_algs <encr_alg> |
     encr_auth_algs <auth_alg> |
     sa <sa_val> |
     dir <dir_val1>

prop_name_value_pair2 ::=
     auth_algs <auth_alg> |
     encr_algs <encr_alg> |
     encr_auth_algs <auth_alg> |
     sa <sa_val>
```

```
auth_alg ::=  <auth_algname> [’(’ <keylen> ’)’]
auth_algname ::= any | md5 | hmac-md5 | sha | sha1 | hmac-sha |
                 hmac-sha1 | hmac-sha256 | hmac-sha384 |
                 hmac-sha512 |<number>

encr_alg ::= <encr_algname> [’(’ <keylen> ’)’]
encr_algname ::= any | aes | aes-cbc | des | des-cbc | 3des |
                 3des-cbc | blowfish | blowfish-cbc | <number>

keylen ::= <number> | <number>’..’ | ’..’<number> | <number>’..’ \
<number>

sa_val ::= shared | unique

dir_val1 ::= out | in
dir_val2 ::= out | in | both

number ::= < 0 | 1 | 2 ... 9> <number>
icmp-type ::= <number> | unreach | echo | echorep | squench |
              redir | timex | paramprob | timest | timestrep |
              inforeq | inforep | maskreq | maskrep | unreach6 |
              pkttoobig6 | timex6 | paramprob6 | echo6 | echorep6 |
              router-sol6 | router-ad6 | neigh-sol6 | neigh-ad6 |
              redir6

icmp-code ::= <number> | net-unr | host-unr | proto-unr | port-unr |
              needfrag | srcfail | net-unk | host-unk | isolate |
              net-prohib | host-prohib | net-tos | host-tos |
              filter-prohib | host-preced | cutoff-preced |
              no-route6 | adm-prohib6 | addr-unr6 | port-unr6 |
              hop-limex6 | frag-re-timex6 | err-head6 | unrec-head6 |
              unreq-opt6
```

Policy entries may contain the following (name value) pairs in the *pattern* field. Each (name value) pair may appear only once in given policy entry.

laddr/plen
local/plen
: The value that follows is the local address of the datagram with the prefix length. Only plen leading bits of the source address of the packet will be matched. plen is optional. Local means destination on incoming and source on outgoing packets. The source address value can be a hostname as described in getaddrinfo(3SOCKET) or a network name as described in `getnetbyname(3XNET)` or a host address or network address in the Internet standard dot notation. See `inet_addr(3XNET)`. If a hostname is given and getaddrinfo(3SOCKET) returns multiple addresses for the host, then policy will be added for each of the addresses with other entries remaining the same.

raddr/plen
remote/plen

The value that follows is the remote address of the datagram with the prefix length. Only plen leading bits of the remote address of the packet will be matched. plen is optional. Remote means source on incoming packets and destination on outgoing packets. The remote address value can be a hostname as described in `getaddrinfo`(3SOCKET) or a network name as described in `getnetbyname`(3XNET) or a host address or network address in the Internet standard dot notation. See `inet_addr`(3XNET). If a hostname is given and `getaddrinfo`(3SOCKET) returns multiple addresses for the host, then policy will be added for each of the addresses with other entries remaining the same.

src/plen
srcaddr/plen
saddr/plen
The value that follows is the source address of the datagram with the prefix length. Only *plen* leading bits of the source address of the packet will be matched. *plen* is optional.

The source address value can be a hostname as described in `getaddrinfo`(3SOCKET) or a network name as described in `getnetbyname`(3XNET) or a host address or network address in the Internet standard dot notation. See `inet_addr`(3XNET).

If a hostname is given and `getaddrinfo`(3SOCKET) returns multiple addresses for the host, then policy will be added for each of the addresses with other entries remaining the same.

daddr/plen
dest/plen
dstaddr/plen
The value that follows is the destination address of the datagram with the prefix length. Only *plen* leading bits of the destination address of the packet will be matched. *plen* is optional.

See *saddr* for valid values that can be given. If multiple source and destination addresses are found, then a policy entry that covers each source address-destination address pair will be added to the system.

smask
For IPv4 only. The value that follows is the source mask. If prefix length is given with *saddr*, this should not be given. This can be represented either in hexadecimal number with a leading `0x` or `0X`, for example, `0xffff0000`, `0Xffff0000` or in the Internet decimal dot notation, for example, `255.255.0.0` and `255.255.255.0`. The mask should be contiguous and the behavior is not defined for non-contiguous masks.

smask is considered only when *saddr* is given.

For both IPv4 and IPv6 addresses, the same information can be specified as a *slen* value attached to the *saddr* parameter.

dmask
Analogous to *smask*.

lport
The value that follows is the local port of the datagram. This can be either a port number or a string searched with a NULL proto argument, as described in getservbyname(3XNET)

rport
The value that follows is the remote port of the datagram. This can be either a port number or a string searched with a NULL proto argument, as described in getservbyname(3XNET)

sport
The value that follows is the source port of the datagram. This can be either a port number or a string searched with a `NULL` proto argument, as described in `getservbyname(3XNET)`

dport
The value that follows is the destination port of the datagram. This can be either a port number or a string as described in `getservbyname(3XNET)` searched with `NULL` proto argument.

`proto` *ulp*
The value that follows is the Upper Layer Protocol that this entry should be matched against. It could be a number or a string as described in `getprotobyname(3XNET)`. If no smask or plen is specified, a plen of 32 for IPv4 or 128 for IPv6 will be used, meaning a host. If the *ulp* is `icmp` or `ipv6-icmp`, any action applying IPsec must be the same for all `icmp` rules.

`type` *num* or *num-num*
The value that follows is the ICMP type that this entry should be matched against. `type` must be a number from 0 to 255, or one of the appropriate `icmp-type` keywords. Also, *ulp* must be present and must specify either `icmp` or `ipv6-icmp`. A range of types can be specified with a hyphen separating numbers.

`code` *num* or *num-num*
The value that follows is the ICMP code that this entry should be matched against. The value following the keyword `code` must be a number from 0 to 254 or one of the appropriate `icmp-code` keywords. Also, `type` must be present. A range of codes can be specified with a hyphen separating numbers.

`tunnel` *name*
Specifies a tunnel network interface, as configured with `ifconfig(1M)`. If a tunnel of *name* does not yet exist, the policy entries are added anyway, and joined with the tunnel state when it is created. If a tunnel is unplumbed, its policy entries disappear.

`negotiate` *tunnel*
`negotiate` *transport*
For per-tunnel security, specify whether the IPsec SAs protecting the traffic should be tunnel-mode SAs or transport-mode SAs. If transport-mode SAs are specified, no addresses can appear in the policy entry. Transport-mode is backward compatible with Solaris 9, and tunnel IPsec policies configured with `ifconfig(1M)` will show up as transport mode entries here.

Policy entries may contain the following (name-value) pairs in the properties field. Each (name-value) pair may appear only once in a given policy entry.

`auth_algs`
An acceptable value following this implies that IPsec AH header will be present in the outbound datagram. Values following this describe the authentication algorithms that will be used while applying the IPsec AH on outbound datagrams and verified to be present on inbound datagrams. See *RFC 2402*.

This entry can contain either a string or a decimal number.

`string`
This should be either MD5 or HMAC-MD5 denoting the HMAC-MD5 algorithm as described in *RFC 2403*, and SHA1, or HMAC-SHA1 or SHA or HMAC-SHA denoting the HMAC-SHA algorithm described in *RFC 2404*. You can use the `ipsecalgs(1M)` command to obtain the complete list of authentication algorithms.

The string can also be `ANY`, which denotes no-preference for the algorithm. Default algorithms will be chosen based upon the SAs available at this time for manual SAs and the key negotiating daemon for automatic SAs. Strings are not case-sensitive.

`number`
A number in the range 1-255. This is useful when new algorithms can be dynamically loaded.

If *auth_algs* is not present, the AH header will not be present in the outbound datagram, and the same will be verified for the inbound datagram.

`encr_algs`
An acceptable value following this implies that IPsec ESP header will be present in the outbound datagram. The value following this describes the encryption algorithms that will be used to apply the IPsec ESP protocol to outbound datagrams and verify it to be present on inbound datagrams. See *RFC 2406*.

This entry can contain either a string or a decimal number. Strings are not case-sensitive.

`string`
Can be one of the following:

string value:	Algorithm Used:	See RFC:
DES or DES-CBC	DES-CBC	2405
3DES or 3DES-CBC	3DES-CBC	2451
BLOWFISH or BLOWFISH-CBC	BLOWFISH-CBC	2451
AES or AES-CBC	AES-CBC	2451

You can use the ipsecalgs(1M) command to obtain the complete list of authentication algorithms.

The value can be NULL, which implies a NULL encryption, pursuant to *RFC 2410*. This means that the payload will not be encrypted. The string can also be ANY, which indicates no-preference for the algorithm. Default algorithms will be chosen depending upon the SAs available at the time for manual SAs and upon the key negotiating daemon for automatic SAs. Strings are not case-sensitive.

`number`
A decimal number in the range 1-255. This is useful when new algorithms can be dynamically loaded.

`encr_auth_algs`
An acceptable value following `encr_auth_algs` implies that the IPsec ESP header will be present in the outbound datagram. The values following `encr_auth_algs` describe the authentication algorithms that will be used while applying the IPsec ESP protocol on outbound datagrams and verified to be present on inbound datagrams. See *RFC 2406*. This entry can contain either a string or a number. Strings are case-insensitive.

`string`
Valid values are the same as the ones described for `auth_algs` above.

`number`
This should be a decimal number in the range 1-255. This is useful when new algorithms can be dynamically loaded.

If `encr_algs` is present and `encr_auth_algs` is not present in a policy entry, the system will use an ESP SA regardless of whether the SA has an authentication algorithm or not.

If `encr_algs` is not present and `encr_auth_algs` is present in a policy entry, null encryption will be provided, which is equivalent to `encr_algs` with `NULL`, for outbound and inbound datagrams.

If both `encr_algs` and `encr_auth_algs` are not present in a policy entry, ESP header will not be present for outbound datagrams and the same will be verified for inbound datagrams.

If both `encr_algs` and `encr_auth_algs` are present in a policy entry, ESP header with integrity checksum will be present on outbound datagrams and the same will be verified for inbound datagrams.

For `encr_algs`, `encr_auth_algs`, and `auth_algs` a key length specification may be present. This is either a single value specifying the only valid key length for the algorithm or a range specifying the valid minimum and/or maximum key lengths. Minimum or maximum lengths may be omitted.

`dir`
Values following this decides whether this entry is for outbound or inbound datagram. Valid values are strings that should be one of the following:

`out`
This means that this policy entry should be considered only for outbound datagrams.

`in`
This means that this policy entry should be considered only for inbound datagrams.

`both`
This means that this policy entry should be considered for both inbound and outbound datagrams

This entry is not needed when the action is “apply”, “permit” or “ipsec”. But if it is given while the action is “apply” or “permit”, it should be “out” or “in” respectively. This is mandatory when the action is “bypass”.

`sa`
Values following this decide the attribute of the security association. Value indicates whether a unique security association should be used or any existing SA can be used. If there is a policy requirement, SAs are created dynamically on the first outbound datagram using the key management daemon. Static SAs can be created using `ipseckey(1M)`. The values used here determine whether a new SA will be used/obtained. Valid values are strings that could be one of the following:

`unique`
Unique Association. A new/unused association will be obtained/used for packets matching this policy entry. If an SA that was previously used by the same 5 tuples, that is, {Source address, Destination address, Source port, Destination Port, Protocol (for example, TCP/UDP)} exists, it will be reused. Thus uniqueness is expressed by the 5 tuples given above. The security association used by the above 5 tuples will not be used by any other socket. For inbound datagrams, uniqueness will not be verified.

For tunnel-mode tunnels, `unique` is ignored. SAs are assigned per-rule in tunnel-mode tunnels. For transport-mode tunnels, `unique` is implicit, because the enforcement happens only on the outer-packet addresses and protocol value of either IPv4-in-IP or IPv6-in-IP.

`shared`
Shared association. If an SA exists already for this source-destination pair, it will be used. Otherwise a new SA will be obtained. This is the default.

This is mandatory only for outbound policy entries and should not be given for entries whose action is “bypass”. If this entry is not given for inbound entries, for example, when “dir” is in or “action” is permit, it will be assumed to be shared.

Action follows the pattern and should be given before properties. It should be one of the following and this field is mandatory.

`ipsec`
Use IPsec for the datagram as described by the properties, if the pattern matches the datagram. If ipsec is given without a dir spec , the pattern is matched to incoming and outgoing datagrams.

`apply`
Apply IPsec to the datagram as described by the properties, if the pattern matches the datagram. If `apply` is given, the pattern is matched only on the outbound datagram.

`permit`
Permit the datagram if the pattern matches the incoming datagram and satisfies the constraints described by the properties. If it does not satisfy the properties, discard the datagram. If `permit` is given, the pattern is matched only for inbound datagrams.

`bypass`
`pass`
Bypass any policy checks if the pattern matches the datagram. `dir` in the properties decides whether the check is done on outbound or inbound datagrams. All the `bypass` entries are checked before checking with any other policy entry in the system. This has the highest precedence over any other entries. `dir` is the only field that should be present when action is `bypass`.

`drop`
Drop any packets that match the pattern.

If the file contains multiple policy entries, for example, they are assumed to be listed in the order in which they are to be applied. In cases of multiple entries matching the outbound and inbound datagram, the first match will be taken. The system will reorder the policy entry, that is, add the new entry before the old entry, only when:

The level of protection is "stronger" than the old level of protection.

Currently, strength is defined as:

```
AH and ESP > ESP > AH
```

The standard uses of AH and ESP were what drove this ranking of "stronger". There are flaws with this. ESP can be used either without authentication, which will allow cut-and-paste or replay attacks, or without encryption, which makes it equivalent or slightly weaker than AH. An administrator should take care to use ESP properly. See `ipsecesp(7P)` for more details.

If the new entry has `bypass` as action, `bypass` has the highest precedence. It can be added in any order, and the system will still match all the `bypass` entries before matching any other entries. This is useful for key management daemons which can use this feature to bypass IPsec as it protects its own traffic.

Entries with both AH (`auth_algs` present in the policy entry) and ESP (`encr_auth_algs` or `encr_auth_algs` present in the policy entry) protection are ordered after all the entries with

AH and ESP and before any AH–only and ESP–only entries. In all other cases the order specified by the user is not modified, that is, newer entries are added at the end of all the old entries. See Examples.

A new entry is considered duplicate of the old entry if an old entry matches the same traffic pattern as the new entry. See Examples for information on duplicates.

Security If, for example, the policy file comes over the wire from an NFS mounted file system, an adversary can modify the data contained in the file, thus changing the policy configured on the machine to suit his needs. Administrators should be cautious about transmitting a copy of the policy file over a network.

To prevent non-privileged users from modifying the security policy, ensure that the configuration file is writable only by trusted users.

The configuration file is defined by a property of the `policy` `smf(5)` service. The default configuration file, is `/etc/inet/ipsecinit.conf`. This can be changed using the `svcprop(1)` command. See `NOTES` for more details.

The policy description language supports the use of tokens that can be resolved by means of a name service, using functions such as `gethostbyname(3NSL)`. While convenient, these functions are only secure as the name service the system is configured to use. Great care should be taken to secure the name service if it is used to resolve elements of the security policy.

If your source address is a host that can be looked up over the network and your naming system itself is compromised, then any names used will no longer be trustworthy.

If the name switch is configured to use a name service that is not local to the system, bypass policy entries might be required to prevent the policy from preventing communication to the name service. See `nsswitch.conf(4)`.

Policy is latched for TCP/UDP sockets on which a `connect(3SOCKET)` or `accept(3SOCKET)` has been issued. Adding new policy entries will not have any effect on them. This feature of latching may change in the future. It is not advisable to depend upon this feature.

The `ipsecconf` command can only be run by a user who has sufficient privilege to open the `pf_key(7P)` socket. The appropriate privilege can be assigned to a user with the Network IPsec Management profile. See `profiles(1)`, `rbac(5)`, `prof_attr(4)`.

Make sure to set up the policies before starting any communications, as existing connections may be affected by the addition of new policy entries. Similarly, do not change policies in the middle of a communication.

Note that certain `ndd` tunables affect how policies configured with this tool are enforced; see `ipsecesp(7P)` for more details.

Examples

EXAMPLE 1 Protecting Outbound TCP Traffic With ESP and the AES Algorithm

The following example specified that any TCP packet from spiderweb to arachnid should be encrypted with AES, and the SA could be a shared one. It does not verify whether or not the inbound traffic is encrypted.

```
#
# Protect the outbound TCP traffic between hosts spiderweb
# and arachnid with ESP and use AES algorithm.
#
{
     laddr spiderweb
     raddr arachnid
     ulp tcp
     dir out
} ipsec {
         encr_algs AES
}
```

EXAMPLE 2 Verifying Whether or Not Inbound Traffic is Encrypted

Example 1 does not verify whether or not the inbound traffic is encrypted. The entry in this example protects inbound traffic:

```
#
# Protect the TCP traffic on inbound with ESP/DES from arachnid
# to spiderweb
#
{
        laddr spiderweb
        raddr arachnid
        ulp tcp
        dir in
} ipsec {
        encr_algs AES
}
```

`sa` can be absent for inbound policy entries as it implies that it can be a shared one. Uniqueness is not verified on inbound. Note that in both the above entries, authentication was never specified. This can lead to cut and paste attacks. As mentioned previously, though the authentication is not specified, the system will still use an ESP SA with `encr_auth_alg` specified, if it was found in the SA tables.

EXAMPLE 3 Protecting All Traffic Between Two Hosts

The following example protects both directions at once:

```
{
        laddr spiderweb
        raddr arachnid
```

EXAMPLE 3 Protecting All Traffic Between Two Hosts *(Continued)*

```
        ulp tcp
} ipsec {
        encr_algs AES
}
```

EXAMPLE 4 Authenticating All Inbound Traffic to the Telnet Port

This entry specifies that any inbound datagram to telnet port should come in authenticated with the SHA1 algorithm. Otherwise the datagram should not be permitted. Without this entry, traffic destined to port number 23 can come in clear. `sa` is not specified, which implies that it is shared. This can be done only for inbound entries. You need to have an equivalent entry to protect outbound traffic so that the outbound traffic is authenticated as well, remove the dir.

```
#
# All the inbound traffic to the telnet port should be
# authenticated.
#
{
       lport telnet
       dir in
} ipsec {
       auth_algs sha1
}
```

EXAMPLE 5 Verifying Inbound Traffic is Null-Encrypted

The first entry specifies that any packet with address host-B should not be checked against any policies. The second entry specifies that all inbound traffic from network-B should be encrypted with a `NULL` encryption algorithm and the MD5 authentication algorithm. `NULL` encryption implies that ESP header will be used without encrypting the datagram. As the first entry is `bypass` it need not be given first in order, as `bypass` entries have the highest precedence. Thus any inbound traffic will be matched against all `bypass` entries before any other policy entries.

```
#
# Make sure that all inbound traffic from network-B is NULL
# encrypted, but bypass for host-B alone from that network.
# Add the bypass first.
{
raddr host-B
    dir in
} bypass {}

# Now add for network-B.
{
    raddr network-B/16
```

EXAMPLE 5 Verifying Inbound Traffic is Null-Encrypted *(Continued)*

```
    dir in
} ipsec {
encr_algs NULL
encr_auth_algs md5
}
```

EXAMPLE 6 Entries to Bypass Traffic from IPsec

The first two entries provide that any datagram leaving the machine with source port 53 or coming into port number 53 should not be subjected to IPsec policy checks, irrespective of any other policy entry in the system. Thus the latter two entries will be considered only for ports other than port number 53.

```
#
# Bypass traffic for port no 53
     #
{lport 53} bypass {}
{rport 53} bypass {}
{raddr spiderweb } ipsec {encr_algs any sa unique}
```

EXAMPLE 7 Protecting Outbound Traffic

```
 #
     # Protect the outbound traffic from all interfaces.
     #
{raddr spiderweb dir out} ipsec {auth_algs any sa unique}
```

If the gethostbyname(3XNET) call for spiderweb yields multiple addresses, multiple policy entries will be added for all the source address with the same properties.

```
{
    laddr arachnid
    raddr spiderweb
    dir in
} ipsec {auth_algs any sa unique}
```

If the `gethostbyname`(3XNET) call for spiderweb and the `gethostbyname`(3XNET) call for arachnid yield multiple addresses, multiple policy entries will be added for each (`saddr daddr`) pair with the same properties. Use `ipsecconf -l` to view all the policy entries added.

EXAMPLE 8 Bypassing Unauthenticated Traffic

```
#
# Protect all the outbound traffic with ESP except any traffic
# to network-b which should be authenticated and bypass anything
# to network-c
#
{raddr network-b/16 dir out} ipsec {auth_algs any}
```

EXAMPLE 8 Bypassing Unauthenticated Traffic *(Continued)*

```
{dir out} ipsec {encr_algs any}
{raddr network-c/16 dir out} bypass {} # NULL properties
```

Note that `bypass` can be given anywhere and it will take precedence over all other entries. `NULL` pattern matches all the traffic.

EXAMPLE 9 Encrypting IPv6 Traffic with 3DES and MD5

The following entry on the host with the link local address `fe80::a00:20ff:fe21:4483` specifies that any outbound traffic between the hosts wtih IPv6 link-local addresses `fe80::a00:20ff:fe21:4483` and `fe80::a00:20ff:felf:e346` must be encrypted with 3DES and MD5.

```
{
    laddr fe80::a00:20ff:fe21:4483
    raddr fe80::a00:20ff:felf:e346
    dir out
} ipsec {
    encr_algs 3DES
    encr_auth_algs MD5
}
```

EXAMPLE 10 Verifying IPv6 Traffic is Authenticated with SHA1

The following two entries require that all IPv6 traffic to and from the IPv6 site-local network `fec0:abcd::0/32` be authenticated with SHA1.

```
{raddr fec0:abcd::0/32} ipsec { auth_algs SHA1 }
```

EXAMPLE 11 Key Lengths

```
# use aes at any key length
{raddr spiderweb} ipsec {encr_algs aes}

# use aes with a 192 bit key
{raddr spiderweb} ipsec {encr_algs aes(192)}

# use aes with any key length up to 192 bits
# i.e. 192 bits or less
{raddr spiderweb} ipsec {encr_algs aes(..192)}

# use aes with any key length of 192 or more
# i.e. 192 bits or more
{raddr spiderweb} ipsec {encr_algs aes(192..)}

#use aes with any key from 192 to 256 bits
{raddr spiderweb} ipsec {encr_algs aes(192..256)}
```

EXAMPLE 11 Key Lengths *(Continued)*

```
#use any algorithm with a key of 192 bits or longer
{raddr spiderweb} ipsec {encr_algs any(192..)}
```

EXAMPLE 12 Correct and Incorrect Policy Entries

The following are examples of correctly formed policy entries:

```
{ raddr that_system rport telnet } ipsec { encr_algs 3des encr_auth_algs
sha1 sa shared}

{
        raddr that_system
        rport telnet
} ipsec {
        encr_algs 3des
        encr_auth_algs sha1
        sa shared
}

{ raddr that_system rport telnet } ipsec
        { encr_algs 3des encr_auth_algs sha1 sa shared}

{ raddr that_system rport telnet } ipsec
        { encr_algs 3des encr_auth_algs sha1 sa shared} or ipsec
        { encr_algs aes encr_auth_algs sha1 sa shared}
```

...and the following is an incorrectly formed entry:

```
{ raddr that_system rport telnet } ipsec
        { encr_algs 3des encr_auth_algs sha1 sa shared}
        or ipsec { encr_algs aes encr_auth_algs sha1 sa shared}
```

In the preceding, incorrect entry, note that the third line begins with "`or ipsec`". Such an entry causes `ipsecconf` to return an error.

EXAMPLE 13 Allowing Neighbor Discovery to Occur in the Clear

The following two entries require that all IPv6 traffic to and from the IPv6 site-local network `fec0:abcd::0/32` be authenticated with SHA1. The second entry allows neighbor discovery to operate correctly.

```
{raddr fec0:abcd::0/32} ipsec { auth_algs SHA1 }
{raddr fec0:abcd::0/32 ulp ipv6-icmp type 133-137  dir both }
    pass { }
```

EXAMPLE 14 Using "or"

The following entry allows traffic using the AES or Blowfish algorithms from the remote machine spiderweb:

EXAMPLE 14 Using "or" *(Continued)*

```
{raddr spiderweb} ipsec {encr_algs aes} or ipsec {encr_algs blowfish}
```

EXAMPLE 15 Configuring a Tunnel to be Backward-Compatible with Solaris 9

The following example is equivalent to "`encr_algs aes encr_auth_algs md5`" in `ifconfig(1M)`:

```
{tunnel ip.tun0 negotiate transport} ipsec {encr_algs aes
                                                encr_auth_algs md5}
```

EXAMPLE 16 Configuring a Tunnel to a VPN client with an Assigned Address

The following example assumes a distinct "inside" network with its own topology, such that a client's default route goes "inside".

```
# Unlike route(1m), the default route has to be spelled-out.
{tunnel ip.tun0 negotiate tunnel raddr client-inside/32
laddr 0.0.0.0/0} ipsec {encr_algs aes encr_auth_algs sha1}
```

EXAMPLE 17 Transit VPN router between Two Tunnelled Subnets and a Third

The following example specifies a configuration for a VPN router that routes between two tunnelled subnets and a third subnet that is on-link. Consider remote-site A, remote-site B, and local site C, each with a `/24` address allocation.

```
# ip.tun0 between me (C) and remote-site A.
# Cover remote-site A to remote-side B.
{tunnel ip.tun0 negotiate tunnel raddr A-prefix/24 laddr
B-prefix/24} ipsec {encr_algs 3des encr_auth_algs md5}

# Cover remote-site A traffic to my subnet.
{tunnel ip.tun0 negotiate tunnel raddr A-prefix/24 laddr
C-prefix/24} ipsec {encr_algs 3des encr_auth_algs md5}

# ip.tun1 between me (C) and remote-site B.
# Cover remote-site B to remote-site A.
{tunnel ip.tun1 negotiate tunnel raddr B-prefix/24 laddr
A-prefix/24} ipsec {encr_algs aes encr_auth_algs sha1}

# Cover remote-site B traffic to my subnet.
{tunnel ip.tun1 negotiate tunnel raddr B-prefix/24 laddr
C-prefix/24} ipsec {encr_algs aes encr_auth_algs md5}
```

Files `/var/run/ipsecpolicy.conf`

Cache of IPsec policies currently configured for the system, maintained by `ipsecconf` command. Do not edit this file.

`/etc/inet/ipsecinit.conf`
File containing IPsec policies to be installed at system restart by the `policy` `smf(5)` service. See `NOTES` for more information.

`/etc/inet/ipsecinit.sample`
Sample input file for `ipseconf`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed

See Also `auths(1)`, `profiles(1)`, `svcprop(1)`, `svcs(1)`, `in.iked(1M)`, `init(1M)`, `ifconfig(1M)`, `ipsecalgs(1M)`, `ipseckey(1M)`, `svcadm(1M)`, `svccfg(1M)`, `gethostbyname(3NSL)`, `accept(3SOCKET)`, `connect(3SOCKET)`, `gethostbyname(3XNET)`, `getnetbyname(3XNET)`, `getprotobyname(3XNET)`, `getservbyname(3XNET)`, `getaddrinfo(3SOCKET)`, `socket(3SOCKET)`, `ike.config(4)`, `nsswitch.conf(4)`, `prof_attr(4)`, `user_attr(4)`, `attributes(5)`, `rbac(5)`, `smf(5)`, `tun(7M)`, `ipsecah(7P)` , `ipsecesp(7P)`, `pf_key(7P)`

Glenn, R. and Kent, S. *RFC 2410, The NULL Encryption Algorithm and Its Use With IPsec*. The Internet Society. 1998.

Kent, S. and Atkinson, R. *RFC 2402, IP Authentication Header*.The Internet Society. 1998.

Kent, S. and Atkinson, R. *RFC 2406, IP Encapsulating Security Payload (ESP)*. The Internet Society. 1998.

Madsen, C. and Glenn, R. *RFC 2403, The Use of HMAC-MD5-96 within ESP and AH*. The Internet Society. 1998.

Madsen, C. and Glenn, R. *RFC 2404, The Use of HMAC-SHA-1-96 within ESP and AH*. The Internet Society. 1998.

Madsen, C. and Doraswamy, N. *RFC 2405, The ESP DES-CBC Cipher Algorithm With Explicit IV*. The Internet Society. 1998.

Pereira, R. and Adams, R. *RFC 2451, The ESP CBC-Mode Cipher Algorithms*. The Internet Society. 1998.

Frankel, S. and Kelly, R. Glenn, *The AES Cipher Algorithm and Its Use With IPsec*. 2001.

Diagnostics Bad “string” on line *N*.
Duplicate “string” on line *N*.
string refers to one of the names in pattern or properties. A Bad string indicates that an argument is malformed; a Duplicate string indicates that there are multiple arguments of a similar type, for example, multiple Source Address arguments.

Interface name already selected
Dual use of `-i` *name* and *name*,*index* for an index.

Error before or at line *N*.
Indicates parsing error before or at line *N*.

Non-existent index
Reported when the *index* for delete is not a valid one.

spd_msg return: File exists
Reported when there is already a policy entry that matches the traffic of this new entry.

Notes IPsec manual keys are managed by the service management facility, `smf(5)`. The services listed below manage the components of IPsec. These services are delivered as follows:

```
svc:/network/ipsec/policy:default (enabled)
svc:/network/ipsec/ipsecalgs:default (enabled)
svc:/network/ipsec/manual-key:default (disabled)
svc:/network/ipsec/ike:default (disabled)
```

The manual-key service is delivered disabled. The system administrator must create manual IPsec Security Associations (SAs), as described in `ipseckey(1M)`, before enabling that service.

The policy service is delivered enabled, but without a configuration file, so that, as a starting condition, packets are not protected by IPsec. After you create the configuration file `/etc/inet/ipsecinit.conf`, as described in this man page, and refresh the service (`svcadm refresh`, see below), the policy contained in the configuration file is applied. If there is an error in this file, the service enters maintenance mode.

Services that are delivered disabled are delivered that way because the system administrator must create configuration files for those services before enabling them. See `ike.config(4)` for the `ike` service.

See `ipsecalgs(1M)` for the `ipsecalgs` service.

The correct administrative procedure is to create the configuration file for each service, then enable each service using `svcadm(1M)`.

If the configuration needs to be changed, edit the configuration file then refresh the service, as follows:

```
example# svcadm refresh policy
```

The `smf(5)` framework will record any errors in the service-specific log file. Use any of the following commands to examine the `logfile` property:

```
example# svcs -l policy
example# svcprop policy
example# svccfg -s policy listprop
```

The following property is defined for the `policy` service:

`config/config_file`

This property can be modified using svccfg(1M) by users who have been assigned the following authorization:

`solaris.smf.value.ipsec`

See auths(1), user_attr(4), rbac(5).

The service needs to be refreshed using svcadm(1M) before the new property is effective. General non-modifiable properties can be viewed with the svcprop(1) command.

```
# svccfg -s ipsec/policy setprop config/config_file = /new/config_file
# svcadm refresh policy
```

Administrative actions on this service, such as enabling, disabling, refreshing, and requesting restart can be performed using svcadm(1M). A user who has been assigned the authorization shown below can perform these actions:

`solaris.smf.manage.ipsec`

The service's status can be queried using the svcs(1) command.

The `ipsecconf` command is designed to be managed by the `policy` smf(5) service. While the `ipsecconf` command can be run from the command line, this is discouraged. If the `ipsecconf` command is to be run from the command line, the `policy` smf(5) service should be disabled first. See svcadm(1M).

Name ipseckey – manually manipulate an IPsec Security Association Database (SADB)

Synopsis

```
ipseckey  [-nvp]

ipseckey  [-nvp] -f filename

ipseckey  -c filename

ipseckey  [-nvp] [delete | delete-pair | get] SA_TYPE {EXTENSION value...}

ipseckey  [-np] [monitor |  passive_monitor |  pmonitor]

ipseckey  [-nvp] flush {SA_TYPE}

ipseckey  [-nvp] dump {SA_TYPE}

ipseckey  [-nvp] save SA_TYPE {filename}

ipseckey  [-nvp] -s filename
```

Description The `ipseckey` command is used to manually manipulate the security association databases of the network security services, `ipsecah(7P)` and `ipsecesp(7P)`. You can use the `ipseckey` command to set up security associations between communicating parties when automated key management is not available.

While the `ipseckey` utility has only a limited number of general options, it supports a rich command language. The user may specify requests to be delivered by means of a programmatic interface specific for manual keying. See `pf_key(7P)`. When `ipseckey` is invoked with no arguments, it will enter an interactive mode which prints a prompt to the standard output and accepts commands from the standard input until the end-of-file is reached. Some commands require an explicit security association ("SA") type, while others permit the SA type to be unspecified and act on all SA types.

`ipseckey` uses a `PF_KEY` socket and the message types `SADB_ADD`, `SADB_DELETE`, `SADB_GET`, `SADB_UPDATE`, `SADB_FLUSH`, and `SADB_X_PROMISC`. Thus, you must be a superuser to use this command.

`ipseckey` handles sensitive cryptographic keying information. Please read the `Security` section for details on how to use this command securely.

Options

`-c` [*filename*]
Analogous to the `-f` option (see following), except that the input is not executed but only checked for syntactical correctness. Errors are reported to `stderr`. This option is provided to debug configurations without making changes. See `SECURITY` and "Service Management Facility" for more information.

`-f` [*filename*]
Read commands from an input file, *filename*. The lines of the input file are identical to the command line language. The `load` command provides similar functionality. The `-s` option or the `save` command can generate files readable by the `-f` argument.

`-n`
Prevent attempts to print host and network names symbolically when reporting actions. This is useful, for example, when all name servers are down or are otherwise unreachable.

`-p`
Paranoid. Do not print any keying material, even if saving SAs. Instead of an actual hexadecimal digit, print an `X` when this flag is turned on.

`-s` [*filename*]
The opposite of the `-f` option. If '`-`' is given for a *filename*, then the output goes to the standard output. A snapshot of all current `SA` tables will be output in a form readable by the `-f` option. The output will be a series of `add` commands, but with some names not used. This occurs because a single name may often indicate multiple addresses.

`-v`
Verbose. Print the messages being sent into the `PF_KEY` socket, and print raw seconds values for lifetimes.

Commands `add`
Add an SA. Because it involves the transfer of keying material, it cannot be invoked from the shell, lest the keys be visible in `ps(1)` output. It can be used either from the interactive `ipseckey>` prompt or in a command file specified by the `-f` command. The `add` command accepts all extension-value pairs described below.

`update`
Update SA lifetime, and in the cases of larval SAs (leftover from aborted automated key management), keying material and other extensions. Like `add`, this command cannot be invoked from the shell because keying material would be seen by the `ps(1)` command. It can be used either from the interactive `ipseckey>` prompt or in a command file specified by the `-f` command. The `update` command accepts all extension-value pairs, but normally is only used for SA lifetime updates.

`update-pair`
As update, but apply the update to the SA and its paired SA, if there is one.

`delete`
Delete a specific SA from a specific `SADB`. This command requires the `spi` extension, and the `dest` extension for IPsec SAs. Other extension-value pairs are superfluous for a delete message. If the SA to be deleted is paired with another SA, the SA is deleted and the paired SA is updated to indicate that it is now unpaired.

`delete-pair`
Delete a specific SA from a specific SADB. If the SA is paired with another SA, delete that SA too. This command requires the `spi` extension and the `dest` extension for the IPsec SA, or its pair.

`get`
Lookup and display a security association from a specific `SADB`. Like `delete`, this command only requires `spi` and `dest` for IPsec.

`flush`
Remove all SA for a given `SA_TYPE`, or all SA for all types.

`monitor`
Continuously report on any `PF_KEY` messages. This uses the `SADB_X_PROMISC` message to enable messages that a normal `PF_KEY` socket would not receive to be received. See `pf_key(7P)`.

`passive_monitor`
Like monitor, except that it does not use the `SADB_X_PROMISC` message.

`pmonitor`
Synonym for `passive_monitor`.

`dump`
Will display all SAs for a given SA type, or will display all SAs. Because of the large amount of data generated by this command, there is no guarantee that all SA information will be successfully delivered, or that this command will even complete.

`save`
Is the command analog of the `-s` option. It is included as a command to provide a way to snapshot a particular SA type, for example, `esp` or `ah`.

`help`
Prints a brief summary of commands.

SA_TYPE `all`
Specifies all known SA types. This type is only used for the `flush` and `dump` commands. This is equivalent to having no SA type for these commands.

`ah`
Specifies the IPsec Authentication Header ("`AH`") SA.

`esp`
Specifies the IPsec Encapsulating Security Payload ("`ESP`") SA.

Extension Value Types

Commands like `add`, `delete`, `get`, and `update` require that certain extensions and associated values be specified. The extensions will be listed here, followed by the commands that use them, and the commands that require them. Requirements are currently documented based upon the IPsec definitions of an SA. Required extensions may change in the future. `<number>` can be in either hex (`0xnnn`), decimal (`nnn`) or octal (`0nnn`).`<string>` is a text string. `<hexstr>` is a long hexadecimal number with a bit-length. Extensions are usually paired with values; however, some extensions require two values after them.

`spi` <*number*>
Specifies the security parameters index of the `SA`. This extension is required for the `add`, `delete`, `get` and `update` commands.

`pair-spi` <*number*>
When `pair-spi` is used with the `add` or `update` commands, the SA being added or updated will be paired with the SA defined by `pair-spi`. A pair of SAs can be updated or deleted with a single command.

The two SAs that make up the pair need to be in opposite directions from the same pair of IP addresses. The command will fail if either of the SAs specified are already paired with another SA.

If the pair-spi token is used in a command and the SA defined by pair-spi does not exist, the command will fail. If the command was `add` and the pairing failed, the SA to be added will instead be removed.

`inbound | outbound`
These optional flags specify the direction of the SA. When the `inbound` or `outbound` flag is specified with the `add` command, the kernel will insert the new SA into the specified hash table for faster lookups. If the flag is omitted, the kernel will decide into which hash table to insert the new SA based on its knowledge the IP addresses specified with the `src` and `dst` extensions.

When these flags are used with the `update`, `delete`, `update-pair` or `get` commands, the flags provide a hint as to the hash table in which the kernel should find the SA.

`replay` <*number*>
Specifies the replay window size. If not specified, the replay window size is assumed to be zero. It is not recommended that manually added SAs have a replay window. This extension is used by the `add` and `update` commands.

`replay_value` <*number*>
Specifies the replay value of the SA. This extension is used by the `add` and `update` commands.

`state` <*string*>|<*number*>
Specifies the SA state, either by numeric value or by the strings "`larval`", "`mature`", "`dying`" or "`dead`". If not specified, the value defaults to `mature`. This extension is used by the `add` and `update` commands.

`auth_alg` <*string*>|<*number*>
`authalg <string>|<number>`
Specifies the authentication algorithm for an `SA`, either by numeric value, or by strings indicating an algorithm name. Current authentication algorithms include:

`HMAC-MD5`
`md5`, `hmac-md5`

`HMAC-SH-1`
`sha`, `sha-1`, `hmac-sha1`, `hmac-sha`

HMAC-SHA-256
sha256, sha-256, hmac-sha256, hmac-sha-256

HMAC-SHA-384
sha384, sha-384, hmac-sha384, hmac-sha-384

HMAC-SHA-512
sha512, sha-512, hmac-sha512, hmac-sha-512

Often, algorithm names will have several synonyms. This extension is required by the add command for certain SA types. It is also used by the update command.

Use the ipsecalgs(1M) command to obtain the complete list of authentication algorithms.

encr_alg <*string*>|<*number*>
encralg <*string*>|<*number*>
Specifies the encryption algorithm for an SA, either by numeric value, or by strings indicating an algorithm name. Current encryption algorithms include DES ("des"), Triple-DES ("3des"), Blowfish ("blowfish"), and AES ("aes"). This extension is required by the add command for certain SA types. It is also used by the update command.

Use the ipsecalgs(1M) command to obtain the complete list of encryption algorithms.

The next six extensions are lifetime extensions. There are two varieties, "hard" and "soft". If a hard lifetime expires, the SA will be deleted automatically by the system. If a soft lifetime expires, an SADB_EXPIRE message will be transmitted by the system, and its state will be downgraded to dying from mature. See pf_key(7P). The monitor command to key allows you to view SADB_EXPIRE messages.

idle_addtime <*number*>
idle_usetime <*number*>
Specifies the number of seconds that this SA can exist if the SA is not used before the SA is revalidated. If this extension is not present, the default value is half of the hard_addtime (see below). This extension is used by the add and update commands.

soft_bytes <*number*>
hard_bytes <*number*>
Specifies the number of bytes that this SA can protect. If this extension is not present, the default value is zero, which means that the SA will not expire based on the number of bytes protected. This extension is used by the add and update commands.

soft_addtime <*number*>
hard_addtime <*number*>
Specifies the number of seconds that this SA can exist after being added or updated from a larval SA. An update of a mature SA does not reset the initial time that it was added. If this extension is not present, the default value is zero, which means the SA will not expire based on how long it has been since it was added. This extension is used by the add and update commands.

`soft_usetime` <*number*>
`hard_usetime` <*number*>
Specifies the number of seconds this SA can exist after first being used. If this extension is not present, the default value is zero, which means the SA will not expire based on how long it has been since it was added. This extension is used by the `add` and `update` commands.

`saddr` *address* | *name*
`srcaddr` *address* | *name*
`saddr6` *IPv6 address*
`srcaddr6` *IPv6 address*
`src` *address* | *name*
`src6` *IPv6 address*
`srcaddr` *address* and `src` *address* are synonyms that indicate the source address of the SA. If unspecified, the source address will either remain unset, or it will be set to a wildcard address if a destination address was supplied. To not specify the source address is valid for IPsec SAs. Future SA types may alter this assumption. This extension is used by the `add`, `update`, `get` and `delete` commands.

`daddr` <*address*>|<*name*>
`dstaddr` <*address*>|<*name*>
`daddr6` <*IPv6 address*>|<*name*>
`dstaddr6` <*IPv6 address*>|<*name*>
`dst` <*addr*>|<*name*>
`dst6` <*IPv6 address*>|<*name*>
`dstaddr` <*addr*> and `dst` <*addr*> are synonyms that indicate the destination address of the `SA`. If unspecified, the destination address will remain unset. Because IPsec SAs require a specified destination address and `spi` for identification, this extension, with a specific value, is required for the `add`, `update`, `get` and `delete` commands.

If a name is given, `ipseckey` will attempt to invoke the command on multiple SAs with all of the destination addresses that the name can identify. This is similar to how `ipsecconf` handles addresses.

If `dst6` or `dstaddr6` is specified, only the IPv6 addresses identified by a name are used.

`sport` <*portnum*>
`sport` specifies the source port number for an SA. It should be used in combination with an upper-layer protocol (see below), but it does not have to be.

`dport` <*portnum*>
sport specifies the destination port number for an SA. It should be used in combination with an upper-layer protocol (see below), but it does not have to be.

`encap` <*protocol*>
Identifies the protocol used to encapsulate NAT-traversal IPsec packets. Other NAT-traversal parameters (`nat_*`) are below. The only acceptable value for <*protocol*> currently is `udp`.

`proto` <*protocol number*>
`ulp` <*protocol number*>
`proto`, and its synonym `ulp`, specify the IP protocol number of the SA.

`nat_loc` <*address*>|<*name*>
If the local address in the SA (source or destination) is behind a NAT, this extension indicates the NAT node's globally-routable address. This address can match the SA's local address if there is a `nat_lport` (see below) specified.

`nat_rem` <*address*>|<*name*>
If the remote address in the SA (source or destination) is behind a NAT, this extension indicates that node's internal (that is, behind-the-NAT) address. This address can match the SA's local address if there is a `nat_rport` (see below) specified.

`nat_lport` <*portnum*>
Identifies the local UDP port on which encapsulation of ESP occurs.

`nat_rport` <*portnum*>
Identifies the remote UDP port on which encapsulation of ESP occurs.

`isrc` <*address*> | <*name*>[/<*prefix*>]
`innersrc` <*address*> | <*name*>[/<*prefix*>]
`isrc6` <*address*> | <*name*>[/<*prefix*>]
`innersrc6` <*address*> | <*name*>[/<*prefix*>]
`proxyaddr` <*address*> | <*name*>[/<*prefix*>]
`proxy` <*address*> | <*name*>[/<*prefix*>]
`isrc` <*address*>[/<*prefix*>] and `innersrc` <*address*>[/<*prefix*>] are synonyms. They indicate the inner source address for a tunnel-mode SA.

An inner-source can be a prefix instead of an address. As with other address extensions, there are IPv6-specific forms. In such cases, use only IPv6-specific addresses or prefixes.

Previous versions referred to this value as the proxy address. The usage, while deprecated, remains.

`idst` <*address*> | <*name*>[/<*prefix*>]
`innerdst` <*address*> | <*name*>[/<*prefix*>]
`idst6` <*address*> | <*name*>[/<*prefix*>]
`innerdst6` <*address*> | <*name*>[/<*prefix*>]
`idst` <*address*>[/<*prefix*>] and `innerdst` <*address*>[/<*prefix*>] are synonyms. They indicate the inner destination address for a tunnel-mode SA.

An inner-destination can be a prefix instead of an address. As with other address extensions, there are IPv6-specific forms. In such cases, use only IPv6-specific addresses or prefixes.

`innersport` <*portnum*>
`isport` <*portnum*>

`innersport` specifies the source port number of the inner header for a tunnel-mode SA. It should be used in combination with an upper-layer protocol (see below), but it does not have to be.

`innerdport` <*portnum*>
`idport` <*portnum*>
`innerdport` specifies the destination port number of the inner header for a tunnel-mode SA. It should be used in combination with an upper-layer protocol (see below), but it does not have to be.

`iproto` <*protocol number*>`iulp` <*protocol number*>
`iproto`, and its synonym `iulp`, specify the IP protocol number of the inner header of a tunnel-mode SA.

`authkey` <*hexstring*>
Specifies the authentication key for this SA. The key is expressed as a string of hexadecimal digits, with an optional `/` at the end, for example, `123/12`. Bits are counted from the most-significant bits down. For example, to express three '1' bits, the proper syntax is the string "`e/3`". For multi-key algorithms, the string is the concatenation of the multiple keys. This extension is used by the `add` and `update` commands.

`encrkey` <*hexstring*>
Specifies the encryption key for this SA. The syntax of the key is the same as `authkey`. A concrete example of a multi-key encryption algorithm is `3des`, which would express itself as a 192-bit key, which is three 64-bit parity-included `DES` keys. This extension is used by the `add` and `update` commands.

Certificate identities are very useful in the context of automated key management, as they tie the `SA` to the public key certificates used in most automated key management protocols. They are less useful for manually added SAs. Unlike other extensions, `srcidtype` takes two values, a *type*, and an actual *value*. The type can be one of the following:

`prefix`
An address prefix.

`fqdn`
A fully-qualified domain name.

`domain`
Domain name, synonym for `fqdn`.

`user_fqdn`
User identity of the form *user*@*fqdn*.

`mailbox`
Synonym for `user_fqdn`.

The *value* is an arbitrary text string that should identify the certificate.

`srcidtype` <*type, value*>
Specifies a source certificate identity for this SA. This extension is used by the `add` and `update` commands.

`dstidtype` <*type, value*>
Specifies a destination certificate identity for this SA. This extension is used by the `add` and `update` commands

Tunnel Mode versus Transport Mode SAs

An IPsec SA is a Tunnel Mode SA if the "proto" value is either 4 (`ipip`) or 41 (`ipv6`) *and* there is an inner-address or inner-port value specified. Otherwise, the SA is a Transport Mode SA.

Security

Keying material is very sensitive and should be generated as randomly as possible. Some algorithms have known weak keys. IPsec algorithms have built-in weak key checks, so that if a weak key is in a newly added `SA`, the `add` command will fail with an invalid value.

The `ipseckey` command allows a privileged user to enter cryptographic keying information. If an adversary gains access to such information, the security of IPsec traffic is compromised. The following issues should be taken into account when using the `ipseckey` command.

1. Is the TTY going over a network (interactive mode)?
   - If it is, then the security of the keying material is the security of the network path for this TTY's traffic. Using `ipseckey` over a clear-text `telnet` or `rlogin` session is risky.
   - Even local windows might be vulnerable to attacks where a concealed program that reads window events is present.
2. Is the file accessed over the network or readable to the world (`-f` option)?
   - A network-mounted file can be sniffed by an adversary as it is being read.
   - A world-readable file with keying material in it is also risky.
3. The `ipseckey` command is designed to be managed by the `manual-key` `smf(5)` service. Because the `smf(5)` log files are world-readable, the `ipseckey` does not record any syntax errors in the log files, as these errors might include secret information.

   If a syntax error is found when the `manual-key` `smf(5)` service is enabled, the service enters maintenance mode. The log file will indicate that there was a syntax error, but will not specify what the error was.

   The administrator should use `ipeckey -c` *filename* from the command line to discover the cause of the errors. See `OPTIONS`.

If your source address is a host that can be looked up over the network and your naming system itself is compromised, then any names used will not be trustworthy.

Security weaknesses often lie in misapplication of tools, not in the tools themselves. Administrators are urged to be cautious when using `ipseckey`. The safest mode of operation is probably on a console or other hard-connected TTY.

For further thoughts on this subject, see the afterward by Matt Blaze in Bruce Schneier's *Applied Cryptography: Protocols, Algorithms, and Source Code in C*.

Service Management Facility

IPsec manual keys are managed by the service management facility, smf(5). The services listed below manage the components of IPsec. These services are delivered as follows:

```
svc:/network/ipsec/policy:default (enabled)
svc:/network/ipsec/ipsecalgs:default (enabled)
svc:/network/ipsec/manual-key:default (disabled)
svc:/network/ipsec/ike:default (disabled)
```

The manual-key service is delivered disabled. The system administrator must create manual IPsec Security Associations (SAs), as described in this man page, before enabling that service.

The policy service is delivered enabled, but without a configuration file, so that, as a starting condition, packets are not protected by IPsec. After you create the configuration file `/etc/inet/ipsecinit.conf` and refresh the service (`svcadm refresh`, see below), the policy contained in the configuration file is applied. If there is an error in this file, the service enters maintenance mode. See ipsecconf(1M).

Services that are delivered disabled are delivered that way because the system administrator must create configuration files for those services before enabling them. See ike.config(4) for the `ike` service.

See ipsecalgs(1M) for the `ipsecalgs` service.

The correct administrative procedure is to create the configuration file for each service, then enable each service using svcadm(1M).

If the configuration needs to be changed, edit the configuration file then refresh the service, as follows:

```
example# svcadm refresh manual-key
```

Warning: To prevent `ipseckey` complaining about duplicate Associations, the `ipseckey` command flushes the Security Association Data Base (SADB) when the `ipseckey` command is run from smf(5), before adding any new Security Associations defined in the configuration file. This differs from the command line behavior where the SADB is not flushed before adding new Security Associations.

The smf(5) framework will record any errors in the service-specific log file. Use any of the following commands to examine the `logfile` property:

```
example# svcs -l manual-key
example# svcprop manual-key
example# svccfg -s manual-key listprop
```

The following property is defined for the `manual-key` service:

```
config/config_file
```

This property can be modified using svccfg(1M) by users who have been assigned the following authorization:

```
solaris.smf.value.ipsec
```

See auths(1), user_attr(4), rbac(5).

The service needs to be refreshed using svcadm(1M) before the new property is effective. General non-modifiable properties can be viewed with the svcprop(1) command.

```
# svccfg -s ipsec/manual-key setprop config/config_file = \
/new/config_file
# svcadm refresh manual-key
```

Administrative actions on this service, such as enabling, disabling, refreshing, and requesting restart can be performed using svcadm(1M). A user who has been assigned the authorization shown below can perform these actions:

```
solaris.smf.manage.ipsec
```

The service's status can be queried using the svcs(1) command.

The ipseckey command is designed to be run under smf(5) management. While the ipsecconf command can be run from the command line, this is discouraged. If the ipseckey command is to be run from the command line, the manual-key smf(5) service should be disabled first. See svcadm(1M).

Examples

EXAMPLE 1 Emptying Out All SAs

To empty out all SA:

```
example# ipseckey flush
```

EXAMPLE 2 Flushing Out IPsec AH SAs Only

To flush out only IPsec AH SAs:

```
example# ipseckey flush ah
```

EXAMPLE 3 Saving All SAs To Standard Output

To save all SAs to the standard output:

```
example# ipseckey save all
```

EXAMPLE 4 Saving ESP SAs To The File /tmp/snapshot

To save ESP SAs to the file /tmp/snapshot:

```
example# ipseckey save esp /tmp/snapshot
```

EXAMPLE 5 Deleting an IPsec SA

To delete an IPsec SA, only the SPI and the destination address are needed:

```
example# ipseckey delete esp spi 0x2112 dst 224.0.0.1
```

An alternative would be to delete the SA and the SAs pair if it has one:

EXAMPLE 5 Deleting an IPsec SA *(Continued)*

```
example# ipseckey delete-pair esp spi 0x2112 dst 224.0.0.1
```

EXAMPLE 6 Getting Information on an IPsec SA

Likewise, getting information on a SA only requires the destination address and `SPI`:

```
example# ipseckey get ah spi 0x5150 dst mypeer
```

EXAMPLE 7 Adding or Updating IPsec SAs

Adding or updating SAs requires entering interactive mode:

```
example# ipseckey
ipseckey> add ah spi 0x90125 src me.domain.com dst you.domain.com \
          authalg md5 authkey 1234567890abcdef1234567890abcdef
ipseckey> update ah spi 0x90125 dst you.domain.com hard_bytes \
          16000000
ipseckey> exit
```

Adding two SAs that are linked together as a pair:

```
example# ipseckey
ipseckey> add esp spi 0x2345 src me.domain.com dst you.domain.com \
   authalg md5 authkey bde359723576fdea08e56cbe876e24ad \
   encralg des encrkey be02938e7def2839
ipseckey> add esp spi 0x5432 src me.domain.com dst you.domain.com \
   authalg md5 authkey bde359723576fdea08e56cbe876e24ad \
   encralg des encrkey be02938e7def2839 pair-spi 0x2345
ipseckey> exit
```

EXAMPLE 8 Adding an SA in the Opposite Direction

In the case of IPsec, SAs are unidirectional. To communicate securely, a second SA needs to be added in the opposite direction. The peer machine also needs to add both SAs.

```
example# ipseckey
ipseckey> add ah spi 0x2112 src you.domain.com dst me.domain.com \
          authalg md5 authkey bde359723576fdea08e56cbe876e24ad \
          hard_bytes 16000000
ipseckey> exit
```

EXAMPLE 9 Monitoring `PF_KEY` Messages

Monitoring for `PF_KEY` messages is straightforward:

```
example# ipseckey monitor
```

EXAMPLE 10 Using Commands in a File

Commands can be placed in a file that can be parsed with the `-f` option. This file may contain comment lines that begin with the "#" symbol. For example:

EXAMPLE 10 Using Commands in a File *(Continued)*

```
# This is a sample file for flushing out the ESP table and
# adding a pair of SAs.

flush esp

### Watch out!  I have keying material in this file.  See the
### SECURITY section in this manual page for why this can be
### dangerous .

add esp spi 0x2112 src me.domain.com dst you.domain.com \
    authalg md5 authkey bde359723576fdea08e56cbe876e24ad \
    encralg des encrkey be02938e7def2839 hard_usetime 28800
add esp spi 0x5150 src you.domain.com dst me.domain.com \
    authalg md5 authkey 930987dbe09743ade09d92b4097d9e93 \
    encralg des encrkey 8bd4a52e10127deb hard_usetime 28800

## End of file  -  This is a gratuitous comment
```

EXAMPLE 11 Adding SAs for IPv6 Addresses

The following commands from the interactive-mode create an SA to protect IPv6 traffic between the site-local addresses

```
example # ipseckey
ipseckey> add esp spi 0x6789 src6 fec0:bbbb::4483 dst6 fec0:bbbb::7843\
           authalg md5 authkey bde359723576fdea08e56cbe876e24ad \
          encralg des encrkey be02938e7def2839 hard_usetime 28800
ipseckey>exit
```

EXAMPLE 12 Linking Two SAs as a Pair

The following command links two SAs together, as a pair:

```
example# ipseckey update esp spi 0x123456 dst 192.168.99.2 \
pair-spi 0x654321
```

Files

`/etc/inet/secret/ipseckeys`
Default configuration file used at boot time. See "Service Management Facility" and `SECURITY` for more information.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed

See Also ps(1), svcprop(1), svcs(1), ipsecconf(1M), ipsecalgs(1M), route(1M), svcadm(1M), svccfg(1M), ike.config(4), attributes(5), smf(5), ipsec(7P), ipsecah(7P), ipsecesp(7P), pf_key(7P)

Schneier, B., *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Second ed. New York, New York: John Wiley & Sons, 1996.

Diagnostics The `ipseckey` command parses the configuration file and reports any errors. In the case of multiple errors, `ipseckey` reports as many of these as possible.

The `ipseckey` command does not attempt to use a `COMMAND` that has a syntax error. A `COMMAND` might be syntactically correct but can nevertheless generate an error because the kernel rejected the request made to `pf_key(7P)`. This might occur because a key had an invalid length or because an unsupported algorithm was specified.

If there are any errors in the configuration file, ipseckey reports the number of valid COMMANDS and the total number of COMMANDS parsed.

`Parse error on line` *N*`.`
: If an interactive use of `ipseckey` would print usage information, this would print instead. Usually proceeded by another diagnostic. Because `COMMANDS` can cover more than a single line in the configuration file by using the backslash character to delimit lines, its not always possible to pinpoint in the configuration file the exact line that caused the error.

`Unexpected end of command line.`
: An additional argument was expected on the command line.

Unknown
: A value for a specific extension was unknown.

`Address type` *N* `not supported.`
: A name-to-address lookup returned an unsupported address family.

N `is not a bit specifier`
`bit length` *N* `is too big for`
`string is not a hex string`
: Keying material was not entered appropriately.

`Can only specify single`
: A duplicate extension was entered.

`Don't use extension for <`*string*`> for <`*command*`>.`
: An extension not used by a command was used.

`One of the entered values is incorrect: Diagnostic code` *NN*`: <`*msg*`>`
: This is a general invalid parameter error. The diagnostic code and message provides more detail about what precise value was incorrect and why.

Notes In spite of its IPsec-specific name, `ipseckey` is analogous to `route`(1M), in that it is a command-line interface to a socket-based administration engine, in this case, `PF_KEY`. `PF_KEY` was originally developed at the United States Naval Research Laboratory.

To have machines communicate securely with manual keying, SAs need to be added by all communicating parties. If two nodes wish to communicate securely, both nodes need the appropriate SAs added.

In the future `ipseckey` may be invoked under additional names as other security protocols become available to `PF_KEY`.

This command requires `sys_ip_config` privilege to operate and thus can run in the global zone and in exclusive-IP zones. The global zone can set up security associations with `ipseckey` to protect traffic for shared-IP zones on the system.

Name iscsiadm – enable management of iSCSI initiators

Synopsis `iscsiadm` *subcommand direct-object* [*options*] [*operand*]

Description The `iscsiadm` command enables management of the iSCSI (Internet SCSI) initiator on a host. `iscsiadm` is implemented as a set of subcommands, many with their own options, which are described in the section for that subcommand. Options not associated with a particular subcommand are described under OPTIONS.

`iscsiadm` works only when the following service is online:

```
svc:/network/iscsi/initiator:default
```

The `iscsiadm` command supports the following subcommands, which are described in detail in subsections that follow:

`add` Adds element(s) to an object.

`list` Lists element(s) of an object.

`modify` Modifies attributes of an object.

`remove` Removes an element from an object.

The `iscsiadm` subcommands operate on a *direct-object*. These are described in the section for each subcommand.

The `iscsiadm` command supports the Internet Storage Name Service (iSNS) for the discovery of iSCSI targets. The command supports the Challenge Handshake Authentication Protocol (CHAP) for authentication.

`add` Subcommand The syntax for the `add` subcommand is:

```
# iscsiadm add direct_object [operands...]
```

The `add` subcommand adds the following *direct_object*s:

`discovery-address` *discovery-address* [...]
Adds a target to a list of discovery addresses. A discovery address (as in the syntax shown below) is an IP *address*:*port* combination used in a SendTargets discovery session. Using this discovery approach, a target device can inform an initiator of the target address and target name of each target exposed by that device. Connection to a target is not attempted unless the SendTargets method of discovery has been enabled on the host. You enable this method with the `modify` subcommand.

The *discovery-address* parameter is formatted as:

<*IP address*>[:*port*]

If *port* is not specified, the default of 3260 will be used.

`isns-server` *isns-server* [...]
Add an iSNS server to the list of iSNS server addresses. An iSNS server address (specified in the syntax shown below) is an IP address-port combination used in an iSNS discovery session. By using iSNS discovery, an iSNS server can provide an initiator with information about a portal and the name of each target that belongs to the same discovery domain as that of the initiator. Connection to the iSNS server is not attempted unless the iSNS method of discovery has been enabled on the host. You enable this method with the `modify` subcommand, described below.

The *isns-server* parameter is formatted as:

IP_address`[:`*port*`]`

If a port is not specified, the default of 3205 is used.

`static-config` *static_target* [...]
Adds a target to the list of statically configured targets. A connection to the target will not be attempted unless the static configuration method of discovery has been enabled.

The *static_target* parameter is formatted as:

<*target-name*>`,`<*target address*>`[:`*port-number*`][,`*tpgt*`]`

<*target-name*> can be up to 223 characters.

`list` Subcommand The syntax for the `list` subcommand is:

```
# iscsiadm list direct-object [options]
```

The `list` subcommand displays data for the following *direct-object*s:

`discovery`
Lists the discovery methods and their current activation state, enabled or disabled. Discovery methods are:

- iSNS (Internet Storage Name Service)
- Static
- SendTargets

`initiator-node`
Lists information for the initiator node on the host. The iSCSI initiator node represents a logical HBA and is a logical host connection point for iSCSI targets. The parameter values listed in the response are default parameter settings for the initiator. Each connected target for an initiator can have parameter values that differ from the parameter values on the initiator node.

`static-config` [*static_target*[, ...]]
Lists the target name and address for specified targets or, if no static targets are specified, all statically discovered targets.

`target` [`-S`] [`-v`] [*target*[, ...]]
Lists a target's current parameters, connection state, and which method was used for the target's discovery. Reports information for specified targets or, if no targets are specified, all targets that have been discovered or have had parameters modified by the `modify target` subcommand.

When used with the `-S` option for a specified target, this subcommand returns:

- target name
- logical unit number
- vendor ID
- product ID
- OS device name (for example, `/dev/rdsk/c0t2d0s0`)

The `-v` options gives more details, such as the current login parameters, the detailed connection information, and the discovery method used to discover the target.

A return of `NA` as the discovery method parameter indicates that the target was created with a `iscsiadm modify target-param` command and does not exist as a discovered object. To remove such targets, use `iscsiadm remove target-param`.

`target-param` [`-v`] *target* [...]
Lists a target's default and user-defined parameters.

`discovery-address` [`-v`] [*discovery-address*[, ...]]
Lists the `discovery-address` objects that have been added using the `iscsiadm add discovery-address` subcommand.

When used with the `-v` option, lists all known targets at a specified *discovery-address*. The `-v` option returns one or more target names along with zero or more target addresses and associated target portal group tags (TPGT), if applicable.

`isns-server` [`-v`] [*isns-server*[, ...]]
Lists the `isns-server` objects that have been added using the `iscsiadm add isns-server` subcommand.

When used with the `-v` option, this subcommand lists all known targets at a specified `isns-server` address. The `-v` option returns one of more target names along with zero or more target addresses and associated target portal group tags, if applicable.

`modify` Subcommand

The syntax for the `modify` subcommand is:

```
# iscsiadm modify direct_object [options]
```

The `modify` subcommand supports the following *direct_object*s:

`discovery` [*options*]
Enabling a discovery method initiates a discovery using that method. Disabling a discovery method that is currently enabled does not affect connections to any targets that have already been discovered by that method.

Options for `modify discovery` are as follows:

`-i`, `-iSNS enable` | `disable`
Enable or disable iSNS discovery.

`-s`, `--static enable` | `disable`
Enable or disable static discovery.

`-t`, `--sendtargets enable` | `disable`
Enable or disable SendTargets discovery.

`initiator-node` [*options*]
Modifies an initiator's properties. If a target is currently connected, this operation can succeed. However, the modified set of parameters will not be in effect for that target until an existing connection session no longer exists and a new connection has been established. The options `-C` and `--CHAP-secret` require a CHAP secret entry in response to a prompt.

Options for `modify initiator-node` are as follows:

`-A`, `--node-alias` <*initiator node alias*>
Modifies the initiator node alias. Maximum length of 223 characters.

`-a`, `--authentication chap` | `none`
Sets the authentication mode.

`-C`, `--CHAP-secret`
Sets the CHAP secret value. There is no default value. Maximum length is 16 characters.

`-c`, `--configured-sessions` <*num_sessions*> | <*IP address*>[,<*IP address*>...]
Sets the number of configured iSCSI sessions that will be created for each iSCSI target. The feature should be used in combination with the Solaris I/O multipathing feature described in `scsi_vhci(7D)`.

`-d`, `--datadigest none` | `CRC32`
Sets whether CRC32 is enabled to check SCSI data transfers.

`-H`, `--CHAP-name` *CHAP name*
Specifies a CHAP username. If you do not use this option, upon initialization, the CHAP name is set to the initiator node name. When the authentication method is set to CHAP (see `-a`/`--authentication` option, above), the CHAP username is displayed with the command `iscsiadm list initiator-node`.

`-h`, `--headerdigest none` | `CRC32`
Sets whether CRC32 is enabled to check SCSI packet headers.

`-N`, `--node-name` <*initiator node name*>
Modifies the initiator node name. Maximum of 223 characters.

Note – During Solaris installation, the initiator node name is set to a globally unique value. Changing this value can adversely affect operation within the iSCSI network.

`-P`, `--radius-shared-secret` (exclusive)
Sets the RADIUS shared secret.

`-R`, `--radius-access` `enable` | `disable`
Sets whether a RADIUS server will be used.

`-r`, `--radius-server` <*IP address*>[:<*port*>]
Sets the IP address and port of the radius server to be used.

`-T`, `--tunable-param` <<*tunable-prop*>=<*value*>, ...>
Specify one or more tunable parameters for all targets that initiator node connected.

Note – These values should only be modified by an administrator with a good working knowledge of the parameter's impact within the iSCSI network.

Supported tunable-prop options are:

`recv-login-rsp-timeout`
Session Login Response Time

The `recv-login-rsp-timeout` option specifies how long iSCSI initiator will wait for the response of iSCSI session login request from the iSCSI target. Valid value is from 0 to 60*60, default to 60 seconds.

`conn-login-max`
Maximized Connection Retry Time

The `conn-login-max` option lets the iSCSI initiator reestablish the connection to the target in case of IO timeout or connection failure during the given time window. Valid value is from 0 to 60*60, default to 180 seconds.

`polling-login-delay`
Login Retry Time Interval

The `polling-login-delay` option specifies the time interval between each login retry when iSCSI initiator to target IO timeout or connection failure. Valid value is from 0 to 60*60, default to 60 seconds.

`target-param` [*options*] *target*
Modifies a target's parameters. If a target is currently connected, the modify operation will succeed, although the modified settings might not take effect for a few seconds. To confirm that these settings are active, use `iscsiadm list target -v`. If a specified target is not associated with any discovery method, a target object is created with the specified parameters. After using this command to modify a target's parameters, the new parameters will persist until they are modified or removed with a `iscsiadm remove target-param` command on that target. The options `-C` and `--CHAP-secret` require a CHAP secret entry in response to a prompt.

Options for `modify target-param` are as follows:

`-B`, `--bi-directional-authentication enable | disable`
Sets the bidirectional option. If set to `enable`, the initiator performs bidirectional authentication for the specified target.

`-C`, `--CHAP-secret`
Sets the target's CHAP secret value. There is no default value. Maximum acceptable length is 16 characters.

`-c`, `--configured-sessions` <*num_sessions*> | <*IP address*>[,<*IP address*>...]
Sets the number of configured iSCSI sessions that will be created for each iSCSI target. The feature should be used in combination with the Solaris I/O multipathing feature described in `scsi_vhci(7D)`.

`-d`, `--datadigest none | CRC32`
Sets whether CRC32 is enabled or disabled for the data.

`-H`, `--CHAP-name` *CHAP name*
Sets a CHAP username. If you do not use this option, upon initialization, the CHAP name is set to the target name. When the authentication method is set to CHAP (see `-a`/`--authentication` option, under the `initiator-node` direct object, above), the CHAP username is displayed with the command `iscsiadm list initiator-node`.

`-h`, `--headerdigest none | CRC32`
Sets whether CRC32 is enabled or disabled for the header.

`-p`, `--login-param`
Specify one or more login parameter settings.

Note – These values should only be modified by an administrator with a good working knowledge of the parameter's impact within the iSCSI network.

The login parameters are derived from iSCSI proposed standard RFC 3720. Valid values are:

`dataseqinorder`	`yes` or `no`
`defaulttime2retain`	0–3600
`defaulttime2wait`	0–3600
`firstburstlength`	512 to 2^{24-1}
`immediatedata`	`yes` or `no`
`initialr2t`	`yes` or `no`
`maxburstlength`	512 to 2^{24-1}
`datapduinorder`	`yes` or `no`
`maxoutstandingr2t`	1 to 65535
`maxrecvdataseglen`	512 to 2^{24-1}

`-T`, `--tunable-param` <<*tunable-prop*>=<*value*>, ...>
Specify one or more tunable parameters for all targets that initiator node connected.

Note – Tunable values should only be modified by an administrator with a good working knowledge of the parameter's impact within the iSCSI network.

Supported *tunable-prop* options are:

`recv-login-rsp-timeout`
Session Login Response Time

The `recv-login-rsp-timeout` option specifies how long iSCSI initiator will wait for the response of iSCSI session login request from the iSCSI target. Valid value is from 0 to 60*60, default to 60 seconds.

`conn-login-max`
Maximized Connection Retry Time

The `conn-login-max` option lets the iSCSI initiator reestablish the connection to the target in case of IO timeout or connection failure during the given time window. Valid value is from 0 to 60*60, default to 180 seconds.

`polling-login-delay`
Login Retry Time Interval

The `polling-login-delay` option specifies the time interval between each login retry when iSCSI initiator to target IO timeout or connection failure. Valid value is from 0 to 60*60, default to 60 seconds.

`remove` Subcommand

The syntax for the `remove` subcommand is:

```
# iscsiadm remove direct_object
```

The `remove` subcommand supports the following *direct_object*s:

`discovery-address` *discovery-address*, ...
Removes a target device from the list of discovery addresses. A discovery address (as in the syntax shown below) is an IP address-port combination used in a SendTargets discovery session. Using this discovery approach, a target device can inform an initiator of the target address and target name of each target exposed by that device. If any target exposed by the discovery address is currently mounted or there is active I/O on the device, an error of "logical unit in use" is returned and the operation fails. If the associated devices are not in use, they are removed.

discovery-address must be formatted as:

<*IP address*>`[`:<*port*>`]`

There are no options associated with this direct object.

`isns-server` *isns-server*, ...
Removes an iSNS server from the list of iSNS server addresses. An iSNS server address (specified in the syntax shown below) is an IP address-port combination used in an iSNS discovery session. By using iSNS discovery, an iSNS server can provide an initiator with information about a portal and the name of each target that belongs to the same discovery domain as that of the initiator. If any target discovered by means of iSNS is currently mounted or there is active I/O on the device, an error of "logical unit in use" is returned and the operation fails. If the associated devices are not in use, they are removed.

isns-server must be formatted as:

IP_address`[:`*port*`]`

There are no options associated with this direct object.

`static-config` *static_target*, ...
Removes a target from the list of statically discovered targets. If the target being removed is currently mounted or there is active I/O on the device, an error of "logical unit in use" is returned and the operation fails. If a device is not in use, it will be removed.

static_target must be formatted as:

<*target-name*>`,`<*target-address*>`[:`*port-number*`][,`*tpgt*`]`

There are no options associated with this direct object.

`target-param` *target-name*
Removes target specified by *target-name*. The target name is formatted as:

<*target-name*>

There are no options associated with this direct object.

Proper Use of Discovery Methods

Do not configure a target to be discovered by both static and dynamic discovery methods. The consequence of using redundant discovery methods might be slow performance when communicating with the iSCSI target device.

Options

The following generic options are supported:

`-V`, `--version` Displays version information. Stops interpretation of subsequent arguments.

`-?`, `--help` Displays help information. Can be used following an `iscsiadm` command with no arguments, following a subcommand, or following a subcommand-direct object combination. Responds with help information appropriate for your entry. For example, if you enter:

```
# iscsiadm modify initiator-node --help
```

...`iscsiadm` responds with a display of the options available for that combination of subcommand and direct object.

Examples

EXAMPLE 1 Adding a Discovery Address

The following command uses the add subcommand to add a discovery address.

```
# iscsiadm add discovery-address 10.0.0.1:3260 10.0.0.2:3260
```

EXAMPLE 2 Adding a Static Target

The following command uses the add subcommand to add a static target.

```
# iscsiadm add static-config \
iqn.1999-08.com.array:sn.01234567,10.0.0.1:3260
```

EXAMPLE 3 Listing Current Discovery Settings

The following command uses the `list` subcommand to list current discovery settings.

```
# iscsiadm list discovery
        Discovery:
                Static: enabled
                Send Targets: disabled
                iSNS: enabled
```

EXAMPLE 4 Obtaining Verbose Discovery Output

The following commands uses the `-v` option (one with, one without) with the `list` subcommand to obtain verbose output.

```
# iscsiadm list discovery-address
        Discovery Address: 10.0.0.1:3260
        Discovery Address: 10.0.0.2:3260

# iscsiadm list discovery-address -v 10.0.0.1:3260
        Discovery Address: 10.0.0.1:3260
                Target name: eui.210000203787d1f7
                        Target address:    10.0.0.1:3260
                Target name: eui.210000203787a693
                        Target address:    10.0.0.1:3260
```

EXAMPLE 5 Displaying Information on the Initiator

The following command uses the `list` subcommand to display information on the initiator.

```
# iscsiadm list initiator-node
Initiator node name: iqn.1986-03.com.company.central.interopv20-1
Initiator node alias: interopv20-1
        Login Parameters (Default/Configured):
                Header Digest: NONE/NONE
                Data Digest: NONE/NONE
        Authentication Type: CHAP
                CHAP Name: iqn.1986-03.com.company.central.interopv20-1
        RADIUS Server: NONE
```

EXAMPLE 5 Displaying Information on the Initiator *(Continued)*

```
        RADIUS access: disabled
        Tunable Parameters (Default/Configured):
                Session Login Response Time: 60/-
                Maximum Connection Retry Time: 180/-
                Login Retry Time Interval: 60/-
        Configured Sessions: 1
```

EXAMPLE 6 Displaying Static Configuration Information

The following command uses the `list` subcommand to display information about static configurations.

```
# iscsiadm list static-config
        Static target: eui.210000203787a693,10.0.0.1:3260
```

EXAMPLE 7 Displaying Target Information

The following commands show the use of the `list` subcommand with various options to display information about targets.

```
# iscsiadm list target
Target: iqn.2004-05.com.abcStorage:Tgt-1
        Alias: -
        TPGT: 12288
        ISID: 4000002a0000
        Connections: 1# iscsiadm list target -v iqn.2004-05.com.abcStorage:Tgt-1
Target: iqn.2004-05.com.abcStorage:Tgt-1
        Alias: -
        TPGT: 12288
        ISID: 4000002a0000
        Connections: 1
                CID: 0
                  IP address (Local): 10.4.52.158:32803
                  IP address (Peer): 10.4.49.70:3260
                  Discovery Method: SendTargets
                  Login Parameters (Negotiated):
                        Data Sequence In Order: yes
                        Data PDU In Order: yes
                        Default Time To Retain: 20
                        Default Time To Wait: 2
                        Error Recovery Level: 0
                        First Burst Length: 65536
                        Immediate Data: yes
                        Initial Ready To Transfer (R2T): yes
                        Max Burst Length: 262144
                        Max Outstanding R2T: 1
                        Max Receive Data Segment Length: 65536
```

EXAMPLE 7 Displaying Target Information *(Continued)*

```
                        Max Connections: 1
                        Header Digest: NONE
                        Data Digest: NONE
# iscsiadm list target -S iqn.2004-05.com.abcStorage:Tgt-1
Target: iqn.2004-05.com.abcStorage:Tgt-1
        Alias: -
        TPGT: 12288
        ISID: 4000002a0000
        Connections: 1
        LUN: 6
             Vendor:  ABCStorage
             Product: iSCSI Target
             OS Device Name: /dev/rdsk/c3t1d0s2
        LUN: 5
             Vendor:  ABCStorage
             Product: iSCSI Target
             OS Device Name: /dev/rdsk/c3t0d0s2
```

EXAMPLE 8 Displaying Target Parameter Information

The following command uses the `list` subcommand to display target information for a specific target.

```
# iscsiadm list target-param -v iqn.2004-05.com.abcStorage:Tgt-1
Target: iqn.2004-05.com.abcStorage:Tgt-1
        Alias: -
        Bi-directional Authentication: disabled
        Authentication Type: NONE
        Login Parameters (Default/Configured):
                Data Sequence In Order: yes/-
                Data PDU In Order: yes/-
                Default Time To Retain: 20/-
                Default Time To Wait: 2/-
                Error Recovery Level: 0/-
                First Burst Length: 65536/-
                Immediate Data: yes/-
                Initial Ready To Transfer (R2T): yes/-
                Max Burst Length: 262144/-
                Max Outstanding R2T: 1/-
                Max Receive Data Segment Length: 65536/-
                Max Connections: 1/-
                Header Digest: NONE/-
                Data Digest: NONE/-
        Tunable Parameters (Default/Configured):
                Session Login Response Time: 60/-
                Maximum Connection Retry Time: 180/-
```

EXAMPLE 8 Displaying Target Parameter Information *(Continued)*

```
                Login Retry Time Interval: 60/-
        Configured Sessions: 1
```

EXAMPLE 9 Enabling Static Discovery Method

The following command uses the `modify` subcommand to enable the static discovery method.

```
# iscsiadm modify discovery --static enable
```

EXAMPLE 10 Setting the IP Address for the Radius Server

The following command uses the `modify` subcommand to set the IP address for the radius server, which will be used for CHAP authentication.

```
# iscsiadm modify initiator --radius-server 10.0.0.1
```

EXAMPLE 11 Setting the Node Name for Initiator

The following command uses the `modify` subcommand to set the node name for the initiator node.

```
# iscsiadm modify initiator-node -N iqn.2004-10.com.SUN.host-1
```

EXAMPLE 12 Changing Target Parameters

The following command uses the `modify` subcommand to change the target parameters for a specified target.

```
# iscsiadm modify target-param -d none -h none eui.210000203787a693
```

EXAMPLE 13 Removing a Discovery Address

The following command uses the `remove` subcommand to remove a discovery address.

```
# iscsiadm remove discovery-address 10.0.0.1:3260
```

EXAMPLE 14 Removing Target Parameters

The following command uses the `remove` subcommand to remove a set of target parameters.

```
# iscsiadm remove target-param eui.210000203787a693
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/iscsi/initiator
Interface Stability	Committed

See Also attributes(5), iscsi(7D), scsi_vhci(7D)

System Administration Guide: Devices and File Systems

Name iscsitadm – administer iSCSI targets

Synopsis

```
iscsitadm create [-? | --help] object [-? | --help]
 [options] operand

iscsitadm modify [-? | --help] object [-? | --help]
 [options] operand

iscsitadm delete [-? | --help] object [-? | --help]
 [options] operand

iscsitadm list [-? | --help] object [-? | --help] [options]
 operand

iscsitadm show [-? | --help] admin

iscsitadm show [-? | --help] object [-? | --help] [options]
 [operand]

iscsitadm -? --help
```

Description The `iscsitadm` command enables you to manage Internet SCSI (iSCSI) target nodes. It is a companion to `iscsiadm(1M)`, which enables you to manage iSCSI initiator nodes.

The `iscsitadm` command has the following subcommands:

`create`	Creates a target using a local target as a reference.
`modify`	Modifies a target or a list of targets.
`delete`	Deletes a target or a list of targets.
`list`	Lists names and information about targets.
`show`	Displays target-related statistics.

The preceding subcommands work on the following objects:

`target`	An iSCSI target node, or list of target nodes.
`initiator`	An iSCSI initiator node, or list of initiator nodes.
`admin`	Stores administrative information, such as server locations and passwords.
`tpgt`	Stands for `TargetPortGroupTag`. A number that represents a list of connections that an initiator can use for a given target.
`stats`	Displays statistics; can accept interval and count values. Used only with the `show` subcommand.

These objects are discussed in greater detail under the options descriptions for each subcommand.

As indicated in the SYNOPSIS, `iscsitadm` has two levels of help. If you invoke `-?` or `--help` following a subcommand, the command displays available operands, options, and objects. If you invoke an help option following an object, `iscsitadm` displays options and operands.

Options The `iscsitadm` options and objects are discussed below in the context of each subcommand. Note that the help options (`-?` or `--help`) are invoked as shown in the SYNOPSIS. See EXAMPLES.

`create` Options The following are the options and objects for the `create` subcommand:

`target --size|-z` *lun_size* [`--lun` *number*]
[`--type disk|tape|raw`] [`--backing-store|-b` *pathname*] *local_name*
Create a target using *local_name* as a reference. *local_name* is only used within the context `iscsitgtd`. `--size` is a multiplier and is specified as a number followed by a single letter. The letter is one of:

`k` kilobyte

`m` megabyte

`g` gigabyte

`t` terabyte

`--lun` specifies the logical unit number. `--type` specifies which type of emulation will occur for the LUN. `disk` and `tape` are the familiar devices. `raw` indicates that the emulator will use the uSCSI interface and pass the command blocks directly to and from the device. The use of `raw` also implies the option `--backing-store` will be entered. The argument to this option is the full pathname to the device node normally found in `/dev`. If you use `--backing-store`, the size of the store is determined by a SCSI `READ_CAPACITY` command or, if the backing store is a regular file, by `stat(2)`.

If *local_name* already exists, a new target name is not generated for this LUN. The LUN is created within the *local_name* storage hierarchy. You can use the `--backing-store` option to specify a different location for the data. If you use `--backing-store`, it is up to you to allocate actual storage instead of having the target create the data file.

`initiator --iqn|-n` *iSCSI_node_name local_initiator*
To use access control lists you must know the name of the initiator. Since the iSCSI initiator name can be quite long (223 bytes) and made up of a long list of numbers, it is best to enter this information once and then refer to the initiator using a simplified name of *local_initiator*.

`tpgt` *tpgt_number*
If a host has multiple NICs, you might want to limit the number of connections that an initiator can use for a given target. To establish this limit, you must first create a TargetPortGroupTag (TPGT), which can be any number from 1 to 65535. Once this tag is created, the IP addresses of the NICs can be added to the TPGT, using the `modify`

subcommand. Then, the TPGT can itself be added to the target.

`modify` Options The following are the options and objects for the `modify` subcommand:

`target --tpgt|-p` *local_tpgt local_target*	Specifies one or more target portal groups to use when initiators reference *local_target* during discovery.
`target --acl|-l` *local_initiator local_target*	Adds to the list a local initiator that can access *local_target*. By adding an initiator to a target all initiators from that point on must be in the ACL.
`target --alias|-a` *TargetAlias local_target*	Sets the alias if it was not done during the creation of the target or change an existing target's alias.
`target --maxrecv|-m` *value local_target*	Sets the `MaxRecvDataSegmentLength`, which can be any value between 512 to $(2^{24} - 1)$. You can use this option to limit the amount of memory used by the target.
`initiator --chap-secret|-C` *local_initiator*	Prompts the user to enter the value, using `getpassphrase(3C)`. Associates the secret used for CHAP authentication during login with *local_initiator*.
`initiator --chap-name|-H` *value local_initiator*	Specifies the CHAP username used during authentication.
`tpgt --ip-address|-i` *address tpgt_number*	Adds the NIC's *address* to *tpgt_number*.
`admin --base-directory|-d` *directory*	Sets the location of where to store the data files that represent the individual LUNs. This should be done before any other function is performed. Otherwise, an error will be generated when attempting to set a persistent value.
`admin --chap-secret|-C`	Upon entering this option, you will be prompted to enter the value using `getpassphrase(3C)`. For bidirectional authentication, this is the value used to generate a response to the initiator's challenge.
`admin --chap-name|-H` *value*	Specifies the user name portion of the CHAP protocol.

`admin --radius-access|-R enable | disable`	Enables or disables the use of the RADIUS server. Even with a RADIUS server address defined, the use of that server must be enabled. If the server becomes inaccessible and you need to fall back on configuration file access, you can use this option to disable the server.
`admin --radius-server|-r` *hostname*:*port*	Location of RADIUS server. *hostname* can be either a resolvable name or an IP address.
`admin --radius-secret|-P`	Used to initiate contact with the RADIUS server. Interaction with server uses `getpassphrase(3C)`.
`admin --isns-access|-S enable | disable`	Enables or disables access to an iSNS server. iSNS servers broadcast their locations.
`admin --isns-server|-s` *hostname*	Location of the iSNS server. "*hostname*" can be either a resolvable host name or an IP address.
`admin --fast-write-ack|-f enable | disable`	Enables or disables fast-write acknowledgment. You should enable this option only if a system is connected to the power grid through a UPS. Otherwise, data corruption could occur if power is lost and some writes that were acknowledged have not been completely flushed to the backing store.

`delete` Options — The following are the options and objects for the `delete` subcommand:

`target --lun|-u` *lun_number local_target*	Removes information about the LUN identified by *lun_number*. This includes the data that is stored in the LUNs.
`target --acl|-l` *local_initiator local_target*	Remove access to *local_target* by *local_initiator*. If the initiator is currently logged into the target, this option sends an asynchronous event message to the initiator.
`target --tpgt|-p` *local_tpgt local_target*	Removes the *local_tpgt* from *local_target*. Does not affect existing connections.

`initiator --all|-A` *local_initiator*
: Removes information about *local_initiator*. Does not affect current connections. This option search all targets, seeking those that reference *local_initiator*. On these, it performs the action defined by the command:

```
# iscsitadm delete target --acl local_initiator target
```

`tpgt --all|-A` *tpgt_number*
: Removes from the system all knowledge of the target portal group identified by *tpgt_number*. This includes removal of the references by targets to this group.

`tpgt --ip-address|-i` *address tpgt_number*
: Removes a NIC's address from the target portal group identified by *tpgt_number*. Does not affect current connections.

`list` Options
: The following are the options and objects for the `list` subcommand:

`target` [`--verbose`] [*local_target*]
`target` [`-v|-s` *num*] [*local_target*]
: By default, displays a list of target local names followed by the iSCSI TargetName, as it was created. By specifying *local_target*, the same information is displayed for that target and can be used to validate the name of *local_target*. With the `--verbose` option, information about the target's LUNs and current connections is displayed.

  You can use the `iostat(1M)` command to obtain information on the number of SCSI commands issued and sectors read and written.

`initiator` [`--verbose|-v`] *local_initiator*
: Displays detailed information about *local_initiator*. Among this data is CHAP information, what target portal groups this initiator belongs to, and any available connections.

`tpgt` [`--verbose|-v`] *tpgt_number*
: Displays detailed information about target group identified by *tpgt_number*. Among this data is the list of NICs that are a part of this target group.

`show` Options
: The following are the options and objects for the `show` subcommand:

`admin`
Displays a list of administrative information, including the base directory used by the target, CHAP, RADIUS, iSNS, and if fast writes are enabled.

`stats` [`--interval`|`-I` *seconds* [`--count`|`-N` *value*]] [*local_target*]
Displays statistics for all available targets, unless you specify *local_target*, in which case, displays statistics only for *local_target*. If you use `--interval`, displays statistics for an interval specified by *seconds*. If you do not specify `--count`, the display continues until you enter a Control-C.

Examples

EXAMPLE 1 Invoking Help

All of the commands shown below are valid ways of invoking help.

```
# iscsitadm -?
# iscsitadm modify -?
# iscsitadm modify target -?
# iscsitadm --help
# iscsitadm create --help
# iscsitadm create tpgt --help
```

EXAMPLE 2 Establishing Backing Store

The following command establishes the default location for the backing store. In addition to the backing store, certain configuration files will be stored in the same location.

```
# iscsitadm modify admin --base-directory /zfs/data/targets
```

The short form of the `--base-directory` option is `-d`.

EXAMPLE 3 Simplest-Case Target Creation

The following command creates a target that will emulate an LBA device that has 10 GB of storage available. With the base directory set up and as well as a single target, it is possible to use the system as an iSCSI target. Note that because the LUN is not specified on the command line, it reverts to the default, 0.

```
# iscsitadm create target --size 10g play_area
```

The short form of the `--size` option is `-z`.

EXAMPLE 4 Creating with Both Size and Backing Store

The following `iscsitadm create` command specifies LUN size and a backing store location. The result of this command is that the daemon will create a LUN file at the named location, of the specified size (20 GB).

```
# iscsitadm create target -z 20g -b /zfs/mirror/data/payroll payroll
```

A target such as the one created by the preceding command might be useful, for example, when most of the LUN can be created in a default area, using whatever redundancy is

EXAMPLE 4 Creating with Both Size and Backing Store *(Continued)*

provided by the underlying file system. Alternatively, you might want to create a special LUN on a higher speed storage medium or one with better failover characteristics.

The long form of the `-z` option is `--size`. The long form of the `-b` option is `--backing-store`

EXAMPLE 5 Specifying a Local Name for a SCSI Initiator

Consider that you want to restrict access to the `payroll` target, created in the previous example, to a limited set of initiators. Because the initiator names can be quite long (and therefore prone to be entered incorrectly), you create a local name for each initiator, as in the command below.

```
# iscsitadm create initiator --iqn \
iqn.1986-03.com.example[node name continues...] multistrada
```

The short form of the `--iqn` option is `-q`.

EXAMPLE 6 Granting an Initiator Access to a Target

Upon completion of the command below, only the initiator `multistrada` is allowed to log into the daemon and access the `payroll` target. This presents a potential gap in security, which is addressed in the following example.

```
# iscsitadm modify target --acl multistrada payroll
```

The short form of the `--acl` option is `-l`.

EXAMPLE 7 Adding CHAP Secret and Name for an Initiator

The initiator is allowed to identify itself. Because of this, it is prudent to add a CHAP secret an name for an initiator. This is accomplished with the following command.

```
# iscsitadm modify initiator -C multistrada
```

The preceding command prompts you for a secret to use. This must be the same secret that was setup on the initiator with the local name of `multistrada`. If it is not, the target daemon will issue a challenge to `multistrada` when it attempts to login. A non-matching response will cause the target to drop the connection. If you have many targets that require authentication, it is probably best to setup a RADIUS server to administer the secrets.

The long form of the `-C` option is `--chap-secret`.

EXAMPLE 8 Displaying Target Information

The following commands displays information about iSCSI targets.

```
# iscsitadm list target
Target: vol0
        iSCSI Name: iqn.1986-03.com.sun:01:00093d12170c.434c5250.vol0
```

EXAMPLE 8 Displaying Target Information *(Continued)*

```
Target: disk0
        iSCSI Name: iqn.1986-03.com.sun:01:00093d12170c.434c6f05.disk0
```

The following command differs from the preceding in that it uses the verbose (`-v`) option and it specifies a single target.

```
# iscsitadm list target -v vol0
Target: vol0
        iSCSI Name: iqn.1986-03.com.sun:01:00093d12170c.434c5250.vol0
        ACL list:
        TPGT list:
        LUN information:
                LUN: 0
                        GUID: 010000093d12170c00002a00434c5251
                        VID: SUN
                        PID: SOLARIS
                        Type: raw
                        Size: 0x1400000 blocks
```

EXAMPLE 9 Displaying Administrative Information

The following command uses the `show` subcommand to display administrative information.

```
# iscsitadm show admin
iscsitadm:
        Base Directory: /zfs/stress/play/targets
        CHAP Name: Not set
        RADIUS Access: Not set
        RADIUS Server: Not set
        iSNS Access: Not set
        Fast Write ACK: Not set
```

EXAMPLE 10 Displaying Statistics

The following command uses the `show` subcommand to display statistics.

```
# iscsitadm show stats
                        operations    bandwidth
device                 read  write   read  write
-------------------  -----  -----  -----  -----
vol0                     0      0     0K     0K
disk0                    0      0     0K     0K
```

Exit Status

`0` Command successful.

`>0` An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiscsitgtu
Interface Stability	Volatile

See Also iostat(1M), iscsiadm(1M), getpassphrase(3C), attributes(5), rbac(5), smf(5)

Notes This command set is considered to be experimental. Future releases, both minor and micro, might introduce incompatible changes to the command set. A future release will stabilize the command set. Any future changes in stability level will be reflected in the ATTRIBUTES section of this man page.

The iSCSI Target daemon, `iscsitgtd`, is managed by the service management facility (smf(5)), under the fault management resource identifier (FMRI):

```
svc:/system/iscsitgt:default
```

Use `iscsitadm` to perform administrative actions, such as are performed by the `create`, `modify`, and `delete` subcommands, on iSCSI Target properties. Such actions require that you become superuser or assume the Primary Administrator role. See rbac(5).

Name iscsitgtd – iSCSI Target daemon

Synopsis

```
iscsitgtd  [-c config_file] [-d door_file]
```

Description The `iscsitgtd` daemon process implements the iSCSI configuration, control, and data paths, providing iSCSI Target Mode support in the Solaris operating system.

The configuration and control path is by means of the Solaris Doors subsystem (see `door_create(3DOOR)`), and provides the interface between the iSCSI Target administration utility, `iscsitadm(1M)`, persistence configuration data stored in `/etc/iscsi/target_config.xml` and the operational state of the daemon process itself.

The data path, managed by the daemon, exists between TPC/IP port 3260, and the files, block devices, or raw SCSI devices configured as iSCSI target LUNs.

Options The following options are supported:

`-c` *config_file*
Override the location of the persistence configuration data from `/etc/iscsi/target_config.xml` to a file location of one's choosing.

`-d` *door_file*
Override the location of the Solaris Door used for configuration from `/var/run/iscsi_tgt_door` to a door of one's choosing.

Files `/etc/iscsi/target_config.xml`
Persistent configuration data.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWiscsitgtu
Interface Stability	Volatile

See Also `iscsitadm(1M)`, `door_create(3DOOR)`, `attributes(5)`, `smf(5)`

Notes The iSCSI Target daemon, `iscsitgtd`, is managed by the service management facility, `smf(5)`, under the fault management resource identifier:

```
svc:/system/iscsitgt:default
```

Name itu – convert packages to Driver Update format and patch Solaris install media for Install Time Update

Synopsis

```
itu makedu -r solaris_release [-v] [-f] [-d output_dir] [-o iso_file]
     [-l iso_label] package [package...]

itu updatemedia -R media_root [-v] [-f] [-o iso_file]
     [-l iso_label] pkg_or_patch [pkg_or_patch...]

itu makeiso -o iso_file [-v] [-l iso_label] media_root
```

Description The `itu` utility converts driver packages to Driver Update (DU) format and patches a Solaris install media with driver packages and patches for Install Time Update (ITU). `itu` has three subcommands: `makedu`, `updatemedia` and `makeiso`.

Options The following options are supported:

`-d` *output_dir*
Directory where the Driver Update directory is to be created.

`-f`
If *output_dir*/DU or *iso_file* already exists, remove it without asking first.

`-l` *iso_label*
Label/volume name of the ISO image (if `-o` option is specified).

`-o` *iso_file*
Path of the ISO image file to be created. For subcommands `updatemedia` and `makeiso`, it will be a bootable ISO image. This option must be specified for subcommand `makeiso`.

`-R` *media_root*
Top-level directory of on-disk image of Solaris installation media. This option must be specified for subcommand `updatemedia`.

`-r` *solaris_release*
Solaris release number for which the Driver Update is intended. It takes the form of the output of `uname -r`, for example, `5.10`. This option must be specified for subcommand `makedu`.

`-v`
Verbose. Multiple `-v` options increase verbosity.

Sub-commands The `itu` subcommands are described as follows.

`makedu` The `makedu` subcommand takes one or more driver packages as input and converts them to DU format. At the beginning of an interactive Solaris installation session, these driver updates can be applied to the running kernel, which will then also automatically apply them to the newly installed Solaris at the end of the installation process.

The `-r` option is required to specify the Solaris release number for which the driver updates apply. The *solaris_release* option argument takes the form `uname -r` output, for example, `5.10` or `5.11`.

If the `-d` option is specified, the resulting DU directory tree is placed in the directory *output_dir*.

If the `-o` option is specified, a (non-bootable) ISO image of the DU directory tree is written in the file *iso_file*. This ISO image can be burned onto a CD/DVD using `cdrw(1)` or `cdrecord(1)` (not a SunOS man page). See the "Examples" section below for an example of creating a DU on a floppy.

At least one of `-d` and `-o` option must be specified. If both are specified, then both an ISO image and a directory tree are generated.

`updatemedia` The `updatemedia` subcommand takes a list of driver packages and patches as input and applies them to the miniroot of a Solaris install media. It also places them in a subdirectory called `ITUs` under the Solaris install media's top-level directory:

media_root`/ITUs`

When booting a system from the updated media, the patches and packages will be part of the booted Solaris image. They will also be applied to the target system being installed at the end of the installation process.

The `-R` option must be entered on the command line to specify the Solaris install media. Note that the install media must be on a location that is writable by `itu`.

If the `-o` option is specified, a bootable ISO image of the patched install media is also created in the file *iso_file*. The ISO image can then be burned onto a CD or DVD.

`makeiso` The `makeiso` subcommand runs `mkisofs(8)` to create a bootable Solaris ISO image of the Solaris install media *media_root* and writes it to the file *iso_file*. The ISO image file can then be burned onto a CD or DVD with utilities such as `cdrw(1)` or `cdrecord(1)`. (Note that `mkisofs(8)` and `cdrecord(1)` are not SunOS man pages.)

Caution – The Solaris install media *media-root* must contain the file `boot/grub/stage2_eltorito`, which will be written to the media boot sectors. This file will be modified with some boot information, thus it has to be writable. If necessary, first save a copy, prior to running this subcommand.

Operands

The following operands are supported:

package [*package*...]
: One or more driver packages.

pkg_or_patch [*pkg_or_patch*...]
: One or more patches or packages.

media_root
: The top-level directory of a Solaris install media.

Examples

EXAMPLE 1 Creating a DU CD/DVD

The following commands create a Driver Update CD/DVD containing the packages `SAMPLEpkg1` and `SAMPLEpkg2`.

```
# itu makedu -r 5.10 -o my.iso SAMPLEpkg1 SAMPLEpkg2
# cdrw -i my.iso
```

EXAMPLE 2 Creating a DU Floppy

The following commands create a Driver Update floppy containing the driver package `MYdriver`.

```
# rmformat -F quick /dev/rdiskette
# mkfs -F pcfs /dev/rdiskette
# mount -F pcfs /dev/diskette /mnt
# /usr/bin/itu makedu -r 5.10 -d /mnt /export/MYdriver
# umount /mnt
# dd if=/dev/rdiskette of=floppy.dd
```

The `itu` command above creates the DU directly onto the floppy mounted on `/mnt`. The `dd` invocation stores an image of the floppy in the file `floppy.dd`. This is useful for distributing a DU floppy in file form. From the file, the floppy can then be recreated as follows:

```
# dd if=floppy.dd of=/dev/rdiskette
```

EXAMPLE 3 Patching the Solaris Install Media

The following command patches the Solaris install media in `/export/s10u1` with patch `/opt/patches/123456-07` and driver package `/opt/pkgs/MYdriver`. The command also creates a bootable ISO image with ISO label "`MyS10U1`" in the file `/tmp/dvd.iso`.

```
# /usr/bin/itu updatemedia -R /export/s10u1 -o /tmp/dvd.iso -l MyS10U1 \
       /opt/patches/123456-07 /opt/pkgs/MYdriver
```

EXAMPLE 4 Creating a Bootable ISO Image

The following commands create the bootable ISO image `mydvd.iso` of the Solaris install image `/export/solaris-10u1` with ISO label "`Special-S10`".

```
# /usr/bin/itu makeiso -o mydvd.iso -l "Special-S10" \
     /export/solaris-10u1
# cdrw -i mydvd.iso
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed

See Also cdrw(1), patchadd(1M), pkgadd(1M), attributes(5)

mkisofs(8), (/usr/share/man/man8/mkisofs.8), in the SUNWfsman package (not a SunOS man page)

Name k5srvutil – host key table (keytab) manipulation utility

Synopsis `/usr/sbin/k5srvutil` *operation* `[-ik] [-f` *filename*`]`

Description The `k5srvutil` command allows a system manager to list or change keys currently in his keytab or to add new keys to the keytab.

The operand *operation* must be one of the following:

`list` Lists the keys in a keytab, showing version number and principal name.

`change` Changes all the keys in the keytab to new randomly-generated keys, updating the keys in the Kerberos server's database to match those by using the `kadmin` protocol. If a key's version number does not match the version number stored in the Kerberos server's database, the operation fails. The old keys are retained so that existing tickets continue to work. If the `-i` flag is specified, `k5srvutil` prompts for `yes` or `no` before changing each key. If the `-k` option is used, the old and new keys are displayed.

`delold` Deletes keys that are not the most recent version from the keytab. This operation should be used at some point after a change operation to remove old keys. If the `-i` flag is specified, `k5srvutil` asks the user whether the old keys associated with each principal should be removed.

`delete` Deletes particular keys in the keytab, interactively prompting for each key.

In all cases, the default keytab file is `/etc/krb5.keytab` file unless this is overridden by the `-f` option.

`k5srvutil` uses the `kadmin(1M)` program to edit the keytab in place. However, old keys are retained, so they are available in case of failure.

Options The following options are supported:

`-f` *filename* Specify a keytab file other than the default file, `/etc/krb5.keytab`.

`-i` Prompts user before changing keys when using the `change` or `delold` operands.

`-k` Displays old and new keys when using the `change` operand.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkdcu
Interface Stability	Committed

See Also ktutil(1), kadmin(1M), attributes(5)

Name kadb – a kernel debugger

Synopsis

SPARC `ok boot` *device_specifier* `kadb [-d] [`*boot-flags*`]`

x86 `select (b)oot or (i)nterpreter: b kadb [-d] [`*boot-flags*`]`

Description `kadb`, an interactive kernel debugger, has been replaced by `kmdb(1)`. For backwards compatibility, the methods used to load `kadb` will be interpreted as requests to load `kmdb(1)`. Unlike with the compatibility link from `adb(1)` to `mdb(1)`, `kmdb(1)` will always load in its native user interface mode, regardless of the name used to load it.

`kmdb(1)` is based on `mdb(1)`, and thus shares `mdb`'s user interface style and feature set. The `mdb(1)` man page describes the features and operation of `mdb`. The `kmdb(1)` man page describes the differences between `mdb` and `kmdb`. This man page describes the major changes and incompatibilities between `kadb` and `kmdb`.

Consult the *Solaris Modular Debugger Guide* for a detailed description of both `mdb` and `kmdb`.

Major changes This section briefly lists the major differences between `kadb` and `kmdb`. It is not intended to be exhaustive.

Debugger Loading and Unloading	`kmdb(1)` may be loaded at boot, as with `kadb`. It may also be loaded after boot, thus allowing for kernel debugging and execution control without requiring a system reboot. If `kmdb(1)` is loaded after boot, it may be unloaded.
mdb Feature Set	The features introduced by `mdb(1)`, including access to kernel type data, debugger commands (dcmds), debugger modules (dmods), and enhanced execution control facilities, are available under `kmdb(1)`. Support for changing the representative CPU (:x) is available for both SPARC and x86. Furthermore, full execution-control facilities are available after the representative CPU has been changed.

Significant Incompatibilities This section lists the significant features that have changed incompatibly between `kadb` and `kmdb(1)`. It is not intended to be exhaustive. All `kmdb(1)` commands referenced here are fully described in the `kmdb(1)` man page. A description as well as examples can be found in the *Solaris Modular Debugger Guide*.

Deferred Breakpoints	The kadb-style “module#symbol:b” syntax is not supported under `kmdb(1)`. Instead, use “::bp module‘symbol”.
Watchpoints	The ::wp dcmd is the preferred way to set watchpoint with `kmdb`. Various options are available to control the type of watchpoint set, including `-p` for physical watchpoints

(SPARC only), and -i for I/O port watchpoints (x86 only). $l is not supported, therefore, the watchpoint size must be specified for each watchpoint created.

Access to I/O Ports (x86 only) — The commands used to access I/O ports under kadb have been replaced with the ::in and ::out dcmds. These two dcmds allow both read and write of all I/O port sizes supported by kadb.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcar

See Also adb(1), mdb(1), kmdb(1), attributes(5)

Solaris Modular Debugger Guide

Name kadmin, kadmin.local – Kerberos database administration program

Synopsis

```
/usr/sbin/kadmin [-r realm] [-p principal] [-q query]
 [-s admin_server [:port]] [ [-c credential_cache]
 | [-k [-t keytab]] | [-w password]] [-x db_args]...

/usr/sbin/kadmin.local [-r realm] [-p principal]
 [-q query] [-d dbname] [-e "enc:salt..."] [-m] [-D]
```

Description `kadmin` and `kadmin.local` are interactive command-line interfaces to the Kerberos V5 administration system. They provide for the maintenance of Kerberos principals, policies, and service key tables (keytabs). `kadmin` and `kadmin.local` provide identical functionality; the difference is that `kadmin.local` can run only on the master KDC and does not use Kerberos authentication.

Except as explicitly noted otherwise, this man page uses `kadmin` to refer to both versions.

By default, both versions of `kadmin` attempt to determine your user name and perform operations on behalf of your "*username*/*admin*" instance. Operations performed are subject to privileges granted or denied to this user instance by the Kerberos ACL file (see `kadm5.acl(4)`). You may perform administration as another user instance by using the `-p` option.

The remote version, `kadmin`, uses Kerberos authentication and an encrypted RPC to operate securely from anywhere on the network. It normally prompts for a password and authenticates the user to the Kerberos administration server, `kadmind`, whose service principal is `kadmin/`*fqdn*. Some options specific to the remote version permit the password prompt to be bypassed. The `-c` option searches the named credentials cache for a valid ticket for the `kadmin/`*fqdn* service and uses it to authenticate the user to the Kerberos admin server without a password. The `-k` option searches a keytab for a credential to authenticate to the `kadmin/`*fqdn* service, and again no password is collected. If `kadmin` has collected a password, it requests a `kadmin/`*fqdn* Kerberos service ticket from the KDC, and uses that service ticket to interact with `kadmind`.

The local version, `kadmin.local`, must be run with an effective UID of root, and normally uses a key from the `/var/krb5/.k5.`*realm* stash file (see `kdb5_util(1M)`) to decrypt information from the database rather than prompting for a password. The `-m` option will bypass the `.k5.`*realm* stash file and prompt for the master password.

Options The following options are supported:

`-c` *credentials_cache*
: Search *credentials_cache* for a service ticket for the `kadmin/`*fqdn* service; it can be acquired with the `kinit(1)` program. If this option is not specified, `kadmin` requests a new service ticket from the KDC, and stores it in its own temporary credentials cache.

`-d` *dbname*
: Specify a non-standard database name. [Local only]

`-D`
: Turn on debug mode. [Local only]

`-e` *"enc:salt ..."*
Specify a different encryption type and/or key salt. [Local only]

`-k` [`-t` *keytab*]
Use the default keytab (`-k`) or a specific keytab (`-t` *keytab*) to decrypt the KDC response instead of prompting for a password. In this case, the default principal will be `host/hostname`. This is primarily used for keytab maintenance.

`-m`
Accept the database master password from the keyboard rather than using the `/var/krb5/.k5.`*realm* stash file. [Local only]

`-p` *principal*
Authenticate *principal* to the `kadmin/`*fqdn* service. Otherwise, `kadmin` will append `/admin` to the primary principal name of the default credentials cache, the value of the `USER` environment variable, or the username as obtained with `getpwuid`, in that order of preference.

`-q` *query*
Pass *query* directly to `kadmin`, which will perform *query* and then exit. This can be useful for writing scripts.

`-r` *realm*
Use *realm* as the default database realm.

`-s` *admin_server*[*:port*]
Administer the specified *admin* server at the specified port number (*port*). This can be useful in administering a realm not known to your client.

`-w` *password*
Use *password* instead of prompting for one. Note that placing the password for a Kerberos principal with administration access into a shell script can be dangerous if unauthorized users gain read access to the script or can read arguments of this command through `ps(1)`.

`-x` *db_args*
Pass database-specific arguments to `kadmin`. Supported arguments are for LDAP and the `Berkeley-db2` plug-in. These arguments are:

`binddn=`*binddn*
LDAP simple bind DN for authorization on the directory server. Overrides the `ldap_kadmind_dn` parameter setting in `krb5.conf(4)`.

`bindpwd=`*bindpwd*
Bind password.

`dbname=`*name*
For the `Berkeley-db2` plug-in, specifies a name for the Kerberos database.

`nconns=`*num*
Maximum number of server connections.

`port=`*num*
Directory server connection port.

Commands `list_requests`
Lists all the commands available for `kadmin`. Aliased by `lr` and `?`.

`get_privs`
Lists the current Kerberos administration privileges (ACLs) for the principal that is currently running `kadmin`. The privileges are based on the `/etc/krb5/kadm5.acl` file on the master KDC. Aliased by `getprivs`.

`add_principal` [*options*] *newprinc*
Creates a new principal, *newprinc*, prompting twice for a password. If the `-policy` option is not specified and a policy named `default` exists, then the `default` policy is assigned to the principal; note that the assignment of the `default` policy occurs automatically only when a principal is first created, so the `default` policy must already exist for the assignment to occur. The automatic assignment of the `default` policy can be suppressed with the `-clearpolicy` option. This command requires the `add` privilege. Aliased by `addprinc` and `ank`. The options are:

`-expire` *expdate*
Expiration date of the principal. See the `Time Formats` section for the valid absolute time formats that you can specify for *expdate*.

`-pwexpire` *pwexpdate*
Password expiration date. See the `Time Formats` section for the valid absolute time formats that you can specify for *pwexpdate*.

`-maxlife` *maxlife*
Maximum ticket life for the principal. See the `Time Formats` section for the valid time duration formats that you can specify for *maxlife*.

`-maxrenewlife` *maxrenewlife*
Maximum renewable life of tickets for the principal. See the `Time Formats` section for the valid time duration formats that you can specify for *maxrenewlife*.

`-kvno` *kvno*
Explicitly set the key version number.

`-policy` *policy*
Policy used by the principal. If both the `-policy` and `-clearpolicy` options are not specified, the `default` policy is used if it exists; otherwise, the principal will have no policy. Also note that the password and principal name must be different when you add a new principal with a specific policy or the `default` policy.

`-clearpolicy`
`-clearpolicy` prevents the `default` policy from being assigned when `-policy` is not specified. This option has no effect if the `default` policy does not exist.

{–|+}allow_postdated
-allow_postdated prohibits the principal from obtaining postdated tickets. (Sets the KRB5_KDB_DISALLOW_POSTDATED flag.) +allow_postdated clears this flag.

{–|+}allow_forwardable
-allow_forwardable prohibits the principal from obtaining forwardable tickets. (Sets the KRB5_KDB_DISALLOW_FORWARDABLE flag.) +allow_forwardable clears this flag.

{–|+}allow_renewable
-allow_renewable prohibits the principal from obtaining renewable tickets. (Sets the KRB5_KDB_DISALLOW_RENEWABLE flag.) +allow_renewable clears this flag.

{–|+}allow_proxiable
-allow_proxiable prohibits the principal from obtaining proxiable tickets. (Sets the KRB5_KDB_DISALLOW_PROXIABLE flag.) +allow_proxiable clears this flag.

{–|+}allow_dup_skey
-allow_dup_skey disables user-to-user authentication for the principal by prohibiting this principal from obtaining a session key for another user. (Sets the KRB5_KDB_DISALLOW_DUP_SKEY flag.) +allow_dup_skey clears this flag.

{–|+}requires_preauth
+requires_preauth requires the principal to preauthenticate before being allowed to kinit. (Sets the KRB5_KDB_REQUIRES_PRE_AUTH flag.) -requires_preauth clears this flag.

{–|+}requires_hwauth
+requires_hwauth requires the principal to preauthenticate using a hardware device before being allowed to kinit. (Sets the KRB5_KDB_REQUIRES_HW_AUTH flag.) -requires_hwauth clears this flag.

{–|+}allow_svr
-allow_svr prohibits the issuance of service tickets for the principal. (Sets the KRB5_KDB_DISALLOW_SVR flag.) +allow_svr clears this flag.

{–|+}allow_tgs_req
–allow_tgs_req specifies that a Ticket-Granting Service (TGS) request for a service ticket for the principal is not permitted. This option is useless for most things. +allow_tgs_req clears this flag. The default is +allow_tgs_req. In effect, –allow_tgs_req sets the KRB5_KDB_DISALLOW_TGT_BASED flag on the principal in the database.

{–|+}allow_tix
–allow_tix forbids the issuance of any tickets for the principal. +allow_tix clears this flag. The default is +allow_tix. In effect, –allow_tix sets the KRB5_KDB_DISALLOW_ALL_TIX flag on the principal in the database.

{–|+}needchange
+needchange sets a flag in attributes field to force a password change; –needchange clears it. The default is –needchange. In effect, +needchange sets the KRB5_KDB_REQUIRES_PWCHANGE flag on the principal in the database.

{–|+}password_changing_service
+password_changing_service sets a flag in the attributes field marking this as a password change service principal (useless for most things). –password_changing_service clears the flag. This flag intentionally has a long name. The default is –password_changing_service. In effect, +password_changing_service sets the KRB5_KDB_PWCHANGE_SERVICE flag on the principal in the database.

-randkey
Sets the key of the principal to a random value.

-pw *password*
Sets the key of the principal to the specified string and does not prompt for a password. Note that using this option in a shell script can be dangerous if unauthorized users gain read access to the script.

-e "enc:salt ..."
Override the list of enctype:salttype pairs given in kdc.conf(4) for setting the key of the principal. The quotes are necessary if there are multiple enctype:salttype pairs. One key for each similar enctype and same salttype will be created and the first one listed will be used. For example, in a list of two similar enctypes with the same salt, "des-cbc-crc:normal des-cbc-md5:normal", one key will be created and it will be of type des-cbc-crc:normal.

Example:

```
kadmin: addprinc tlyu/admin
WARNING: no policy specified for "tlyu/admin@ACME.COM";
defaulting to no policy.
Enter password for principal tlyu/admin@ACME.COM:
Re-enter password for principal tlyu/admin@ACME.COM:
Principal "tlyu/admin@ACME.COM" created.
kadmin:
```

Errors:
KADM5_AUTH_ADD (requires add privilege)

KADM5_BAD_MASK (should not happen)

KADM5_DUP (principal exists already)

KADM5_UNK_POLICY (policy does not exist)

KADM5_PASS_Q_* (password quality violations)

`delete_principal` [`-force`] *principal*
Deletes the specified principal from the database. This command prompts for deletion, unless the `-force` option is given. This command requires the `delete` privilege. Aliased by `delprinc`.

Example:

```
kadmin: delprinc mwm_user
Are you sure you want to delete the principal
"mwm_user@ACME.COM"? (yes/no): yes
Principal "mwm_user@ACME.COM" deleted.
Make sure that you have removed this principal from
all kadmind ACLs before reusing.
kadmin:
```

Errors:
`KADM5_AUTH_DELETE` (requires `delete` privilege)

`KADM5_UNK_PRINC` (principal does not exist)

`modify_principal` [*options*] *principal*
Modifies the specified principal, changing the fields as specified. The options are as above for `add_principal`, except that password changing is forbidden by this command. In addition, the option `-clearpolicy` will clear the current policy of a principal. This command requires the `modify` privilege. Aliased by `modprinc`.

Errors:
`KADM5_AUTH_MODIFY` (requires `modify` privilege)

`KADM5_UNK_PRINC` (principal does not exist)

`KADM5_UNK_POLICY` (policy does not exist)

`KADM5_BAD_MASK` (should not happen)

`change_password` [*options*] *principal*
Changes the password of *principal*. Prompts for a new password if neither `-randkey` or `-pw` is specified. Requires the `changepw` privilege, or that the principal that is running the program to be the same as the one changed. Aliased by `cpw`. The following options are available:

`-randkey`
Sets the key of the principal to a random value.

`-pw` *password*
Sets the password to the specified string. Not recommended.

`-e` “enc:salt ...”
Override the list of enctype:salttype pairs given in `kdc.conf(4)` for setting the key of the principal. The quotes are necessary if there are multiple enctype:salttype pairs. For each key, the first matching similar enctype and same salttype in the list will be used to set the new key(s).

`-keepold`
Keeps the previous kvno's keys around. There is no easy way to delete the old keys, and this flag is usually not necessary except perhaps for TGS keys as it will allow existing valid TGTs to continue to work.

Example:

```
kadmin: cpw systest
Enter password for principal systest@ACME.COM:
Re-enter password for principal systest@ACME.COM:
Password for systest@ACME.COM changed.
kadmin:
```

Errors:
`KADM5_AUTH_MODIFY` (requires the `modify` privilege)

`KADM5_UNK_PRINC` (principal does not exist)

`KADM5_PASS_Q_*` (password policy violation errors)

`KADM5_PASS_REUSE` (password is in principal's password history)

`KADM5_PASS_TOOSOON` (current password minimum life not expired)

`get_principal` [`-terse`] *principal*
Gets the attributes of *principal*. Requires the `inquire` privilege, or that the principal that is running the program to be the same as the one being listed. With the `-terse` option, outputs fields as quoted tab-separated strings. Aliased by `getprinc`.

Examples:

```
kadmin: getprinc tlyu/admin
Principal: tlyu/admin@ACME.COM
Expiration date: [never]
Last password change: Mon Aug 12 14:16:47 EDT 1996
Password expiration date: [none]
Maximum ticket life: 0 days 10:00:00
Maximum renewable life: 7 days 00:00:00
Last modified: Mon Aug 12 14:16:47 EDT 1996
(example_user/admin@ACME.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 2 Key: vno 1, DES cbc mode with CRC-32,
```

```
no salt Key: vno 1, DES cbc mode with CRC-32,
Version 4 Attributes:
Policy: [none]
kadmin: getprinc -terse systest
systest@ACME.COM 3    86400    604800  1 785926535    753241234
785900000
tlyu/admin@ACME.COM    786100034    0    0
kadmin:
```

Errors:

`KADM5_AUTH_GET` (requires the get [inquire] privilege)

`KADM5_UNK_PRINC` (principal does not exist)

`list_principals` [*expression*]

Retrieves all or some principal names. *expression* is a shell-style glob expression that can contain the wild-card characters ?, *, and []'s. All principal names matching the expression are printed. If no expression is provided, all principal names are printed. If the expression does not contain an "@" character, an "@" character followed by the local realm is appended to the expression. Requires the `list` privilege. Aliased by `listprincs`, `get_principals`, and `getprincs`.

Examples:

```
kadmin: listprincs test*
test3@ACME.COM
test2@ACME.COM
test1@ACME.COM
testuser@ACME.COM
kadmin:
```

`add_policy` [*options*] *policy*

Adds the named policy to the policy database. Requires the `add` privilege. Aliased by `addpol`. The following options are available:

`-maxlife` *maxlife*

sets the maximum lifetime of a password. See the `Time Formats` section for the valid time duration formats that you can specify for *maxlife*.

`-minlife` *minlife*

sets the minimum lifetime of a password. See the `Time Formats` section for the valid time duration formats that you can specify for *minlife*.

`-minlength` *length*

sets the minimum length of a password.

`-minclasses` *number*

sets the minimum number of character classes allowed in a password. The valid values are:

1
only letters (himom)

2
both letters and numbers (hi2mom)

3
letters, numbers, and punctuation (hi2mom!)

`-history` *number*
sets the number of past keys kept for a principal.

Errors:
`KADM5_AUTH_ADD` (requires the `add` privilege)

`KADM5_DUP` (policy already exists)

`delete_policy` `[-force]` *policy*
Deletes the named policy. Unless the `-force` option is specified, prompts for confirmation before deletion. The command will fail if the policy is in use by any principals. Requires the `delete` privilege. Aliased by `delpol`.

Example:

```
kadmin: del_policy guests
Are you sure you want to delete the
policy "guests"? (yes/no): yes
Policy "guests" deleted.
kadmin:
```

Errors:
`KADM5_AUTH_DELETE` (requires the delete privilege)

`KADM5_UNK_POLICY` (policy does not exist)

`KADM5_POLICY_REF` (reference count on policy is not zero)

`modify_policy` `[`*options*`]` *policy*
Modifies the named policy. Options are as above for `add_policy`. Requires the `modify` privilege. Aliased by `modpol`.

Errors:
`KADM5_AUTH_MODIFY` (requires the `modify` privilege)

`KADM5_UNK_POLICY` (policy does not exist)

`get_policy` `[-terse]` *policy*
Displays the values of the named policy. Requires the `inquire` privilege. With the `-terse` flag, outputs the fields as quoted strings separated by tabs. Aliased by `getpol`.

Examples:

```
kadmin: get_policy admin
Policy: admin
Maximum password life: 180 days 00:00:00
Minimum password life: 00:00:00
Minimum password length: 6
Minimum number of password character classes: 2
Number of old keys kept: 5
Reference count: 17
kadmin: get_policy -terse
admin admin    15552000    0    6    2    5    17
kadmin:
```

Errors:

`KADM5_AUTH_GET` (requires the `get` privilege)

`KADM5_UNK_POLICY` (policy does not exist)

`list_policies` [*expression*]

Retrieves all or some policy names. *expression* is a shell-style glob expression that can contain the wild-card characters ?, *, and []'s. All policy names matching the expression are printed. If no expression is provided, all existing policy names are printed. Requires the `list` privilege. Aliased by `listpols`, `get_policies`, and `getpols`.

Examples:

```
kadmin: listpols
test-pol dict-only once-a-min test-pol-nopw
kadmin: listpols t*
test-pol test-pol-nopw kadmin:
```

`ktadd` [`-k` *keytab*] [`-q`] [`-e` *enctype*`:salt`]

Adds a principal or all principals matching *princ-exp* to a keytab, randomizing each principal's key in the process.

`ktadd` requires the `inquire` and `changepw` privileges. An entry for each of the principal's unique encryption types is added, ignoring multiple keys with the same encryption type but different `salt` types. If the `-k` argument is not specified, the default keytab file, `/etc/krb5/krb5.keytab`, is used.

The "`-e` *enctype*`:salt`" option overrides the list of *enctypes* given in `krb5.conf(4)`, in the `permitted_enctypes` parameter. If "`-e` *enctype*`:salt`" is not used and `permitted_enctypes` is not defined in `krb5.conf(4)`, a key for each *enctype* supported by the system on which `kadmin` is run will be created and added to the `keytab`. Restricting the *enctypes* of keys in the `keytab` is useful when the system for which keys are being created does not support the same set of *enctypes* as the KDC. Note that `ktadd` modifies the *enctype* of the keys in the principal database as well.

If the `-q` option is specified, less status information is displayed. Aliased by `xst`. The `-glob` option requires the `list` privilege. Also, note that if you use `-glob` to create a keytab, you need to remove `/etc/krb5/kadm5.keytab` and create it again if you want to use `-p */admin` with `kadmin`.

`princ-exp`
princ-exp follows the same rules described for the `list_principals` command.

Example:

```
kadmin: ktadd -k /tmp/new-keytab nfs/chicago
Entry for principal nfs/chicago with kvno 2,
encryption type DES-CBC-CRC added to keytab
WRFILE:/tmp/new-keytab.
kadmin:
```

`ktremove` [`-k` *keytab*] [`-q`] *principal* [*kvno* | `all` | `old`]
Removes entries for the specified principal from a keytab. Requires no privileges, since this does not require database access. If `all` is specified, all entries for that principal are removed; if `old` is specified, all entries for that principal except those with the highest kvno are removed. Otherwise, the value specified is parsed as an integer, and all entries whose kvno match that integer are removed. If the `-k` argument is not specified, the default keytab file, `/etc/krb5/krb5.keytab`, is used. If the `-q` option is specified, less status information is displayed. Aliased by `ktrem`.

Example:

```
kadmin: ktremove -k /tmp/new-keytab nfs/chicago
Entry for principal nfs/chicago with kvno 2
removed from keytab
WRFILE:/tmp/new-keytab.
kadmin:
```

`quit`
Quits `kadmin`. Aliased by `exit` and `q`.

Time Formats

Various commands in `kadmin` can take a variety of time formats, specifying time durations or absolute times. The `kadmin` option variables *maxrenewlife*, *maxlife*, and *minlife* are time durations, whereas *expdate* and *pwexpdate* are absolute times.

Examples:

```
kadmin: modprinc -expire "12/31 7pm" jdb
kadmin: modprinc -maxrenewlife "2 fortnight" jdb
kadmin: modprinc -pwexpire "this sunday" jdb
kadmin: modprinc -expire never jdb
kadmin: modprinc -maxlife "7:00:00pm tomorrow" jdb
```

Note that times which do not have the “ago” specifier default to being absolute times, unless they appear in a field where a duration is expected. In that case, the time specifier will be interpreted as relative. Specifying “ago” in a duration can result in unexpected behavior.

The following time formats and units can be combined to specify a time. The time and date format examples are based on the date and time of July 2, 1999, 1:35:30 p.m.

Time Format	Examples
hh[:*mm*][:*ss*][am/pm/a.m./p.m.]	1p.m., 1:35, 1:35:30pm

Variable	Description
hh	hour (12-hour clock, leading zero permitted but not required)
mm	minutes
ss	seconds

Date Format	Examples
mm/dd[/*yy*]	07/02, 07/02/99
yyyy-*mm*-dd	1999-07-02
dd-*month*-*yyyy*	02-July-1999
month [,*yyyy*]	Jul 02, July 02,1999
dd *month*[*yyyy*]	02 JULY, 02 july 1999

Variable Description	
dd	day
mm	month
yy	year within century (00-38 is 2000 to 2038; 70-99 is 1970 to 1999)
yyyy	year including century
month	locale's full or abbreviated month name

Time Units	Examples
[+|- #] year	“-2 year”
[+|- #] month	“2 months”
[+|- #] fortnight	

[+|- #] week	
[+|- #] day	
[+|- #] hour	
[+|- #] minute	
[+|- #] min	
[+|- #] second	
[+|- #] sec	
tomorrow	
yesterday	
today	
now	
this	"this year"
last	"last saturday"
next	"next month"
sunday	
monday	
tuesday	
wednesday	
thursday	
friday	
saturday	
never	

You can also use the following time modifiers: `first`, `second`, `third`, `fourth`, `fifth`, `sixth`, `seventh`, `eighth`, `ninth`, `tenth`, `eleventh`, `twelfth`, and `ago`.

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `kadmin`:

`PAGER`
The command to use as a filter for paging output. This can also be used to specify options. The default is `more(1)`.

Files

`/var/krb5/principal`
Kerberos principal database.

`/var/krb5/principal.ulog`
The update log file for incremental propagation.

`/var/krb5/principal.kadm5`
Kerberos administrative database. Contains policy information.

`/var/krb5/principal.kadm5.lock`
Lock file for the Kerberos administrative database. This file works backwards from most other lock files (that is, `kadmin` will exit with an error if this file does *not* exist).

`/var/krb5/kadm5.dict`
Dictionary of strings explicitly disallowed as passwords.

`/etc/krb5/kadm5.acl`
List of principals and their `kadmin` administrative privileges.

`/etc/krb5/kadm5.keytab`
Keytab for `kadmind` principals: `kadmin/`*fqdn*, `changepw/`*fqdn*, and `kadmin/changepw`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkdcu
Interface Stability	Evolving

See Also

`kpasswd(1)`, `more(1)`, `gkadmin(1M)`, `kadmind(1M)`, `kdb5_util(1M)`, `kdb5_ldap_util(1M)`, `kproplog(1M)`, `kadm5.acl(4)`, `kdc.conf(4)`, `krb5.conf(4)`, `attributes(5)`, `environ(5)`, `kerberos(5)`, `krb5envvar(5)`

History

The `kadmin` program was originally written by Tom Yu at MIT, as an interface to the OpenVision Kerberos administration program.

Diagnostics

The `kadmin` command is currently incompatible with the MIT `kadmind` daemon interface, so you cannot use this command to administer an MIT-based Kerberos database. However, clients running the Solaris implementation of Kerberos can still use an MIT-based KDC.

Name kadmind – Kerberos administration daemon

Synopsis

```
/usr/lib/krb5/kadmind [-d] [-m] [-p port-number] [-r realm]
     -x db_args]...
```

Description `kadmind` runs on the master key distribution center (KDC), which stores the principal and policy databases. `kadmind` accepts remote requests to administer the information in these databases. Remote requests are sent, for example, by `kpasswd(1)`, `gkadmin(1M)`, and `kadmin(1M)` commands, all of which are clients of `kadmind`. When you install a KDC, `kadmind` is set up in the `init` scripts to start automatically when the KDC is rebooted.

`kadmind` requires a number of configuration files to be set up for it to work:

`/etc/krb5/kdc.conf`
: The KDC configuration file contains configuration information for the KDC and the Kerberos administration system. `kadmind` understands a number of configuration variables (called relations) in this file, some of which are mandatory and some of which are optional. In particular, `kadmind` uses the `acl_file`, `dict_file`, `admin_keytab`, and `kadmind_port` relations in the [*realms*] section. Refer to the `kdc.conf(4)` man page for information regarding the format of the KDC configuration file.

`/etc/krb5/kadm5.keytab`
: `kadmind` requires a `keytab` (key table) containing correct entries for the `kadmin/`*fqdn*, `kadmin/changepw` and `kadmin/changepw` principals for every realm that `kadmind` answers requests. The `keytab` can be created with the `kadmin(1M)` or `kdb5_util(1M)` command. The location of the keytab is determined by the `admin_keytab` relation in the `kdc.conf`(4) file.

`/etc/krb5/kadm5.acl`
: `kadmind` uses an ACL (access control list) to determine which principals are allowed to perform Kerberos administration actions. The path of the ACL file is determined by the `acl_file` relation in the `kdc.conf` file. See `kdc.conf(4)`. For information regarding the format of the ACL file, refer to `kadm5.acl(4)`.

  The `kadmind` daemon will need to be restarted to reread the `kadm5.acl` file after it has been modified. You can do this, as root, with the following command:

  ```
  # svcadm restart svc:/network/security/kadmin:default
  ```

After `kadmind` begins running, it puts itself in the background and disassociates itself from its controlling terminal.

`kadmind` can be configured for incremental database propagation. Incremental propagation allows slave KDC servers to receive principal and policy updates incrementally instead of receiving full dumps of the database. These settings can be changed in the `kdc.conf(4)` file:

`sunw_dbprop_enable = [true | false]`
: Enable or disable incremental database propagation. Default is `false`.

`sunw_dbprop_master_ulogsize = N`
Specifies the maximum amount of log entries available for incremental propagation to the slave KDC servers. The maximum value that this can be is 2500 entries. Default value is 1000 entries.

The `kiprop/`<*hostname*>`@`<*REALM*> principal must exist in the master's `kadm5.keytab` file to enable the slave to authenticate incremental propagation from the master. In the principal syntax above, <*hostname*> is the master KDC's host name and <*REALM*> is the realm in which the master KDC resides.

Kerberos client machines can automatically migrate Unix users to the default Kerberos realm specified in the local `krb5.conf(4)`, if the user does not have a valid kerberos account already. You achieve this by using the `pam_krb5_migrate(5)` service module for the service in question. The Kerberos service principal used by the client machine attempting the migration needs to be validated using the `u` privilege in `kadm5.acl(4)`. When using the `u` privilege, `kadmind` validates user passwords using PAM, specifically using a `PAM_SERVICE` name of `k5migrate` by calling `pam_authenticate(3PAM)` and `pam_acct_mgmt(3PAM)`.

A suitable PAM stack configuration example for `k5migrate` would look like:

```
k5migrate        auth    required        pam_unix_auth.so.1
k5migrate        account required        pam_unix_account.so.1
```

Options The following options are supported:

`-d`
Specifies that `kadmind` does not put itself in the background and does not disassociate itself from the terminal. In normal operation, you should use the default behavior, which is to allow the daemon to put itself in the background.

`-m`
Specifies that the master database password should be retrieved from the keyboard rather than from the stash file. When using `-m`, the `kadmind` daemon receives the password prior to putting itself in the background. If used in combination with the `-d` option, you must explicitly place the daemon in the background.

`-p` *port-number*
Specifies the port on which the `kadmind` daemon listens for connections. The default is controlled by the `kadmind_port` relation in the `kdc.conf(4)` file.

`-r` *realm*
Specifies the default realm that `kadmind` serves. If *realm* is not specified, the default *realm* of the host is used. `kadmind` answers requests for any realm that exists in the local KDC database and for which the appropriate principals are in its `keytab`.

`-x` *db_args*
Pass database-specific arguments to `kadmind`. Supported arguments are for LDAP and the `Berkeley-db2` plug-in. These arguments are:

`binddn=`*binddn*
LDAP simple bind DN for authorization on the directory server. Overrides the `ldap_kadmind_dn` parameter setting in `krb5.conf(4)`.

`bindpwd=`*bindpwd*
Bind password.

`dbname=`*name*
For the `Berkeley-db2` plug-in, specifies a name for the Kerberos database.

`nconns=`*num*
Maximum number of server connections.

`port=`*num*
Directory server connection port.

Files `/var/krb5/principal`
Kerberos principal database.

`/var/krb5/principal.ulog`
The update log file for incremental propagation.

`/var/krb5/principal.kadm5`
Kerberos administrative database containing policy information.

`/var/krb5/principal.kadm5.lock`
Kerberos administrative database lock file. This file works backwards from most other lock files (that is, `kadmin` exits with an error if this file does not exist).

`/var/krb5/kadm5.dict`
Dictionary of strings explicitly disallowed as passwords.

`/etc/krb5/kadm5.acl`
List of principals and their `kadmin` administrative privileges.

`/etc/krb5/kadm5.keytab`
Keytab for `kadmin` principals: `kadmin/`*fqdn*, `changepw/`*fqdn*, and `kadmin/changepw`.

`/etc/krb5/kdc.conf`
KDC configuration information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkdcu
Interface Stability	Evolving

See Also kpasswd(1), svcs(1), gkadmin(1M), kadmin(1M), kadmin(1M), kdb5_util(1M), kdb5_ldap_util(1M), kproplog(1M), svcadm(1M), pam_acct_mgmt(3PAM), pam_authenticate(3PAM), kadm5.acl(4), kdc.conf(4), krb5.conf(4), attributes(5), kerberos(5), krb5envvar(5), pam_krb5_migrate(5), smf(5)

Notes The Kerberos administration daemon (kadmind) is now compliant with the change-password standard mentioned in RFC 3244, which means it can now handle change-password requests from non-Solaris Kerberos clients.

The kadmind service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/security/kadmin
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name kcfd – kernel-level cryptographic framework daemon

Synopsis

```
kcfd
```

Description The `kcfd` daemon helps in managing CPU usage by cryptographic operations performed in software by kernel threads. The system utilization associated with these threads is charged to the `kcfd` process. It also does module verification for kernel cryptographic modules.

Only a privileged user can run this daemon.

The `kcfd` daemon is automatically invoked in run level 1, after `/usr` is mounted. A previously invoked `kcfd` daemon that is still running must be stopped before invoking another `kcfd` command.

Manually starting and restarting `kcfd` is not recommended. If it is necessary to do so, use the `cryptoadm`(1M) `start` and `stop` subcommands.

Exit Status The following exit values are returned:

`0` Daemon started successfully.

`> 1` Daemon failed to start.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsl/SUNWcslx
Interface Stability	Evolving

See Also `cryptoadm`(1M), `attributes`(5)

Name kclient – set up a machine as a Kerberos client

Synopsis

```
/usr/sbin/kclient [-n] [-R realm] [-k kdc] [-a adminuser]
     [-c filepath] [-d dnsarg] [-f fqdn_list] [-p profile]
```

Description You can use the `kclient` utility to:

- Configure a machine as a Kerberos client for a specified realm and for KDC by setting up `krb5.conf(4)`.
- Add the Kerberos host principal to the local host's `keytab` file (`/etc/krb5/krb5.keytab`).
- Optionally set up the machine to do kerberized NFS.
- Optionally bring over a master `krb5.conf` copy from a specified pathname.
- Optionally setup a machine to do server and/or host/domain name-to-realm mapping lookups by means of DNS.

The `kclient` utility needs to be run on the client machine with root permission and can be run either interactively or non-interactively. In the non-interactive mode, the user feeds in the required inputs by means of a profile, command-line options, or a combination of profile and command-line options. The user is prompted for "required" parameter values (`realm`, `kdc`, and `adminuser`), if found missing in the non-interactive run. The interactive mode is invoked when the utility is run without any command-line arguments.

Both the interactive and non-interactive forms of `kclient` always add the `host/fqdn` entry to the local host's `keytab` file. They also require the user to enter the password for the administrative user requested, to obtain the Kerberos Ticket Granting Ticket (TGT) for `adminuser`. The `host/fqdn`, `nfs/fqdn`, and `root/fqdn` principals are added to the KDC database (if not already present) before their addition to the local host's `keytab`.

The `kclient` utility assumes that the local host has been setup for DNS and requires the presence of a valid `resolv.conf(4)`. Also, `kclient` can fail if the localhost time is not synchronized with that of the KDC. For Kerberos to function the localhost time must be within five minutes of that of the KDC. It is advised that both systems run some form of time synchronization protocol, such as the Network Time Protocol (NTP). See `xntpd(1M)`.

Options The non-interactive mode supports the following options:

`-n`	Set up the machine for kerberized NFS. This involves making changes to `nfssec.conf(4)` and addition of the `nfs/fqdn` and `root/fqdn` entries to the local host's `keytab` file.
`-R` [*realm*]	Specifies the Kerberos realm.
`-k` [*kdc*]	Specifies the machine to be used as the Kerberos Key Distribution Center (KDC).
`-a` [*adminuser*]	Specifies the Kerberos administrative user.

`-c` [*filepath*] Specifies the pathname to the `krb5.conf(4)` master file, to be copied over to the local host. The path specified normally points to a master copy on a remote host and brought over to the local host by means of NFS.

`-d` [*dnsarg*] Specifies the DNS lookup option to be used and specified in the `krb5.conf(4)` file. Valid *dnsarg* entries are: `none`, `dns_lookup_kdc`, `dns_lookup_realm` and `dns_fallback`. Any other entry is considered invalid. The latter three *dnsarg* values assume the same meaning as those described in `krb5.conf`. `dns_lookup_kdc` implies DNS lookups for the KDC and the other servers. `dns_lookup_realm` is for host/domain name-to-realm mapping by means of DNS. `dns_fallback` is a superset and does DNS lookups for both the servers and the host/domain name-to-realm mapping. A lookup option of `none` specifies that DNS is not be used for any kind of mapping lookup.

`-f` [*fqdn_list*] This option creates a service principal entry (host/nfs/root) associated with each of the listed fqdn's, if required, and subsequently adds the entries to the local host's `keytab`.

fqdn_list is a comma-separated list of one or more fully qualified DNS domain names.

This option is especially useful in Kerberos realms having systems offering kerberized services, but situated in multiple different DNS domains.

`-p` [*profile*] Specifies the profile to be used to enable the reading in of the values of all the parameters required for setup of the machine as a Kerberos client.

The profile should have entries in the format:

PARAM <value>

Valid *PARAM* entries are: `REALM`, `KDC`, `ADMIN`, `FILEPATH`, `NFS`, `DNSLOOKUP`, and `FQDN`. These profile entries correspond to the `-R` [*realm*], `-k` [*kdc*], `-a` [*adminuser*], `-c` [*filepath*], `-n`, `-d` [*dnsarg*], and `-f` [*fqdn_list*] command-line options, respectively. Any other *PARAM* entry is considered invalid and is ignored.

The NFS profile entry can have a value of 0 (do nothing) or 1 (operation is requested). Any other value is considered invalid and is ignored.

Keep in mind that the command line options override the *PARAM* values listed in the profile.

Examples

EXAMPLE 1 Setting Up a Kerberos Client Using Command-Line Options

To setup a Kerberos client using the `clntconfig/admin` administrative principal for realm `'ABC.COM'`, `kdc` '`example1.com`' and that also does kerberized NFS, enter:

```
# /usr/sbin/kclient -n -R ABC.COM -k example1.com -a clntconfig
```

Alternatively, to set up a Kerberos client using the `clntconfig/admin` administrative principal for the realm '`EAST.ABC.COM`', `kdc` '`example2.east.abc.com`' and that also needs service principal(s) created and/or added to the local `keytab` for multiple DNS domains, enter:

```
# /usr/sbin/kclient -n -R EAST.ABC.COM -k example2.east.abc.com \
-f west.abc.com,central.abc.com -a clntconfig
```

Note that the `krb5` administrative principal used by the administrator needs to have only `add`, `inquire`, `change-pwd` and `modify` privileges (for the principals in the KDC database) in order for the `kclient` utility to run. A sample `kadm5.acl(4)` entry is:

```
clntconfig/admin@ABC.COM acmi
```

EXAMPLE 2 Setting Up a Kerberos Client Using the Profile Option

To setup a Kerberos client using the `clntconfig/admin` administrative principal for realm '`ABC.COM`', `kdc` '`example1.com`' and that also copies over the master `krb5.conf` from a specified location, enter:

```
# /usr/sbin/kclient -p /net/example1.com/export/profile.krb5
```

The contents of `profile.krb5`:

```
REALM ABC.COM
KDC example1
ADMIN clntconfig
FILEPATH /net/example1.com/export/krb5.conf
NFS 0
DNSLOOKUP none
```

Files

`/etc/krb5/kadm5.acl`	Kerberos access control list (ACL) file.
`/etc/krb5/krb5.conf`	Default location for the local host's configuration file.
`/etc/krb5/krb5.keytab`	Default location for the local host's `keytab` file.
`/etc/nfssec.conf`	File listing NFS security modes.
`/etc/resolv.conf`	DNS resolver configuration file.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkdcu
Interface Stability	Evolving

See Also xntpd(1M), kadm5.acl(4), krb5.conf(4), nfssec.conf(4), resolv.conf(4), attributes(5)

Notes `fqdn` stands for the Fully Qualified Domain Name of the local host. The `kclient` utility saves copies of both the krb5.conf(4) and nfssec.conf(4) files to files with corresponding names and `.sav` extensions. The optional copy of the krb5.conf(4) master file is neither encrypted nor integrity-protected and it takes place over regular NFS.

Name kdb5_ldap_util – Kerberos configuration utility

Synopsis

```
kdb5_ldap_util  [-D user_dn [-w passwd]] [-H ldap_uri] command
    [command_options]
```

Description The `kdb5_ldap_util` utility allows an administrator to manage realms, Kerberos services, and ticket policies. The utility offers a set of general options, described under OPTIONS, and a set of commands, which, in turn, have their own options. Commands and their options are described in their own subsections, below.

Options `kdb5_ldap_util` has a small set of general options that apply to the `kdb5_ldap_util` utility itself and a larger number of options that apply to specific commands. A number of these command-specific options apply to multiple commands and are described in their own section, below.

General Options The following general options are supported:

`-D` *user_dn*
Specifies the distinguished name (DN) of a user who has sufficient rights to perform the operation on the LDAP server.

`-H` *ldap_uri*
Specifies the URI of the LDAP server.

`-w` *passwd*
Specifies the password of *user_dn*. This option is not recommended.

Common Command-specific Options The following options apply to a number of `kdb5_ldap_util` commands.

`-subtrees` *subtree_dn_list*
Specifies the list of subtrees containing the principals of a realm. The list contains the DNs of the subtree objects separated by a colon.

`-sscope` *search_scope*
Specifies the scope for searching the principals under a subtree. The possible values are 1 or `one` (one level), 2 or `sub` (subtrees).

`-containerref` *container_reference_dn*
Specifies the DN of the container object in which the principals of a realm will be created. If the container reference is not configured for a realm, the principals will be created in the realm container.

`-maxtktlife` *max_ticket_life*
Specifies maximum ticket life for principals in this realm.

`-maxrenewlife` *max_renewable_ticket_life*
Specifies maximum renewable life of tickets for principals in this realm.

`-r` *realm*
Specifies the Kerberos realm of the database; by default the realm returned by `krb5_default_local_realm(3)` is used.

kdb5_ldap_util Commands

The `kdb5_ldap_util` utility comprises a set of commands, each with its own set of options. These commands are described in the following subsections.

The `create` Command

The `create` command creates a realm in a directory. The command has the following syntax:

```
create \
[-subtrees subtree_dn_list]
[-sscope search_scope]
[-containerref container_reference_dn]
[-k mkeytype]
[-m|-P password| -sf stashfilename]
[-s]
[-r realm]
[-maxtktlife max_ticket_life]
[-kdcdn kdc_service_list]
[-admindn admin_service_list]
[-maxrenewlife max_renewable_ticket_life]
[ticket_flags]
```

The `create` command has the following options:

`-subtree` *subtree_dn_list*
See "Common Command-specific Options," above.

`-sscope` *search_scope*
See "Common Command-specific Options," above.

`-containerref` *container_reference_dn*
See "Common Command-specific Options," above.

`-k` *mkeytype*
Specifies the key type of the master key in the database; the default is that given in `kdc.conf(4)`.

`-m`
Specifies that the master database password should be read from the TTY rather than fetched from a file on the disk.

`-P` *password*
Specifies the master database password. This option is not recommended.

`-sf` *stashfilename*
Specifies the stash file of the master database password.

`-s`
Specifies that the stash file is to be created.

`-maxtktlife` *max_ticket_life*
See "Common Command-specific Options," above.

`-maxrenewlife` *max_renewable_ticket_life*
See "Common Command-specific Options," above.

`-r` *realm*
See "Common Command-specific Options," above.

ticket_flags
Specifies the ticket flags. If this option is not specified, by default, none of the flags are set. This means all the ticket options will be allowed and no restriction will be set. See "Ticket Flags" for a list and descriptions of these flags.

The `modify` Command

The `modify` command modifies the attributes of a realm. The command has the following syntax:

```
modify \
[-subtrees subtree_dn_list]
[-sscope search_scope]
[-containerref container_reference_dn]
[-r realm]
[-maxtktlife max_ticket_life]
[-maxrenewlife max_renewable_ticket_life]
[ticket_flags]
```

The `modify` command has the following options:

`-subtree` *subtree_dn_list*
See "Common Command-specific Options," above.

`-sscope` *search_scope*
See "Common Command-specific Options," above.

`-containerref` *container_reference_dn*
See "Common Command-specific Options," above.

`-maxtktlife` *max_ticket_life*
See "Common Command-specific Options," above.

`-maxrenewlife` *max_renewable_ticket_life*
See "Common Command-specific Options," above.

`-r` *realm*
See "Common Command-specific Options," above.

ticket_flags
Specifies the ticket flags. If this option is not specified, by default, none of the flags are set. This means all the ticket options will be allowed and no restriction will be set. See "Ticket Flags" for a list and descriptions of these flags.

The `view` Command

The `view` command displays the attributes of a realm. The command has the following syntax:

```
view [-r realm]
```

The `view` command has the following option:

`-r` *realm*
See "Common Command-specific Options," above.

The `destroy` Command

The `destroy` command destroys a realm, including the master key stash file. The command has the following syntax:

```
destroy [-f] [-r realm]
```

The `destroy` command has the following options:

`-f`
If specified, `destroy` does not prompt you for confirmation.

`-r` *realm*
See "Common Command-specific Options," above.

The `list` Command

The `list` command displays the names of realms. The command has the following syntax:

```
list
```

The `list` command has no options.

The `stashsrvpw` Command

The `stashsrvpw` command enables you to store the password for service object in a file so that a KDC and Administration server can use it to authenticate to the LDAP server. The command has the following syntax:

```
stashsrvpw [-f filename] servicedn
```

The `stashsrvpw` command has the following option and argument:

`-f` *filename*
Specifies the complete path of the service password file. The default is:

```
/var/krb5/service_passwd
```

servicedn
Specifies the distinguished name (DN) of the service object whose password is to be stored in file.

The `create_policy` Command

The `create_policy` command creates a ticket policy in a directory. The command has the following syntax:

```
create_policy \
[-r realm]
[-maxtktlife max_ticket_life]
[-maxrenewlife max_renewable_ticket_life]
[ticket_flags]
policy_name
```

The `create_policy` command has the following options:

`-r` *realm*
See "Common Command-specific Options," above.

`-maxtktlife` *max_ticket_life*
See "Common Command-specific Options," above.

`-maxrenewlife` *max_renewable_ticket_life*
See "Common Command-specific Options," above.

ticket_flags
Specifies the ticket flags. If this option is not specified, by default, none of the flags are set. This means all the ticket options will be allowed and no restriction will be set. See "Ticket Flags" for a list and descriptions of these flags.

policy_name
Specifies the name of the ticket policy.

The `modify_policy` Command

The `modify_policy` command modifies the attributes of a ticket policy. The command has the following syntax:

```
modify_policy \
[-r realm]
[-maxtktlife max_ticket_life]
[-maxrenewlife max_renewable_ticket_life]
[ticket_flags]
policy_name
```

The `modify_policy` command has the same options and argument as those for the `create_policy` command.

The `view_policy` Command

The `view_policy` command displays the attributes of a ticket policy. The command has the following syntax:

```
view_policy [-r realm] policy_name
```

The `view_policy` command has the following options:

`-r` *realm*
See "Common Command-specific Options," above.

policy_name
Specifies the name of the ticket policy.

The `destroy_policy` Command

The `destroy_policy` command destroys an existing ticket policy. The command has the following syntax:

```
destroy_policy [-r realm] [-force] policy_name
```

The `destroy_policy` command has the following options:

`-r` *realm*
See "Common Command-specific Options," above.

`-force`
Forces the deletion of the policy object. If not specified, you will be prompted for confirmation before the policy is deleted. Enter `yes` to confirm the deletion.

policy_name
Specifies the name of the ticket policy.

The `list_policy` Command

The `list_policy` command lists the ticket policies in the default or a specified realm. The command has the following syntax:

```
list_policy [-r realm]
```

The `list_policy` command has the following option:

`-r` *realm*
See "Common Command-specific Options," above.

Ticket Flags

A number of `kdb5_ldap_util` commands have `ticket_flag` options. These flags are described as follows:

`{-|+}allow_dup_skey`
`-allow_dup_skey` disables user-to-user authentication for principals by prohibiting principals from obtaining a session key for another user. This setting sets the `KRB5_KDB_DISALLOW_DUP_SKEY` flag. `+allow_dup_skey` clears this flag.

`{-|+}allow_forwardable`
`-allow_forwardable` prohibits principals from obtaining forwardable tickets. This setting sets the `KRB5_KDB_DISALLOW_FORWARDABLE` flag. `+allow_forwardable` clears this flag.

`{-|+}allow_postdated`
`-allow_postdated` prohibits principals from obtaining postdated tickets. This setting sets the `KRB5_KDB_DISALLOW_POSTDATED` flag. `+allow_postdated` clears this flag.

`{-|+}allow_proxiable`
`-allow_proxiable` prohibits principals from obtaining proxiable tickets. This setting sets the `KRB5_KDB_DISALLOW_PROXIABLE` flag. `+allow_proxiable` clears this flag.

`{-|+}allow_renewable`
`-allow_renewable` prohibits principals from obtaining renewable tickets. This setting sets the `KRB5_KDB_DISALLOW_RENEWABLE` flag. `+allow_renewable` clears this flag.

`{-|+}allow_svr`
`-allow_svr` prohibits the issuance of service tickets for principals. This setting sets the `KRB5_KDB_DISALLOW_SVR` flag. `+allow_svr` clears this flag.

`{-|+}allow_tgs_req`
`-allow_tgs_req` specifies that a Ticket-Granting Service (TGS) request for a service ticket for principals is not permitted. This option is useless for most purposes. `+allow_tgs_req` clears this flag. The default is `+allow_tgs_req`. In effect, `-allow_tgs_req` sets the `KRB5_KDB_DISALLOW_TGT_BASED` flag on principals in the database.

{-|+}allow_tix
-allow_tix forbids the issuance of any tickets for principals. +allow_tix clears this flag. The default is +allow_tix. In effect, -allow_tix sets the KRB5_KDB_DISALLOW_ALL_TIX flag on principals in the database.

{-|+}needchange
+needchange sets a flag in the attributes field to force a password change; -needchange clears that flag. The default is -needchange. In effect, +needchange sets the KRB5_KDB_REQUIRES_PWCHANGE flag on principals in the database.

{-|+}password_changing_service
+password_changing_service sets a flag in the attributes field marking a principal as a password-change-service principal (a designation that is most often not useful). -password_changing_service clears the flag. That this flag has a long name is intentional. The default is -password_changing_service. In effect, +password_changing_service sets the KRB5_KDB_PWCHANGE_SERVICE flag on principals in the database.

{-|+}requires_hwauth
+requires_hwauth requires principals to preauthenticate using a hardware device before being allowed to kinit(1). This setting sets the KRB5_KDB_REQUIRES_HW_AUTH flag. -requires_hwauth clears this flag.

{-|+}requires_preauth
+requires_preauth requires principals to preauthenticate before being allowed to kinit(1). This setting sets the KRB5_KDB_REQUIRES_PRE_AUTH flag. -requires_preauth clears this flag.

Examples

EXAMPLE 1 Using create

The following is an example of the use of the create command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
create -subtrees o=org -sscope SUB -r ATHENA.MIT.EDU
Password for "cn=admin,o=org":  password entered
Initializing database for realm 'ATHENA.MIT.EDU'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key: master key entered
Re-enter KDC database master key to verify: master key re-enteredjjjjjj
```

EXAMPLE 2 Using modify

The following is an example of the use of the modify command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
modify +requires_preauth -r ATHENA.MIT.EDU
Password for "cn=admin,o=org":  password entered
Password for "cn=admin,o=org":  password entered
```

EXAMPLE 3 Using `view`

The following is an example of the use of the `view` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
view -r ATHENA.MIT.EDU
          Password for "cn=admin,o=org":
                            Realm Name: ATHENA.MIT.EDU
                               Subtree: ou=users,o=org
                               Subtree: ou=servers,o=org
                           SearchScope: ONE
                   Maximum ticket life: 0 days 01:00:00
                Maximum renewable life: 0 days 10:00:00
                          Ticket flags: DISALLOW_FORWARDABLE REQUIRES_PWCHANGE
```

EXAMPLE 4 Using `destroy`

The following is an example of the use of the `destroy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
destroy -r ATHENA.MIT.EDU
Password for "cn=admin,o=org": password entered
Deleting KDC database of 'ATHENA.MIT.EDU', are you sure?
(type 'yes' to confirm)? yes
OK, deleting database of 'ATHENA.MIT.EDU'...
```

EXAMPLE 5 Using `list`

The following is an example of the use of the `list` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu list
Password for "cn=admin,o=org": password entered
Re-enter Password for "cn=admin,o=org": password re-entered
ATHENA.MIT.EDU
OPENLDAP.MIT.EDU
MEDIA-LAB.MIT.EDU
```

EXAMPLE 6 Using `stashsrvpw`

The following is an example of the use of the `stashsrvpw` command.

```
# kdb5_ldap_util stashsrvpw -f \
/home/andrew/conf_keyfile cn=service-kdc,o=org
Password for "cn=service-kdc,o=org": password entered
Re-enter password for "cn=service-kdc,o=org": password re-entered
```

EXAMPLE 7 Using `create_policy`

The following is an example of the use of the `create_policy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
create_policy -r ATHENA.MIT.EDU \
-maxtktlife "1  day" -maxrenewlife "1 week" \
```

EXAMPLE 7 Using `create_policy` *(Continued)*

```
-allow_postdated +needchange -allow_forwardable tktpolicy
Password for "cn=admin,o=org": password entered
```

EXAMPLE 8 Using `modify_policy`

The following is an example of the use of the `modify_policy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
modify_policy -r ATHENA.MIT.EDU \
-maxtktlife "60 minutes" -maxrenewlife "10 hours" \
+allow_postdated -requires_preauth tktpolicy
Password for "cn=admin,o=org": password entered
```

EXAMPLE 9 Using `view_policy`

The following is an example of the use of the `view_policy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
view_policy -r ATHENA.MIT.EDU tktpolicy
Password for "cn=admin,o=org": password entered
            Ticket policy: tktpolicy
      Maximum ticket life: 0 days 01:00:00
   Maximum renewable life: 0 days 10:00:00
             Ticket flags: DISALLOW_FORWARDABLE REQUIRES_PWCHANGE
```

EXAMPLE 10 Using `destroy_policy`

The following is an example of the use of the `destroy_policy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
destroy_policy -r ATHENA.MIT.EDU tktpolicy
Password for "cn=admin,o=org": password entered
This will delete the policy object 'tktpolicy', are you sure?
(type 'yes' to confirm)? yes
** policy object 'tktpolicy' deleted.
```

EXAMPLE 11 Using `list_policy`

The following is an example of the use of the `list_policy` command.

```
# kdb5_ldap_util -D cn=admin,o=org -H ldaps://ldap-server1.mit.edu \
list_policy -r ATHENA.MIT.EDU
Password for "cn=admin,o=org": password entered
tktpolicy
tmppolicy
userpolicy
```

EXAMPLE 12 Using `setsrvpw`

The following is an example of the use of the `setsrvpw` command.

EXAMPLE 12 Using `setsrvpw` *(Continued)*

```
# kdb5_ldap_util setsrvpw -D cn=admin,o=org setsrvpw \
-fileonly -f /home/andrew/conf_keyfile cn=service-kdc,o=org
Password for "cn=admin,o=org": password entered
Password for "cn=service-kdc,o=org": password entered
Re-enter password for "cn=service-kdc,o=org": password re-entered
```

EXAMPLE 13 Using `create_service`

The following is an example of the use of the `create_service` command.

```
# kdb5_ldap_util -D cn=admin,o=org create_service \
-kdc -randpw -f /home/andrew/conf_keyfile cn=service-kdc,o=org
Password for "cn=admin,o=org": password entered
File does not exist. Creating the file /home/andrew/conf_keyfile...
```

EXAMPLE 14 Using `modify_service`

The following is an example of the use of the `modify_service` command.

```
# kdb5_ldap_util -D cn=admin,o=org modify_service \
-realm ATHENA.MIT.EDU cn=service-kdc,o=org
Password for "cn=admin,o=org": password entered
Changing rights for the service object. Please wait ... done
```

EXAMPLE 15 Using `view_service`

The following is an example of the use of the `view_service` command.

```
# kdb5_ldap_util -D cn=admin,o=org view_service \
cn=service-kdc,o=org
Password for "cn=admin,o=org": password entered
                      Service dn: cn=service-kdc,o=org
                    Service type: kdc
               Service host list:
                   Realm DN list: cn=ATHENA.MIT.EDU,cn=Kerberos,cn=Security
```

EXAMPLE 16 Using `destroy_service`

The following is an example of the use of the `destroy_service` command.

```
# kdb5_ldap_util -D cn=admin,o=org destroy_service \
cn=service-kdc,o=org
Password for "cn=admin,o=org": password entered
This will delete the service object 'cn=service-kdc,o=org', are you sure?
(type 'yes' to confirm)? yes
** service object 'cn=service-kdc,o=org' deleted.
```

EXAMPLE 17 Using `list_service`

The following is an example of the use of the `list_service` command.

EXAMPLE 17 Using `list_service` *(Continued)*

```
# kdb5_ldap_util -D cn=admin,o=org list_service
Password for "cn=admin,o=org": password entered
cn=service-kdc,o=org
cn=service-adm,o=org
cn=service-pwd,o=org
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkrbu
Interface Stability	Volatile

See Also

`kinit(1)`, `kadmin(1M)`, `kdc.conf(4)`, `attributes(5)`

Name kdb5_util – Kerberos Database maintenance utility

Synopsis

```
/usr/sbin/kdb5_util  [-d dbname] [-f stashfile_name]
     [-k mkeytype] [-m ] [-M mkeyname] [-P password] [-r realm]
     [-x db_args]... cmd
```

Description The `kdb5_util` utility enables you to create, dump, load, and destroy the Kerberos V5 database. You can also use `kdb5_util` to create a stash file containing the Kerberos database master key.

Options The following options are supported:

`-d` *dbname*
Specify the database name. `.db` is appended to whatever name is specified. You can specify an absolute path. If you do not specify the `-d` option, the default database name is `/var/krb5/principal`.

`-f` *stashfile_name*
Specify the stash file name. You can specify an absolute path.

`-k` *mkeytype*
Specify the master key type. Valid values are `des3-cbc-sha1`, `des-cbc-crc`, `des-cbc-md5`, `des-cbc-raw`, `arcfour-hmac-md5`, `arcfour-hmac-md5-exp`, `aes128-cts-hmac-sha1-96`, and `aes256-cts-hmac-sha1-96`.

`-m`
Enter the master key manually.

`-M` *mkeyname*
Specify the master key name.

`-P` *password*
Use the specified *password* instead of the stash file.

`-r` *realm*
Use *realm* as the default database realm.

`-x` *db_args*
Pass database-specific arguments to `kadmin`. Supported arguments are for LDAP and the `Berkeley-db2` plug-in. These arguments are:

`binddn=`*binddn*
LDAP simple bind DN for authorization on the directory server. Overrides the `ldap_kadmind_dn` parameter setting in `krb5.conf(4)`.

`bindpwd=`*bindpwd*
Bind password.

`dbname=`*name*
For the `Berkeley-db2` plug-in, specifies a name for the Kerberos database.

`nconns=`*num*
Maximum number of server connections.

`port=`*num*
Directory server connection port.

Operands The following operands are supported:

cmd
Specifies whether to create, destroy, dump, or load the database, or to create a stash file.

You can specify the following commands:

`create -s`
Creates the database specified by the `-d` option. You will be prompted for the database master password. If you specify `-s`, a stash file is created as specified by the `-f` option. If you did not specify `-f`, the default stash file name is `/var/krb5/.k5.realm`. If you use the `-f`, `-k`, or `-M` options when you create a database, then you must use the same options when modifying or destroying the database.

`destroy`
Destroys the database specified by the `-d` option.

`stash`
Creates a stash file. If `-f` was not specified, the default stash file name is `/var/krb5/.k5.realm`. You will be prompted for the master database password. This command is useful when you want to generate the stash file from the password.

`dump` [`-old`] [`-b6`] [`-b7`] [`-ov`] [`-verbose`] [`-mkey_convert`] [`-new_mkey_file` *mkey_file*] [`-rev`] [`-recurse`] [*filename* [*principals*...]]
Dumps the current Kerberos and KADM5 database into an ASCII file. By default, the database is dumped in current format, "`kdb5_util load_dumpversion 5`". If *filename* is not specified or is the string "-", the dump is sent to standard output. Options are as follows:

`-old`
Causes the dump to be in the Kerberos 5 Beta 5 and earlier dump format ("`kdb5_edit load_dump version 2.0`").

`-b6`
Causes the dump to be in the Kerberos 5 Beta 6 format ("`kdb5_edit load_dump version 3.0`").

`-b7`
Causes the dump to be in the Kerberos 5 Beta 7 format ("`kdb5_util load_dump version 4`"). This was the dump format produced on releases prior to 1.2.2.

`-ov`
Causes the dump to be in `ovsec_adm_export` format.

`-verbose`
Causes the name of each principal and policy to be displayed as it is dumped.

`-mkey_convert`
Prompts for a new master key. This new master key will be used to re-encrypt the key data in the dumpfile. The key data in the database will not be changed.

`-new_mkey_file` *mkey_file*
The filename of a stash file. The master key in this stash file will be used to re-encrypt the key data in the dumpfile. The key data in the database will not be changed.

`-rev`
Dumps in reverse order. This might recover principals that do not dump normally, in cases where database corruption has occured.

`-recurse`
Causes the dump to walk the database recursively (`btree` only). This might recover principals that do not dump normally, in cases where database corruption has occurred. In cases of such corruption, this option will probably retrieve more principals than will the `-rev` option.

`load` [`-old`] [`-b6`] [`-b7`] [`-ov`] [`-hash`] [`-verbose`] [`-update`] *filename dbname* [*admin_dbname*]
Loads a database dump from *filename* into *dbname*. Unless the `-old` or `-b6` option is specified, the format of the dump file is detected automatically and handled appropriately. Unless the `-update` option is specified, `load` creates a new database containing only the principals in the dump file, overwriting the contents of any existing database. The `-old` option requires the database to be in the Kerberos 5 Beta 5 or earlier format ("`kdb5_edit load_dump version 2.0`").

`-b6`
Requires the database to be in the Kerberos 5 Beta 6 format ("`kdb5_edit load_dump version 3.0`").

`-b7`
Requires the database to be in the Kerberos 5 Beta 7 format ("`kdb5_util load_dump version 4`").

`-ov`
Requires the database to be in `ovsec_adm_import` format. Must be used with the `-update` option.

`-hash`
Requires the database to be stored as a hash. If this option is not specified, the database will be stored as a `btree`. This option is not recommended, as databases stored in hash format are known to corrupt data and lose principals.

`-verbose`
Causes the name of each principal and policy to be displayed as it is dumped.

`-update`
Records from the dump file are added to or updated in the existing database. Otherwise, a new database is created containing only what is in the dump file and the old one is destroyed upon successful completion.

filename
Required argument that specifies a path to a file containing database dump.

dbname
Required argument that overrides the value specified on the command line or overrides the default.

admin_dbname
Optional argument that is derived from *dbname* if not specified.

Examples

EXAMPLE 1 Creating File that Contains Information about Two Principals

The following example creates a file named `slavedata` that contains the information about two principals, `jdb@ACME.COM` and `pak@ACME.COM`.

```
# /usr/krb5/bin/kdb5_util dump -verbose slavedata
jdb@ACME.COM pak@ACME.COM
```

Files

`/var/krb5/principal`
Kerberos principal database.

`/var/krb5/principal.kadm5`
Kerberos administrative database. Contains policy information.

`/var/krb5/principal.kadm5.lock`
Lock file for the Kerberos administrative database. This file works backwards from most other lock files (that is, `kadmin` exits with an error if this file does not exist).

`/var/krb5/principal.ulog`
The update log file for incremental propagation.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkdcu
Interface Stability	Evolving

See Also

`kpasswd(1)`, `gkadmin(1M)`, `kadmin(1M)`, `kadmind(1M)`, `kadmin.local(1M)`, `kdb5_ldap_util(1M)`, `kproplog(1M)`, `kadm5.acl(4)`, `kdc.conf(4)`, `attributes(5)`, `kerberos(5)`

Name kdmconfig – configure or unconfigure keyboard, display, and mouse options for OpenWindows and internationalization

Synopsis

```
kdmconfig

kdmconfig [-fv] [-s hostname] -c | -t | -u | -d filename
```

Description The `kdmconfig` program applies only to the `Xsun` window system environment, which is no longer the default in Solaris x86. If you want to use `Xsun`, you must run `kdmconfig` from the command line, select `Xsun`, and perform the remaining configuration steps.

The `kdmconfig` program configures or unconfigures the `/etc/openwin/server/etc/OWconfig` file with the keyboard, display, and mouse information relevant to a client's machine on x86 based systems for Solaris software. `kdmconfig` can also be used to set up the *display*, *pointer*, and *keyboard* entries in the `bootparams(4)` database on a server machine or the `monitor`, *keyboard*, *display*, and *pointer* keywords in a `sysidcfg(4)` file. `kdmconfig` can only be run as root or with privileges comparable to root. Upon completion of device selection, `kdmconfig` prompts the user to test the configuration, which is done by running the window system.

Options The valid options are:

`-c`
: Run the program in the configuration mode. This mode is used to create or update the `OWconfig` file. When invoked in this way, `kdmconfig` first looks for the relevant configuration information in the `bootparams(4)` databases. It also takes into account the information returned from device probes, unless the `-s` option is also used. The `bootparams(4)` databases available to the client are all of the `/etc/bootparams` files on servers on the same subnet as the client, provided the server machine is running the `bootparamd(1M)` daemon. `kdmconfig` is invoked with the `-c` option when called by `sysidconfig(1M)`

`-d` *filename*
: Set up a `sysidcfg(4)` file. This option displays the same screens as the `-c` option, but the information you specify is saved as `sysidcfg(4)` keywords (`monitor`, *keyboard*, *display*, and *pointer*). This enables you to use a `sysidcfg(4)` file to preconfigure a system's device information and bypass `kdmconfig` during an installation.

  filename is the `sysidcfg(4)` file that is created, and it is created in the directory where `kdmconfig` is being run unless a path is specified. If filename already exists in the specified directory, the keywords are appended to the existing file.

`-f`
: Force screens mode. When this option is invoked, no network probing will be performed. This is helpful when debugging the client's configuration environment. Note that the `-s` option implies the use of `-f`, bypassing network probing when setting up a server.

`-s` *hostname*
Set up the `bootparams(4)` database on this machine for the specified client. This option presents the same screens as it does when run on a client, but instead writes the resulting information to the `/etc/bootparams` file. Also, `-s` implies the use of the `-f` option. That is, the program will always present the screens to the user when invoked this way. This option will reconfigure the `nsswitch.conf(4)` file to look for a `bootparams(4)` database on a local server. This option is only available to the super-user.

`-t`
Run the program in test mode. In this mode, `kdmconfig` will use device probe information to determine whether the `OWconfig` file contains complete and up-to-date information about the keyboard, display, and mouse. If the information is accurate, `kdmconfig` will exit silently. Otherwise, `kdmconfig` will prompt for the super-user password and proceed to a normal editing session (as though it had been run without options).

`-u`
Unconfigure the system, returning it to an "out-of-the-box" state. In this state, the factory default keyboard, mouse, and display are selected as a result of removing the device configuration entries from the `/etc/openwin/server/etc/OWconfig` file. This may result in an unusable configuration for the display server.

`-v`
Enable verbose mode. Normally, `kdmconfig` will not produce any output. This option is helpful for debugging, as it records the different actions taken by `kdmconfig` on `stderr`.

No Options Run without options, `kdmconfig` is used to edit the current configuration. `kdmconfig` uses the information from the `OWconfig` file in addition to information obtained from the `bootparams(4)` file and from device probes. In other respects, it is similar to using the `-c` option of `kdmconfig`.

Files

`/etc/openwin/server/etc/OWconfig`	OpenWindows configuration file
`/etc/bootparams`	contains list of clients that diskless clients use for booting
`/etc/nsswitch.conf`	name service configuration file

x86 Only `/dev/openprom` installed devices and properties

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWos86r

See Also bootparamd(1M), sys-unconfig(1M), sysidconfig(1M), bootparams(4), nsswitch.conf(4), sysidcfg(4), attributes(5)

See also the Xorg(1) and xorg.conf(4) man pages, which are found under /usr/X11/man on some Solaris systems. These man pages are not part of the SunOS man page collection.

Name kernel – UNIX system executable file containing basic operating system services

Synopsis

```
kernel-name [-asrvx] [-m smf_options] [-i altinit]
```

Description The operating system image, or kernel, is the collection of software comprising the image files (`unix` and `genunix`) and the modules loaded at any instant in time. The system will not function without a kernel to control it.

The kernel is loaded by the `boot(1M)` command in a machine-specific way. The kernel may be loaded from disk, CD-ROM, or DVD (`diskfull boot`) or over the network (`diskless boot`). In either case, the directories under `/platform` and `/kernel` must be readable and must contain executable code which is able to perform the required kernel service. If the `-a` flag is given, the user is able to supply different pathnames for the default locations of the kernel and modules. See `boot(1M)` for more information on loading a specific kernel.

The `moddir` variable contains a list of module directories separated by whitespace. moddir can be set in the `/etc/system` file. The minimal default is:

```
/platform/platform-name/kernel /kernel /usr/kernel
```

This default can be supplemented by a specific platform. It is common for many SPARC systems to override the default path with:

```
/platform/platform-name/kernel:/platform/hardware-class-name\
     /kernel:/kernel:/usr/kernel
```

where *platform-name* can be found using the `-i` option of `uname(1)`, and *hardware-class-name* can be found using the `-m` option of `uname(1)`.

The kernel configuration can be controlled using the `/etc/system` file (see `system(4)`).

`genunix` is the platform-independent component of the base kernel.

Options The following options are supported:

`-a`
: Asks the user for configuration information, such as where to find the system file, where to mount `root`, and even override the name of the kernel itself. Default responses will be contained in square brackets ([ ]), and the user may simply enter RETURN to use the default response (note that RETURN is labeled ENTER on some keyboards). To help repair a damaged `/etc/system` file, enter **`/dev/null`** at the prompt that asks for the pathname of the system configuration file. See `system(4)`.

`-i` *altinit*
: Select an alternative executable to be the primordial process. *altinit* must be a valid path to an executable. The default primordial process is `init(1M)`.

`-m` *smf_options*
: The *smf_options* include two categories of options to control booting behavior of the service management facility: recovery options and messages options.

Message options determine the type and amount of messages that smf(5) displays during boot. Service options determine the services which are used to boot the system.

Recovery options

debug
Prints standard per-service output and all `svc.startd` messages to log.

milestone=[milestone]
Boot with some SMF services temporarily disabled, as indicated by *milestone*. *milestone* can be "none", "single-user", "multi-user", "multi-user-server", or "all". See the `milestone` subcommand of svcadm(1M).

Messages options

quiet
Prints standard per-service output and error messages requiring administrative intervention.

verbose
Prints standard per-service output with more informational messages.

`-r`
Reconfiguration boot. The system will probe all attached hardware devices and configure the logical namespace in `/dev`. See add_drv(1M) and rem_drv(1M) for additional information about maintaining device drivers.

`-s`
Boots only to init level 's'. See init(1M).

`-v`
Boots with verbose messages enabled. If this flag is not given, the messages are still printed, but the output is directed to the system logfile. See syslogd(1M).

`-x`
Does not boot in clustered mode. This option only has an effect when a version of Sun Cluster software that supports this option has been installed.

Examples See boot(1M) for examples and instructions on how to boot.

Files `/kernel`
Contains kernel components common to all platforms within a particular instruction set that are needed for booting the system. of the core image file.

`/platform/`*platform-name*`/kernel`
The platform-specific kernel components.

`/platform/`*hardware-class-name*`/kernel`
The kernel components specific to this hardware class.

`/usr/kernel`
Contains kernel components common to all platforms within a particular instruction set.

The directories in this section can potentially contain the following subdirectories:

drv
Loadable device drivers

exec
The modules that execute programs stored in various file formats.

fs
File system modules

misc
Miscellaneous system-related modules

sched
Operating system schedulers

strmod
System V STREAMS loadable modules

sys
Loadable system calls

SPARC cpu
Processor specific modules

tod
Time-Of-Day hardware interface modules

As only 64-bit SPARC platforms are supported, all SPARC executable modules are contained within `sparcv9` directories in the directories listed above.

x86 mach
x86 hardware support

Modules comprising the 32-bit x86 kernel are contained in the above directories, with the 64-bit x86 kernel components contained within `amd64` subdirectories.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcar, SUNWcarx

See Also

`uname(1)`, `isainfo(1)`, `add_drv(1M)`, `boot(1M)`, `init(1M)`, `kadb(1M)`, `rem_drv(1M)`, `savecore(1M)`, `svc.startd(1M)`, `svcadm(1M)`, `syslogd(1M)`, `system(4)`, `attributes(5)`, `smf(5)`, `devfs(7FS)`

SPARC Only `monitor(1M)`

Diagnostics The kernel gives various warnings and error messages. If the kernel detects an unrecoverable fault, it will panic or halt.

Notes Reconfiguration boot will, by design, not remove `/dev` entries for some classes of devices that have been physically removed from the system.

Name keyserv – server for storing private encryption keys

Synopsis `keyserv [-c] [-d | -e] [-D] [-n] [-s` *sizespec*`]`

Description `keyserv` is a daemon that is used for storing the private encryption keys of each user logged into the system. These encryption keys are used for accessing secure network services such as secure NFS and NIS+.

Normally, root's key is read from the file `/etc/.rootkey` when the daemon is started. This is useful during power-fail reboots when no one is around to type a password.

`keyserv` does not start up if the system does not have a secure `rpc` domain configured. Set up the domain name by using the `/usr/bin/domainname` command. Usually the `svc:/system/identity:domain` service reads the domain from `/etc/defaultdomain`. Invoking the `domainname` command without arguments tells you if you have a domain set up.

The `/etc/default/keyserv` file contains the following default parameter settings. See Files.

`ENABLE_NOBODY_KEYS` Specifies whether default keys for `nobody` are used. `ENABLE_NOBODY_KEYS=NO` is equivalent to the `-d` command-line option. The default value for `ENABLE_NOBODY_KEYS` is `YES`.

Options The following options are supported:

`-c` Do not use disk caches. This option overrides any `-s` option.

`-D` Run in debugging mode and log all requests to `keyserv`.

`-d` Disable the use of default keys for `nobody`. See Files.

`-e` Enable the use of default keys for `nobody`. This is the default behavior. See Files.

`-n` Root's secret key is not read from `/etc/.rootkey`. Instead, `keyserv` prompts the user for the password to decrypt root's key stored in the `publickey` database and then stores the decrypted key in `/etc/.rootkey` for future use. This option is useful if the `/etc/.rootkey` file ever gets out of date or corrupted.

`-s` *sizespec* Specify the size of the extended Diffie-Hellman common key disk caches. The *sizespec* can be one of the following forms:

mechtype`=size` `size` is an integer specifying the maximum number of entries in the cache, or an integer immediately followed by the letter *M*, denoting the maximum size in MB.

`size` This form of *sizespec* applies to all caches.

See `nisauthconf(1M)` for mechanism types. Note that the `des` mechanism, `AUTH_DES`, does not use a disk cache.

Files `/etc/.rootkey`

`/etc/default/keyserv` Contains default settings. You can use command-line options to override these settings.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `keylogin(1)`, `svcs(1)`, `keylogout(1)`, `nisauthconf(1M)`, `svcadm(1M)`, `publickey(4)`, `attributes(5)`, `smf(5)`

http://www.sun.com/directory/nisplus/transition.html

Notes NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

The `keyserv` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/keyserv:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name killall – kill all active processes

Synopsis

```
/usr/sbin/killall [signal]
```

Description `killall` is used by `shutdown(1M)` to kill all active processes not directly related to the shutdown procedure.

`killall` terminates all processes with open files so that the mounted file systems will be unbusied and can be unmounted.

`killall` sends *signal* (see `kill(1)`) to the active processes. If no *signal* is specified, a default of `15` is used.

The `killall` command can be run only by the super-user.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `kill(1)`, `ps(1)`, `fuser(1M)`, `shutdown(1M)`, `signal(3C)`, `attributes(5)`

Name kprop – Kerberos database propagation program

Synopsis

```
/usr/lib/krb5/kprop [-d] [-f file] [-p port-number]
     [-r realm] [-s keytab] [host]
```

Description `kprop` is a command-line utility used for propagating a Kerberos database from a master KDC to a slave KDC. This command must be run on the master KDC. See the *Solaris System Administration Guide, Vol. 6* on how to set up periodic propagation between the master KDC and slave KDCs.

To propagate a Kerberos database, the following conditions must be met:

- The slave KDCs must have an `/etc/krb5/kpropd.acl` file that contains the principals for the master KDC and all the slave KDCs.
- A keytab containing a host principal entry must exist on each slave KDC.
- The database to be propagated must be dumped to a file using `kdb5_util(1M)`.

Options The following options are supported:

`-d`
Enable debug mode. Default is debug mode disabled.

`-f` *file*
File to be sent to the slave KDC. Default is the `/var/krb5/slave_datatrans` file.

`-p` *port-number*
Propagate *port-number*. Default is port `754`.

`-r` *realm*
Realm where propagation will occur. Default *realm* is the local realm.

`-s` *keytab*
Location of the keytab. Default location is `/etc/krb5/krb5.keytab`.

Operands The following operands are supported:

host
Name of the slave KDC.

Examples **EXAMPLE 1** Propagating the Kerberos Database

The following example propagates the Kerberos database from the `/tmp/slave_data` file to the slave KDC `london`. The machine `london` must have a host principal keytab entry and the `kpropd.acl` file must contain an entry for the all the KDCs.

```
# kprop -f /tmp/slave_data london
```

Files `/etc/krb5/kpropd.acl`
List of principals of all the KDCs; resides on each slave KDC.

`/etc/krb5/krb5.keytab`
Keytab for Kerberos clients.

`/var/krb5/slave_datatrans`
Kerberos database propagated to the KDC slaves.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkdcu

See Also `kpasswd(1)`, `svcs(1)`, `gkadmin(1M)`, `inetadm(1M)`, `inetd(1M)`, `kadmind(1M)`, `kadmin.local(1M)`, `kdb5_util(1M)`, `svcadm(1M)`, `kadm5.acl(4)`, `kdc.conf(4)`, `attributes(5)`, `kerberos(5)`

System Administration Guide: Security Services

Name kpropd – Kerberos propagation daemon for slave KDCs

Synopsis

```
/usr/lib/krb5/kpropd [-d] [-f temp_dbfile] [-F dbfile]
     [-p kdb_util] [-P port_number] [-r realm]
     [-s srv_tabfile] [-S] [-a acl_file]
```

Description The `kpropd` command runs on the slave KDC server. It listens for update requests made by `kprop(1M)` from the master KDC and periodically requests incremental updates from the master KDC.

When the slave receives a kprop request from the master, `kpropd` copies principal data to a temporary text file. Next, `kpropd` invokes `kdb5_util(1M)` (unless a different database utility is selected) to load the text file in database format.

When the slave periodically requests incremental updates, `kpropd` update its `principal.ulog` file with any updates from the master. `kproplog(1M)` can be used to view a summary of the update entry log on the slave KDC.

kpropd is not configured for incremental database propagation by default. These settings can be changed in the `kdc.conf(4)` file:

`sunw_dbprop_enable = [true | false]`
Enables or disables incremental database propagation. Default is `false`.

`sunw_dbprop_slave_poll = N[s, m, h]`
Specifies how often the slave KDC polls for any updates that the master might have. Default is `2m` (two minutes).

The `kiprop/`<*hostname*>`@`<*REALM*> principal must exist in the slave's `keytab` file to enable the master to authenticate incremental propagation requests from the slave. In this syntax, <*hostname*> is the slave KDC's host name and <*REALM*> is the realm in which the slave KDC resides.

Options The following options are supported:

`-d`
Enable debug mode. Default is debug mode disabled.

`-f` *temp_dbfile*
The location of the slave's temporary principal database file. Default is `/var/krb5/from_master`.

`-F` *dbfile*
The location of the slave's principal database file. Default is `/var/krb5/principal`.

`-p` *kdb_util*
The location of the Kerberos database utility used for loading principal databases. Default is `/usr/sbin/kdb5_util`.

`-P` *port_number*
Specifies the port number on which `kpropd` will listen. Default is 754 (service name: `krb5_prop`).

`-r` *realm*
Specifies from which Kerberos realm kpropd will receive information. Default is specified in `/etc/krb5/krb5.conf`.

`-s` *srv_tabfile*
The location of the service table file used to authenticate the `kpropd` daemon.

`-S`
Run the daemon in standalone mode, instead of having `inetd` listen for requests. Default is non-standalone mode.

`-a` *acl_file*
The location of the `kpropd`'s access control list to verify if this server can run the `kpropd` daemon. The file contains a list of principal name(s) that will be receiving updates. Default is `/etc/krb5/kpropd.acl`.

Files

`/var/krb5/principal`
Kerberos principal database.

`/var/krb5/principal.ulog`
The update log file.

`/etc/krb5/kdc.conf`
KDC configuration information.

`/etc/krb5/kpropd.acl`
List of principals of all the KDCs; resides on each slave KDC.

`/var/krb5/from_master`
Temporary file used by kpropd before loading this to the principal database.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkdcu
Interface Stability	Evolving

See Also

`kdb5_util(1M)`, `kprop(1M)`, `kproplog(1M)`, `kdc.conf(4)`, `krb5.conf(4)`, `attributes(5)`, `kerberos(5)`

Name kproplog – display the contents of the Kerberos principal update log

Synopsis

```
/usr/sbin/kproplog [-h | -e num]
```

Description The `kproplog` displays the contents of the Kerberos principal update log to standard output. This command can be used to keep track of the incremental updates to the principal database, which is enabled by default. The `/var/krb5/principal.ulog` file contains the update log maintained by the `kadmind`(1M) process on the master KDC server and the `kpropd`(1M) process on the slave KDC servers. When updates occur, they are logged to this file. Subsequently any KDC slave configured for incremental updates will request the current data from the master KDC and update their `principal.ulog` file with any updates returned.

The `kproplog` command can only be run on a KDC server by someone with privileges comparable to the superuser. It will display update entries for that server only.

If no options are specified, the summary of the update log is displayed. If invoked on the master, all of the update entries are also displayed. When invoked on a slave KDC server, only a summary of the updates are displayed, which includes the serial number of the last update received and the associated time stamp of the last update.

Options The following options are supported:

`-h` Display a summary of the update log. This information includes the database version number, state of the database, the number of updates in the log, the time stamp of the first and last update, and the version number of the first and last update entry.

`-e` *num* Display the last *num* update entries in the log. This is useful when debugging synchronization between KDC servers.

`-v` Display individual attributes per update. An example of the output generated for one entry:

```
Update Entry
    Update serial # : 4
    Update operation : Add
    Update principal : test@EXAMPLE.COM
    Update size : 424
    Update committed : True
    Update time stamp : Fri Feb 20 23:37:42 2004
    Attributes changed : 6
          Principal
          Key data
          Password last changed
          Modifying principal
          Modification time
          TL data
```

Files `/var/krb5/principal.ulog` The update log file for incremental propagation.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkdcu
Interface Stability	Evolving

See Also `kpasswd(1)`, `gkadmin(1M)`, `kadmin(1M)`, `kadmind(1M)`, `kdb5_util(1M)`, `kprop(1M)`, `kpropd(1M)`, `kadm5.acl(4)`, `kdc.conf(4)`, `attributes(5)`, `kerberos(5)`

Name krb5kdc – KDC daemon

Synopsis

```
/usr/lib/krb5/krb5kdc [-d dbpath] [-r realm]  [-m]
     [-k masterenctype] [-M masterkeyname]
     [-p port] [-n] [-x db_args]...
```

Description `krb5kdc` is the daemon that runs on the master and slave KDCs to process the Kerberos tickets. For Kerberos to function properly, `krb5kdc` must be running on at least one KDC that the Kerberos clients can access. Prior to running `krb5kdc`, you must initialize the Kerberos database using `kdb5_util(1M)`. See the *System Administration Guide: Security Services* for information regarding how to set up KDCs and initialize the Kerberos database.

Options The following options are supported:

`-d` *dbpath*
Specify the path to the database; default value is `/var/krb5`.

`-k` *masterenctype*
Specify the encryption type for encrypting the database. The default value is `des-cbc-crc`. `des3-cbc-sha1`, `arcfour-hmac-md5`, `arcfour-hmac-md5-exp`, `aes128-cts-hmac-sha1-96`, and `aes256-cts-hmac-sha1-96` are also valid.

`-m`
Specify that the master key for the database is to be entered manually.

`-M` *masterkeyname*
Specify the principal to retrieve the master Key for the database.

`-n`
Specify that `krb5kdc` should not detach from the terminal.

`-p` *port*
Specify the port that will be used by the KDC to listen for incoming requests.

`-r` *realm*
Specify the realm name; default is the local realm name.

`-x` *db_args*
Pass database-specific arguments to `kadmin`. Supported arguments are for the LDAP plug-in. These arguments are:

`binddn=`*binddn*
Specifies the DN of the object used by the KDC server to bind to the LDAP server. This object should have the rights to read the realm container, principal container and the subtree that is referenced by the realm. Overrides the `ldap_kdc_dn` parameter setting in `krb5.conf(4)`.

`bindpwd=`*bindpwd*
Specifies the password for the above-mentioned *binddn*. It is recommended not to use this option. Instead, the password can be stashed using the `stashsrvpw` command of `kdb5_ldap_util(1M)`.

`nconns=`*num*
Specifies the number of connections to be maintained per LDAP server.

`host=`*ldapuri*
Specifies, by an LDAP URI, the LDAP server to which to connect.

Files `/var/krb5/principal.db`
Kerberos principal database.

`/var/krb5/principal.kadm5`
Kerberos administrative database. This file contains policy information.

`/var/krb5/principal.kadm5.lock`
Kerberos administrative database lock file. This file works backwards from most other lock files (that is, `kadmin` will exit with an error if this file does *not* exist).

`/etc/krb5/kdc.conf`
KDC configuration file. This file is read at startup.

`/etc/krb5/kpropd.acl`
File that defines the access control list for propagating the Kerberos database using `kprop`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkdcu

See Also `kill(1)`, `kpasswd(1)`, `gkadmin(1M)`, `kadmind(1M)`, `kadmin.local(1M)`, `kdb5_util(1M)`, `kdb5_ldap_util(1M)`, `logadm(1M)`, `krb5.conf(4)`, `attributes(5)`, `krb5envvar(5)`, `kerberos(5)`,

System Administration Guide: Security Services

Notes The following signal has the specified effect when sent to the server process using the `kill(1)`command:

`SIGHUP`
`krb5kdc` closes and re-opens log files that it directly opens. This can be useful for external log-rotation utilities such as `logadm(1M)`. If this method is used for log file rotation, set the `krb5.conf(4)` `kdc_rotate` period relation to `never`.

Name ksslcfg – enable and configure SMF instance of Kernel SSL

Synopsis

```
ksslcfg create -f pkcs11 -T token_label -C certificate_label
     [-d softtoken_directory]
     [-p password_file [-u username]]
     [-h ca_certchain_file] [-c ciphersuites]
     [-t ssl_session_cache_timeout]
     [-z ssl_session_cache_size] [-v] -x proxy_port [host] ssl_port

ksslcfg create -f pkcs12 -i cert_and_key_pk12file
     [-p password_file [-u username]]
     [-c ciphersuites] [-t ssl_session_cache_timeout]
     [-z ssl_session_cache_size] [-v] -x proxy_port [host] ssl_port

ksslcfg create -f pem -i cert_and_key_pemfile
     [-p password_file [-u username]]
     [-c ciphersuites] [-t ssl_session_cache_timeout]
     [-z ssl_session_cache_size] [-v] -x proxy_port [host] ssl_port

ksslcfg delete [-v] [host] ssl_port

ksslcfg -V

ksslcfg -?
```

Description `ksslcfg` manages `smf(5)` instances for the Kernel SSL proxy module. An SSL-enabled web server can use the services of its Kernel SSL proxy to improve the performance of the HTTPS packets processing. It does so by creating an instance of the Kernel SSL service, specifying the SSL proxy port and parameters, and by listening on the proxy port.

The `create` subcommand creates an instance and enables the service for the given address and SSL port.

The `delete` subcommand disables the service for the given address and port, if it is enabled, and deletes the instance from the SMF repository.

`ksslcfg` can be run as root or by other users assigned to the Network Security profile. See `rbac(5)` and `user_attr(4)`. You must run `ksslcfg` to configure your Kernel SSL proxy before you start your application.

`ksslcfg` allows you to specify an *ssl_port* operand, described under OPERANDS, and, with the `-x` option, a *proxy_port* value. When specified for use with the Kernel SSL proxy, these values cannot also be configured for the Solaris Network Cache and Acceleration (NCA) feature. See `nca(1)` for a description of the NCA feature.

The Fault Managed Resource Identifier (FMRI) for the kernel SSL proxy instances is `svc://network/ssl/proxy`. `ksslcfg` creates an instance of that service unique to the combination of host and SSL port. Instance FMRIs for particular proxy entries can be found with `svcs(1)` and used for dependencies of other services.

Options The following options are supported:

`-c` *ciphersuites*
: Set of ciphers a client is allowed to negotiate in a sorted order. The supported SSL version3 and TLS ciphers are listed below. Note that the names are case-insensitive.

```
rsa_rc4_128_sha
rsa_rc4_128_md5
rsa_3des_ede_cbc_sha
rsa_des_cbc_sha
```

`-f` *key_format*
: Uses the certificate/key format specified in *key_format*. The supported options are `pkcs11`, `pkcs12`, and `pem`.

`-i` *key_and_certificate_file*
: When `pkcs12` or `pem` is specified with the `-f` option, reads a key and a certificate of the web server from *key_and_certificate_file*. This file can also contain any intermediate CA certificates that form the certificate chain to the root CA for the server certificate. These certificates must follow the server certificate in the file and the order must be bottom up: lowest level CA certificate followed by the next higher level CA certificate, and so on.

`-C` *certificate_label*
: PKCS#11 can store multiple certificates in single token. This option enables you to specify a single certificate, identified by *certificate_label*. This label must match the `CKA_LABEL` on the certificate object in the token specified by `-T`. This option is to be used only with `-f pkcs11`.

`-d` *softtoken_directory*
: This option is applicable only with the `pkcs11` key format, when the token label is the Sun Software PKCS#11 softtoken. Use this option to override the default location of the PKCS#11 softtoken directory (`$HOME/.sunw`). See `pkcs11_softtoken(5)`.

`-h` *ca_certchain_file*
: When `pkcs11` is specified with the `-f` option, reads a set of intermediate CA certificates that form the certificate chain to the root CA for the server certificate (specified with the `-C` option), from *ca_certchain_file*. The file must be in PEM format.

`-p` *password_file*
: Obtains the password used to encrypt the private key from *password_file*. When using the `pkcs11` option (see `-f`, above), the password is used to authenticate the user to the PKCS #11 token.

`-t` *ssl_session_cache_timeout*	The timeout value, in seconds, for an SSL session. It corresponds to `SSL3SessionTimeout` of the Sun ONE web server configuration or `SSLSessionCacheTimeout` of `mod_ssl`.
`-T` *token_label*	When `pkcs11` is specified with `-f`, uses the PKCS#11 token specified in *token_label*. Use `cryptoadm list -v` to display all PKCS#11 tokens available.
`-u` *username*	The username of the user who owns the password file. If omitted, the system will try to read the password file as root.
`-v`	Verbose mode.
`-V`	Displays the version.
`-x` *proxy_port*	The SSL proxy port. The port number is designated exclusively for clear-text HTTP communication between the web server and the kernel SSL proxy module. No external HTTP packets are delivered to this port.
`-z` *ssl_session_cache_size*	The maximum number of SSL sessions that can be cached. It corresponds to `SSLCacheEntries` of the Sun ONE web server configuration. When this option is not specified, the default is 5000 entries.
`-?`	Displays the usage of the command.

Operands

[*host*] [*ssl_port*]	The address and the port of the web server for which the kernel SSL entry is created. If *host* is omitted, the entry will be used for all requests that arrived at the *ssl_port*, regardless of the destination address. Both a host name and an IP address are acceptable forms for *host*. *ssl_port* is required. Typically, this has a value of 443.

Examples

EXAMPLE 1 Create and Enable a Kernel SSL Instance

The following command creates and enables a Kernel SSL instance using a certificate and a key in PKCS#11 format.

```
# ksslcfg create -f pkcs11 -T "Sun Software PKCS#11 softtoken"  \
-C "Server-Cert" -p /some/directory/password -u webservd \
-x 8080 www.mysite.com 443

% svcs svc:/network/ssl/proxy
STATE          STIME    FMRI
online         Sep_27   svc:/network/ssl/proxy:kssl-www-mysite-com-443
```

EXAMPLE 2 Create and Enable a Default Instance for All Addresses

The following command creates and enables a default instance for all addresses from a certicate and key in a `pkcs#12` file.

```
# ksslcfg create -x 8888 -f pkcs12 -i /some/directory/keypair.p12 \
    -p /some/directory/password -u webservd 443
```

EXAMPLE 3 Create and Enable an Instance with Specific Cipher Suites

The following command creates and enables an instance with specific cipher suites.

```
# ksslcfg create -x 8080 -f pem \
-i /some/directory/keypair.pem -p /some/directory/password \
-c "rsa_rc4_128_md5,rsa_rc4_128_sha" \
209.249.116.195 443
```

EXAMPLE 4 Disable and Delete an Instance

The following command disables and deletes an instance.

```
# ksslcfg delete www.mysite.com 443
```

Exit Status

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWksslu
Interface Stability	See below.

Command line options are Evolving; command output is Unstable. The FMRI service name (`svc://network/ssl/proxy`) is Unstable, as is the FMRI instance's name format. The utility name is Stable.

See Also

`nca(1)`, `svcprop(1)`, `svcs(1)`, `cryptoadm(1M)`, `svcadm(1M)`, `svccfg(1M)`, `user_attr(4)`, `attributes(5)`, `pkcs11_softtoken(5)`, `rbac(5)`, `smf(5)`

Notes

`ksslcfg create` without an host argument creates an `INADDR_ANY` `smf` instance. `ksslcfg delete` without an host argument deletes only the `INADDR_ANY` instance. `ksslcfg delete` needs a host argument to delete any non-`INADDR_ANY` instance.

On a system with `zones(5)` installed, the `ksslcfg` command can be used only in the global zone at this time.

Name kstat – display kernel statistics

Synopsis

```
kstat [-lpq] [-T u | d ] [-c class] [-m module]
     [-i instance] [-n name] [-s statistic]
     [interval [count]]

kstat [-lpq] [-T u | d ] [-c class]
     [module:instance:name:statistic]...
     [interval [count]]
```

Description The `kstat` utility examines the available kernel statistics, or kstats, on the system and reports those statistics which match the criteria specified on the command line. Each matching statistic is printed with its module, instance, and name fields, as well as its actual value.

Kernel statistics may be published by various kernel subsystems, such as drivers or loadable modules; each kstat has a module field that denotes its publisher. Since each module might have countable entities (such as multiple disks associated with the `sd`(7D) driver) for which it wishes to report statistics, the kstat also has an instance field to index the statistics for each entity; kstat instances are numbered starting from zero. Finally, the kstat is given a name unique within its module.

Each kstat may be a special kstat type, an array of name-value pairs, or raw data. In the name-value case, each reported value is given a label, which we refer to as the statistic. Known raw and special kstats are given statistic labels for each of their values by `kstat`; thus, all published values can be referenced as *module*:*instance*:*name*:*statistic*.

When invoked without any module operands or options, kstat will match all defined statistics on the system. Example invocations are provided below. All times are displayed as fractional seconds since system boot.

Options The tests specified by the following options are logically ANDed, and all matching kstats will be selected. A regular expression containing shell metacharacters must be protected from the shell by enclosing it with the appropriate quotes.

The argument for the `-c`, `-i`, `-m`, `-n`, and `-s` options may be specified as a shell glob pattern, or a Perl regular expression enclosed in '/' characters.

`-c` *class* Displays only kstats that match the specified class. *class* is a kernel-defined string which classifies the “type” of the kstat.

`-i` *instance* Displays only kstats that match the specified instance.

`-l` Lists matching kstat names without displaying values.

`-m` *module* Displays only kstats that match the specified module.

`-n` *name* Displays only kstats that match the specified name.

`-p` Displays output in parseable format. All example output in this document is given in this format. If this option is not specified, `kstat` produces output in a human-readable, table format.

`-q`	Displays no output, but return appropriate exit status for matches against given criteria.
`-s` *statistic*	Displays only kstats that match the specified statistic.
`-T d | u`	Displays a time stamp before each statistics block, either in `ctime(3C)` format ('d') or as an alphanumeric representation of the value returned by `time(2)` ('u').

Operands The following operands are supported:

module:*instance*:*name*:*statistic*	Alternate method of specifying module, instance, name, and statistic as described above. Each of the module, instance, name, or statistic specifiers may be a shell glob pattern or a Perl regular expression enclosed by '/' characters. It is possible to use both specifier types within a single operand. Leaving a specifier empty is equivalent to using the '*' glob pattern for that specifier.
interval	The number of seconds between reports.
count	The number of reports to be printed.

Examples In the following examples, all the command lines in a block produce the same output, as shown immediately below. The exact statistics and values will of course vary from machine to machine.

EXAMPLE 1 Using the `kstat` Command

```
example$ kstat -p -m unix -i 0 -n system_misc -s 'avenrun*'
example$ kstat -p -s 'avenrun*'
example$ kstat -p 'unix:0:system_misc:avenrun*'
example$ kstat -p ':::avenrun*'
example$ kstat -p ':::/^avenrun_\d+min$/'

unix:0:system_misc:avenrun_15min        3
unix:0:system_misc:avenrun_1min 4
unix:0:system_misc:avenrun_5min 2
```

EXAMPLE 2 Using the `kstat` Command

```
example$ kstat -p -m cpu_stat -s 'intr*'
example$ kstat -p cpu_stat:::/^intr/

cpu_stat:0:cpu_stat0:intr       29682330
cpu_stat:0:cpu_stat0:intrblk    87
cpu_stat:0:cpu_stat0:intrthread 15054222
cpu_stat:1:cpu_stat1:intr       426073
cpu_stat:1:cpu_stat1:intrblk    51
```

EXAMPLE 2 Using the kstat Command *(Continued)*

```
cpu_stat:1:cpu_stat1:intrthread 289668
cpu_stat:2:cpu_stat2:intr       134160
cpu_stat:2:cpu_stat2:intrblk    0
cpu_stat:2:cpu_stat2:intrthread 131
cpu_stat:3:cpu_stat3:intr       196566
cpu_stat:3:cpu_stat3:intrblk    30
cpu_stat:3:cpu_stat3:intrthread 59626
```

EXAMPLE 3 Using the kstat Command

```
example$ kstat -p :::state ':::avenrun*'
example$ kstat -p :::state :::/^avenrun/

cpu_info:0:cpu_info0:state      on-line
cpu_info:1:cpu_info1:state      on-line
cpu_info:2:cpu_info2:state      on-line
cpu_info:3:cpu_info3:state      on-line
unix:0:system_misc:avenrun_15min        4
unix:0:system_misc:avenrun_1min 10
unix:0:system_misc:avenrun_5min 3
```

EXAMPLE 4 Using the kstat Command

```
example$ kstat -p 'unix:0:system_misc:avenrun*' 1 3
unix:0:system_misc:avenrun_15min        15
unix:0:system_misc:avenrun_1min 11
unix:0:system_misc:avenrun_5min 21

unix:0:system_misc:avenrun_15min        15
unix:0:system_misc:avenrun_1min 11
unix:0:system_misc:avenrun_5min 21

unix:0:system_misc:avenrun_15min        15
unix:0:system_misc:avenrun_1min 11
unix:0:system_misc:avenrun_5min 21
```

EXAMPLE 5 Using the kstat Command

```
example$ kstat -p -T d 'unix:0:system_misc:avenrun*' 5 2
Thu Jul 22 19:39:50 1999
unix:0:system_misc:avenrun_15min        12
unix:0:system_misc:avenrun_1min 0
unix:0:system_misc:avenrun_5min 11

Thu Jul 22 19:39:55 1999
unix:0:system_misc:avenrun_15min        12
unix:0:system_misc:avenrun_1min 0
```

EXAMPLE 5 Using the `kstat` Command *(Continued)*

```
unix:0:system_misc:avenrun_5min 11
```

EXAMPLE 6 Using the `kstat` Command

```
example$ kstat -p -T u 'unix:0:system_misc:avenrun*'
932668656
unix:0:system_misc:avenrun_15min        14
unix:0:system_misc:avenrun_1min 5
unix:0:system_misc:avenrun_5min 18
```

Exit Status The following exit values are returned:

`0` One or more statistics were matched.

`1` No statistics were matched.

`2` Invalid command line options were specified.

`3` A fatal error occurred.

Files `/dev/kstat` kernel statistics driver

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `sh(1)`, `time(2)`, `ctime(3C)`, `gmatch(3GEN)`, `kstat(3KSTAT)`, `attributes(5)`, `kstat(7D)`, `sd(7D)`, `kstat(9S)`

Notes If the pattern argument contains glob or Perl RE metacharacters which are also shell metacharacters, it will be necessary to enclose the pattern with appropriate shell quotes.

Name ktkt_warnd – Kerberos warning daemon

Synopsis

```
/usr/lib/krb5/ktkt_warnd
```

Description `ktkt_warnd` is a daemon on Kerberos clients that can warn users when their Kerberos tickets are about to expire or renew the tickets before they expire. It is invoked by `inetd` when a ticket-granting ticket (TGT) is obtained for the first time, such as after using the `kinit` command. `ktkt_warnd` can be configured through the `/etc/krb5/warn.conf` file on the client. In `warn.conf`, you can specify that you be supplied notice, through `syslog` or terminal or mail, of ticket expiration or to renew the TGT.

Files `/etc/krb5/warn.conf` Kerberos warning configuration file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `warn.conf(4)`, `attributes(5)`, `kerberos(5)`, `smf(5)`

Notes The `ktkt_warnd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/security/ktkt_warn:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name labeld – Trusted Extensions label daemon

Synopsis

```
/usr/lib/labeld
```

Description The `label` daemon is a system daemon which is started at boot time when Trusted Extensions is enabled. This daemon is started automatically and should not be invoked directly. It does not constitute a programming interface.

To enable Trusted Extensions, enter the following at the command line:

```
% svcadm enable svc:/system/labeld
```

Because Trusted Extensions affects the initialization and operation of zones, all zones must be removed before enabling Trusted Extensions. After enabling, the system should be rebooted to allow Trusted Extensions to come up properly initialized.

Enabling or disabling Trusted Extensions can only be done by a user or role with the `solaris.smf.manage.labels` authorization. (For example, a user or role that has either the Information Security or Object Label Management Rights Profile, or superuser.)

Other configuration steps are needed before using Trusted Extensions functionality. For more information, see the *Solaris Trusted Extensions Installation and Configuration Guide*.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	Uncommitted

See Also `svcs(1)`, `svcadm(1M)`, `syslogd(1M)`, `attributes(5)`, `smf(5)`

Solaris Trusted Extensions Administrator's Procedures

Notes The `labels(5)` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/labeld:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name labelit – list or provide labels for file systems

Synopsis

```
labelit [-F FSType] [-V] [-o FSType-specific-options] special
     [operands]
```

Description The `labelit` utility is used to write or display labels on unmounted disk file systems.

Options The following options are supported:

`-F` *FSType* Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *special* with an entry in the table. If no matching entry is found, the default file system type specified in `/etc/default/fs` will be used.

`-V` Echo complete command line. This option may be used to verify and validate the command line. Additional information obtained using a `/etc/vfstab` lookup is included in the output. The command is not executed.

`-o` Specify *FSType*-specific options. See the manual page for the `labelit` module specific to the file system type.

Operands The following operands are supported. If no operands are specified, `labelit` will display the value of the labels.

special The disk partition (for example, `/dev/rdsk/c0t3d0s6`). The device may not be on a remote machine.

operands *FSType*-specific operands. Consult the manual page of the *FSType*-specific `labelit` command for detailed descriptions.

Usage See `largefile(5)` for the description of the behavior of `labelit` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status The following exit values are returned:

`0` Write or display of labels was successful.

non-zero An error occurred.

Files

`/etc/vfstab` List of default parameters for each file system.

`/etc/default/fs` Default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example:

```
LOCAL=ufs
```

`LOCAL` The default partition for a command if no *FSType* is specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `labelit_hsfs(1M)`, `labelit_udfs(1M)`, `labelit_ufs(1M)`, `volcopy(1M)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`

Notes This utility may not be supported for all *FSTypes*.

Name labelit_hsfs – provide and print labels for hsfs file systems

Synopsis

```
/usr/sbin/labelit -F hsfs [generic_options]
     [-o specific_options] special
```

Description `labelit` can be used to provide labels for unmounted CD-ROM images (CD-ROMs may not be labeled, as they are read-only media).

generic_options are options supported by the generic `labelit` command.

If no *specific_options* are specified, `labelit` prints the current value of all label fields.

The *special* name should be the physical disk section (for example, `/dev/dsk/c0d0s6`).

Options

`-o` Use one or more of the following *name=value* pairs separated by commas (with no intervening spaces) to specify values for specific label fields. According to the ISO 9660 specification, only certain sets of characters may be used to fill in these labels. Thus, "d-characters" below refers to the characters 'A' through 'Z', the digits '0' through '9', and the '_' (underscore) character. "a-characters" below refers to 'A' through 'Z', '0' through '9', space, and the following characters: !"%&'()*+,-./:;<=>?_.

`absfile=` Abstract file identifier, d-characters, 37 characters maximum.

`applid=` Application identifier, d-characters, 128 characters maximum.

`bibfile=` Bibliographic file identifier, d-characters, 37 characters maximum.

`copyfile=` Copyright file identifier, d-characters, 128 maximum.

`prepid=` Data preparer identifier, d-characters, 128 maximum.

`pubid=` Publisher identifier, d-characters, 128 maximum.

`sysid=` System identifier, a-characters, 32 maximum.

`volid=` Volume identifier, d-characters, 32 maximum.

`volsetid=` Volume set identifier, d-characters, 128 maximum.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `labelit(1M)`, `volcopy(1M)`, `attributes(5)`

Name labelit_udfs – provide and print labels for udf file systems

Synopsis

```
labelit -F  udfs [generic_options] [-o specific_options] special
     [fsname  volume]
```

Description The `labelit` command writes labels on an unmounted disk that contains a universal disk file (udf) system. These labels can be used to identify volumes.

Options The following options are supported:

generic_options Specify *generic_options* supported by the generic `labelit` command. See `labelit(1M)` for descriptions of supported options.

`-o` *specific_options* Specify udfs-file-system-specific options in a comma-separated list with no intervening spaces. The following *specific_options* are available:

`lvinfo1=`*string* Specify information to be inserted in the `LVInfo1` field of the Implementation Use Volume Descriptor. Information in `LVInfo1` is generally used to identify the person creating the file system. The maximum length of the string specified is 35 bytes.

`lvinfo2=`*string* Specify information to be inserted into the `LVInfo2` field of the Implementation Use Volume Descriptor. Information in `LVInfo2` is generally used to identify the organization responsible for creating the file system. The maximum length of the string specified is 35 bytes.

`lvinfo3=`*string* Specify information to be inserted into the `LVInfo3` field of the Implementation Use Volume Descriptor. Information in `LVInfo3` is generally used to identify the contact information for the medium. The maximum length of the string specified is 35 bytes.

Operands The following operands are supported:

special Specify *special* as the physical disk slice, for example, `/dev/rdsk/c0t0d0s6`. The device can not be on a remote machine.

fsname Specify *fsname* as the mount point, (for example, `root`, `u1`, and so forth), of the file system.

volume Specify *volume* as the physical volume name.

If none of the options (*fsname*, *volume*, *specific_options*) is specified, `labelit` prints the current values of *fsname*, *volume*, `LVInfo1`, `LVInfo2` and `LVInfo3`.

Exit Status The following exit values are returned:

0 Successful completion.

non-zero An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWudf

See Also labelit(1M), attributes(5)

Name labelit_ufs – provide and print labels for ufs file systems

Synopsis `labelit -F ufs [`*generic_options*`] special [`*fsname volume*`]`

Description `labelit` is used to write labels on unmounted disk file systems. Such labels may be used to uniquely identify volumes and are used by volume-oriented programs such as `volcopy(1M)`.

Options The following option is supported:

generic_options options supported by the generic `labelit` command. See `labelit(1M)`.

Operands The following operands are supported:

special name should be the physical disk section (for example, `/dev/dsk/c0d0s6`). The device may not be on a remote machine.

fsname represents the mount point (for example, `root`, `u1`, and so on) of the file system.

volume may be used to represent the physical volume name.

If *fsname* and *volume* are not specified, `labelit` prints the current values of these labels. Both *fsname* and *volume* are limited to six or fewer characters.

Exit Status The following exit values are returned:

`0` Write or display of labels was successful.

non-zero An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `labelit(1M)`, `volcopy(1M)`, `attributes(5)`, `ufs(7FS)`

Name ldapaddent – create LDAP entries from corresponding /etc files

Synopsis

```
ldapaddent [-cpv] [-a authenticationMethod] [-b baseDN]
 -D bindDN -w bind_password [-f filename] database

ldapaddent [-cpv] -asasl/GSSAPI [-b baseDN] [-f filename]
 database

ldapaddent -d [-v] [-a authenticationMethod] [-D bindDN]
 [-w bind_password] database
```

Description `ldapaddent` creates entries in LDAP containers from their corresponding `/etc` files. This operation is customized for each of the standard containers that are used in the administration of Solaris systems. The *database* argument specifies the type of the data being processed. Legal values for this type are one of `aliases`, `auto_*`, `bootparams`, `ethers`, `group`, `hosts` (including both IPv4 and IPv6 addresses), `ipnodes` (alias for `hosts`), `netgroup`, `netmasks`, `networks`, `passwd`, `shadow`, `protocols`, `publickey`, `rpc`, and `services`. In addition to the preceding, the *database* argument can be one of the RBAC-related files (see `rbac(5)`):

- `/etc/user_attr`
- `/etc/security/auth_attr`
- `/etc/security/prof_attr`
- `/etc/security/exec_attr`

By default, `ldapaddent` reads from the standard input and adds this data to the LDAP container associated with the database specified on the command line. An input file from which data can be read is specified using the `-f` option.

The entries will be stored in the directory based on the client's configuration, thus the client must be configured to use LDAP naming services. The location where entries are to be written can be overridden by using the `-b` option.

If the entry to be added exists in the directory, the command displays an error and exits, unless the `-c` option is used.

Although, there is a `shadow` database type, there is no corresponding `shadow` container. Both the `shadow` and the `passwd` data is stored in the `people` container itself. Similarly, data from `networks` and `netmasks` databases are stored in the `networks` container.

The `user_attr` and `audit_user` data is stored by default in the `people` container. The `prof_attr` and `exec_attr` data is stored by default in the `SolarisProfAttr` container.

You must add entries from the `passwd` database before you attempt to add entries from the `shadow` database. The addition of a `shadow` entry that does not have a corresponding `passwd` entry will fail.

The `passwd` database must precede both the `user_attr` and `audit_user` databases.

For better performance, the recommended order in which the databases should be loaded is as follows:

- `passwd` database followed by `shadow` database
- `networks` database followed by `netmasks` database
- `bootparams` database followed by `ethers` database

Only the first entry of a given type that is encountered will be added to the LDAP server. The `ldapaddent` command skips any duplicate entries.

Options The `ldapaddent` command supports the following options:

`-a` *authenticationMethod* Specify authentication method. The default value is what has been configured in the profile. The supported authentication methods are:

```
simple
sasl/CRAM-MD5
sasl/DIGEST-MD5
sasl/GSSAPI
tls:simple
tls:sasl/CRAM-MD5
tls:sasl/DIGEST-MD5
```

Selecting `simple` causes passwords to be sent over the network in clear text. Its use is strongly discouraged. Additionally, if the client is configured with a profile which uses no authentication, that is, either the `credentialLevel` attribute is set to `anonymous` or `authenticationMethod` is set to `none`, the user must use this option to provide an authentication method. If the authentication method is `sasl/GSSAPI`, *bindDN* and *bind_password* is not required and the `hosts` and `ipnodes` fields of `/etc/nsswitch.conf` must be configured as:

```
hosts: dns files
ipnodes: dns files
```

See `nsswitch.conf(4)`.

`-b` *baseDN* Create entries in the *baseDN* directory. *baseDN* is not relative to the client's default search base, but rather. it is the actual location where the entries will be created. If this parameter is not specified, the first search descriptor defined for the service or the default container will be used.

`-c` Continue adding entries to the directory even after an error. Entries will not be added if the directory server is not responding or if there is an authentication problem.

`-D` *bindDN*	Create an entry which has write permission to the *baseDN*. When used with `-d` option, this entry only needs read permission.
`-d`	Dump the LDAP container to the standard output in the appropriate format for the given database.
`-f` *filename*	Indicates input file to read in an `/etc/` file format.
`-p`	Process the `password` field when loading password information from a file. By default, the `password` field is ignored because it is usually not valid, as the actual password appears in a `shadow` file.
`-w` *bind_password*	Password to be used for authenticating the *bindDN*. If this parameter is missing, the command will prompt for a password. `NULL` passwords are not supported in LDAP. When you use `-w` *bind_password* to specify the password to be used for authentication, the password is visible to other users of the system by means of the `ps` command, in script files or in shell history.
`-v`	Verbose.

Operands The following operands are supported:

database The name of the database or service name. Supported values are: `aliases`, `auto_*`, `bootparams`, `ethers`, `group`, `hosts` (including IPv6 addresses), `netgroup`, `netmasks`, `networks`, `passwd`, `shadow`, `protocols`, `publickey`, `rpc`, and `services`. Also supported are `auth_attr`, `prof_attr`, `exec_attr`, `user_attr`, and `projects`.

Examples **EXAMPLE 1** Adding Password Entries to the Directory Server

The following example show how to add password entries to the directory server:

```
example# ldapaddent -D "cn=directory manager" -w secret \
      -f /etc/passwd passwd
```

EXAMPLE 2 Adding Group Entries

The following example shows how to add `group` entries to the directory server using `sasl/CRAM-MD5` as the authentication method:

```
example# ldapaddent -D "cn=directory manager" -w secret \
     -a "sasl/CRAM-MD5" -f /etc/group group
```

EXAMPLE 3 Adding `auto_master` Entries

The following example shows how to add `auto_master` entries to the directory server:

EXAMPLE 3 Adding `auto_master` Entries *(Continued)*

```
example# dapaddent -D "cn=directory manager" -w secret \
     -f /etc/auto_master auto_master
```

EXAMPLE 4 Dumping password Entries from the Directory to File

The following examples shows how to dump `password` entries from the directory to a file foo:

```
example# ldapaddent -d passwd > foo
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/var/ldap/ldap_client_file`
`/var/ldap/ldap_client_cred` Files containing the LDAP configuration of the client. These files are not to be modified manually. Their content is not guaranteed to be human readable. Use `ldapclient(1M)` to update these files.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu
Interface Stability	Evolving

See Also

`ldap(1)`, `ldaplist(1)`, `ldapmodify(1)`, `ldapmodrdn(1)`, `ldapsearch(1)`, `idsconfig(1M)`, `ldapclient(1M)`, `suninstall(1M)`, `nsswitch.conf(4)`, `attributes(5)`

System Administration Guide: Security Services

Name ldap_cachemgr – LDAP daemon to manage client configuration for LDAP based Network Information Service lookups

Synopsis

```
/usr/lib/ldap/ldap_cachemgr [-l log-file] [-g]
```

Description The `ldap_cachemgr` daemon is a process that provides an up-to-date configuration cache for LDAP naming services. It is started during multi-user boot.

The `ldap_cachemgr` utility provides caching for all parameters as specified and used by the LDAP naming service clients. The `ldap_cachemgr` utility uses the cache files which are originally created by executing the `ldapclient(1M)` utility, as cold start files. Updates to the cache files take place dynamically if profiles are used to configure the client. See the `init` option to `ldapclient(1M)`.

The `ldap_cachemgr` utility helps improve the performance of the clients that are using LDAP as the Naming service repository. In order for the LDAP naming services to function properly, the `ldap_cachemgr` daemon must be running. `ldap_cachemgr` also improves system security by making the configuration files readable by superuser only.

The cache maintained by this daemon is shared by all the processes that access LDAP Naming information. All processes access this cache through a door call. On startup, `ldap_cachemgr` initializes the cache from the cache files. See `ldapclient(1M)`. Thus, the cache survives machine reboots.

The `ldap_cachemgr` daemon also acts as its own administration tool. If an instance of `ldap_cachemgr` is already running, commands are passed transparently to the running version.

Options The following options are supported:

`-g`
: Print current configuration and statistics to standard output. This is the only option executable without superuser privileges.

`-l` *log-file*
: Cause `ldap_cachemgr` to use a log file other than the default `/var/ldap/cachemgr.log`.

Examples **EXAMPLE 1** Stopping and Restarting the `ldap_cachemgr` Daemon

The following example shows how to stop and to restart the `ldap_cachemgr` daemon.

```
example# svcadm disable network/ldap/client
example# svcadm enable network/ldap/client
```

EXAMPLE 2 Forcing `ldap_cachemgr` to Reread Configuration Files

The following example shows how to force `ldap_cachemgr` to reread the `/var/ldap/ldap_client_file` and `/var/ldap/ldap_client_cred` files

```
example# pkill -HUP ldap_cachemgr
```

Files

`/var/ldap/cachemgr.log`
Default log file.

`/var/ldap/ldap_client_file`
`/var/ldap/ldap_client_cred`
Files containing the LDAP configuration of the client. These files are not to be modified manually. Their content is not guaranteed to be human readable. Use `ldapclient(1M)` to update these files.

Warnings

The `ldap_cachemgr` utility is included in the current Solaris release on an uncommitted basis only. It is subject to change or removal in a future minor release.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also

`ldap(1)`, `ldapadd(1)`, `ldapdelete(1)`, `ldaplist(1)`, `ldapmodify(1)`, `ldapmodrdn(1)`, `ldapsearch(1)`, `pkill(1)`, `svcs(1)`, `idsconfig(1M)`, `ldapaddent(1M)`, `ldapclient(1M)`, `suninstall(1M)`, `svcadm(1M)`, `signal.h(3HEAD)`, `resolv.conf(4)`, `attributes(5)`, `smf(5)`

Notes

The `ldap_cachemgr` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/ldap/client
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name ldapclient – initialize LDAP client machine or output an LDAP client profile in LDIF format

Synopsis

```
/usr/sbin/ldapclient [-v | -q] init [-a profileName=profileName]
     [-a domainName=domain] [-a proxyDN=proxyDN]
     [-a proxyPassword=password]
     [-a enableShadowUpdate=true | false]
     [-a adminDN=adminDN]
     [-a adminPassword=adminPassword]
     [-a certificatePath=path] LDAP_server[:port_number]
     [-j passwdFile] [-y passwdFile]
     [-z adminrPasswdFile] LDAP_server[:port_number]

/usr/sbin/ldapclient [-v | -q] manual [-a attrName=attrVal]

/usr/sbin/ldapclient [-v | -q] mod [-a attrName=attrVal]

/usr/sbin/ldapclient [-v | -q] list

/usr/sbin/ldapclient [-v | -q] uninit

/usr/sbin/ldapclient [-v | -q] genprofile -a profileName=profileName
 [-a attrName=attrVal]
```

Description The `ldapclient` utility can be used to:

- initialize LDAP client machines
- restore the network service environment on LDAP clients
- list the contents of the LDAP client cache in human readable format.

The `init` form of the `ldapclient` utility is used to initialize an LDAP client machine, using a profile stored on an LDAP server specified by `LDAP_server`. The LDAP client will use the attributes in the specified profile to determine the configuration of the LDAP client. Using a configuration profile allows for easy installation of LDAP client and propagation of configuration changes to LDAP clients. The `ldap_cachemgr(1M)` utility will update the LDAP client configuration when its cache expires by reading the profile. For more information on the configuration profile refer to IETF document *A Configuration Schema for LDAP Based Directory User Agents*.

The `manual` form of the `ldapclient` utility is used to initialize an LDAP client machine manually. The LDAP client will use the attributes specified on the command line. Any unspecified attributes will be assigned their default values. At least one server must be specified in the `defaultServerList` or the `preferredServerList` attributes.The `domainName` attribute must be specified if the client's `domainName` is not set.

The `mod` form of the `ldapclient` utility is used to modify the configuration of an LDAP client machine that was setup manually. This option modifies only those LDAP client configuration attributes specified on the command line. The `mod` option should only be used on LDAP clients that were initialized using the `manual` option.

Regardless of which method is used for initialization, if a client is to be configured to use a proxy `credentialLevel`, proxy credentials must be provided using `-a proxyDN=`*proxyD* and

`-a proxyPassword=`*proxyPassword* options. However, if `-a proxyPassword=`*proxyPassword* is not specified, `ldapclient` will prompt for it. Note that `NULL` passwords are not allowed in LDAP. If a self `credentialLevel` is configured, `authenticationMethod` must be `sasl/GSSAPI`.

Similarily, if a client is to be configured to enable shadow information update and use a proxy `credentialLevel`, administrator credentials must be provided using `-a adminDN=`*adminDN* and `-a adminPassword=`*adminPassword*. However, the shadow information update does not need the administrator credentials if a self `credentialLevel` is configured.

If any file is modified during installation, it will be backed up to `/var/ldap/restore`. The files that are typically modified during initialization are:

- `/etc/nsswitch.conf`
- `/etc/defaultdomain` (if it exists)
- `/var/yp/binding/`‘`domainname`‘ (for a NIS(YP) client)
- `/var/nis/NIS_COLD_START` (for a NIS+ client)
- `/var/ldap/ldap_client_file` (for an existing LDAP client)
- `/var/ldap/ldap_client_cred` (for an existing LDAP client)

`ldapclient` does not set up a client to resolve hostnames using DNS. It simply copies `/etc/nsswitch.ldap` to `/etc/nsswitch.conf`. If you prefer to use DNS for host resolution, please refer to the DNS documentation for information on setting up DNS. See `resolv.conf(4)`. If you want to use `sasl/GSSAPI` as the authentication method, you have to use DNS for `hosts` and `ipnodes` resolution.

The `list` form of the `ldapclient` utility is used to list the LDAP client configuration. The output will be human readable. LDAP configuration files are not guaranteed to be human readable. Note that the attributes `proxyDN`, `proxyPassword`, `certificatePath`, `domainName`, `enableShadowUpdate`, `adminDN`, and `adminPassword` are not part of the configuration profile and thus are not permitted.

The `uninit` form of the `ldapclient` utility is used to uninitialize the network service environment, restoring it to the state it was in prior to the last execution of `ldapclient` using `init` or `manual`. The restoration will succeed only if the machine was initialized with the `init` or `manual` form of `ldapclient`, as it uses the backup files created by these options.

The `genprofile` option is used to write an LDIF formatted configuration profile based on the attributes specified on the command line to standard output. This profile can then be loaded into an LDAP server to be used as the client profile, which can be downloaded by means of the `ldapclient init` command. Loading the LDIF formatted profile to the directory server can be done through `ldapadd(1)`, or through any server specific import tool. Note that the attributes `proxyDN`, `proxyPassword`, `certificatePath`, and `domainName` are not part of the configuration profile and thus are not permitted.

You must have superuser privileges to run the `ldapclient` command, except with the `genprofile` option.

To access the information stored in the directory, clients can either authenticate to the directory, or use an unauthenticated connection. The LDAP client is configured to have a credential level of either `anonymous` or `proxy`. In the first case, the client does not authenticate to the directory. In the second case, client authenticates to the directory using a proxy identity for read access, and using a administrator identity for write access if `enableShadowUpdate` is configured. In the third case, client authenticates to the directory using a Kerberos principal that is mapped to an LDAP identity by the LDAP server. Refer to the chapter on implementing security in the *Sun ONE Directory Server Administration Guide* or your appropriate directory server documentation for identity mapping details.

If a client is configured to use an identity, you can configure which authentication method the client will use. The LDAP client supports the following authentication methods:

```
none
simple
sasl/CRAM-MD5
sasl/DIGEST-MD5
sasl/GSSAPI
tls:simple
tls:sasl/CRAM-MD5
tls:sasl/DIGEST-MD5
```

Note that some directory servers may not support all of these authentication methods. For `simple`, be aware that the bind password will be sent in the clear to the LDAP server. For those authentication methods using TLS (transport layer security), the entire session is encrypted. You will need to install the appropriate certificate databases to use TLS.

Commands The following commands are supported:

`init`
Initialize client from a profile on a server.

`manual`
Manually initialize client with the specified attribute values.

`mod`
Modify attribute values in the configuration file after a manual initialization of the client.

`list`
Write the contents of the LDAP client cache to standard output in human readable form.

`uninit`
Uninitialize an LDAP client, assuming that `ldapclient` was used to initialize the client.

`genprofile`
Generate a configuration profile in LDIF format that can then be stored in the directory for clients to use, with the `init` form of this command.

Attributes The following attributes are supported:

`adminDN`
Specify the Bind Distinguished Name for the administrator identity that is used for shadow information update. This option is required if the credential level is `proxy`, and `enableShadowUpdate` is set to `true`. There is no default value.

`adminPassword`
Specify the administrator password. This option is required if the credential level is `proxy`, and `enableShadowUpdate` is set to `true`. There is no default value.

`attributeMap`
Specify a mapping from an attribute defined by a service to an attribute in an alternative schema. This can be used to change the default schema used for a given service. The syntax of `attributeMap` is defined in the profile IETF draft. This option can be specified multiple times. The default value for all services is `NULL`. In the example,

```
attributeMap: passwd:uid=employeeNumber
```

the LDAP client would use the LDAP attribute `employeeNumber` rather than `uid` for the `passwd` service. This is a multivalued attribute.

`authenticationMethod`
Specify the default authentication method used by all services unless overridden by the `serviceAuthenticationMethod` attribute. Multiple values can be specified by using a semicolon-separated list. The default value is `none`. For those services that use `credentialLevel` and `credentialLevel` is `anonymous`, this attribute is ignored. Services such as `pam_ldap` will use this attribute, even if `credentialLevel` is anonymous. The supported authentication methods are described above. If the authenticationMethod is `sasl/GSSAPI`, the `hosts` and `ipnodes` of `/etc/nsswitch.conf` must be configured with DNS support, for example:

```
hosts: dns files
ipnodes: dns files
```

`bindTimeLimit`
The maximum time in seconds that a client should spend performing a bind operation. Set this to a positive integer. The default value is 30.

`certificatePath`
The certificate path for the location of the certificate database. The value is the path where security database files reside. This is used for TLS support, which is specified in the `authenticationMethod` and `serviceAuthenticationMethod` attributes. The default is `/var/ldap`.

`credentialLevel`
Specify the credential level the client should use to contact the directory. The credential levels supported are either `anonymous` or `proxy`. If a `proxy` credential level is specified, then the `authenticationMethod` attribute must be specified to determine the authentication mechanism. Further, if the credential level is `proxy` and at least one of the authentication

methods require a bind DN, the `proxyDN` and `proxyPassword` attribute values must be set. In addition, if `enableShadowUpdate` is set to `true`, the `adminDN` and `adminPassword` values must be set. If a self credential level is specified, the `authenticationMethod` must be `sasl/GSSAPI`.

`defaultSearchBase`
Specify the default search base DN. There is no default. The `serviceSearchDescriptor` attribute can be used to override the `defaultSearchBase` for given services.

`defaultSearchScope=one | sub`
Specify the default search scope for the client's search operations. This default can be overridden for a given service by specifying a `serviceSearchDescriptor`. The default is one level search.

`defaultServerList`
A space separated list of server names or server addresses, either IPv4 or IPv6. If you specify server names, be sure that the LDAP client can resolve the name without the LDAP name service. You must resolve the LDAP servers' names by using either `files` or `dns`. If the LDAP server name cannot be resolved, your naming service will fail.

The port number is optional. If not specified, the default LDAP server port number 389 is used, except when TLS is specified in the authentication method. In this case, the default LDAP server port number is 636.

The format to specify the port number for an IPv6 address is:

```
[ipv6_addr]:port
```

To specify the port number for an IPv4 address, use the following format:

```
ipv4_addr:port
```

If the host name is specified, use the format:

```
host_name:port
```

If you use TLS, the LDAP server's hostname must match the hostname in the TLS certificate. Typically, the hostname in the TLS certificate is a fully qualified domain name. With TLS, the LDAP server host addresses must resolve to the hostnames in the TLS certificate. You must use `files` or `dns` to resolve the host address.

`domainName`
Specify the DNS domain name. This becomes the default domain for the machine. The default is the current domain name. This attribute is only used in client initialization.

`enableShadowUpdate=true | false`
Specify whether the client is allowed to update shadow information. If set to `true` and the credential level is `proxy`, `adminDN` and `adminPassword` must be specified.

`followReferrals=true | false`
Specify the referral setting. A setting of true implies that referrals will be automatically followed and false would result in referrals not being followed. The default is true.

`objectclassMap`
Specify a mapping from an `objectclass` defined by a service to an `objectclass` in an alternative schema. This can be used to change the default schema used for a given service. The syntax of `objectclassMap` is defined in the profile IETF draft. This option can be specified multiple times. The default value for all services is `NULL`. In the example,

```
objectclassMap=passwd:posixAccount=unixAccount
```

the LDAP client would use the LDAP `objectclass` of `unixAccount` rather than the `posixAccount` for the `passwd` service. This is a multivalued attribute.

`preferredServerList`
Specify the space separated list of server names or server addresses, either IPv4 or IPv6, to be contacted before servers specified by the `defaultServerList` attribute. If you specify server names, be sure that the LDAP client can resolve the name without the LDAP name service. You must resolve the LDAP servers' names by using either `files` or `dns`. If the LDAP server name cannot be resolved, your naming service will fail.

The port number is optional. If not specified, the default LDAP server port number 389 is used, except when TLS is specified in the authentication method. In this case, the default LDAP server port number is 636.

The format to specify the port number for an IPv6 address is:

```
[ipv6_addr]:port
```

To specify the port number for an IPv4 address, use the following format:

```
ipv4_addr:port
```

If the host name is specified, use the format:

```
host_name:port
```

If you use TLS, the LDAP server's hostname must match the hostname in the TLS certificate. Typically, the hostname in the TLS certificate is a fully qualified domain name. With TLS, the LDAP server host addresses must resolve to the hostnames in the TLS certificate. You must use `files` or `dns` to resolve the host address.

`profileName`
Specify the profile name. For `ldapclient init`, this attribute is the name of an existing profile which may be downloaded periodically depending on the value of the `profileTTL` attribute. For `ldapclient genprofile`, this is the name of the profile to be generated. The default value is `default`.

`profileTTL`
Specify the TTL value in seconds for the client information. This is only relevant if the machine was initialized with a client profile. If you do not want `ldap_cachemgr(1M)` to attempt to refresh the LDAP client configuration from the LDAP server, set `profileTTL` to 0 (zero). Valid values are either zero 0 (for no expiration) or a positive integer in seconds. The default value is 12 hours.

`proxyDN`
Specify the Bind Distinguished Name for the proxy identity. This option is required if the credential level is `proxy`, and at least one of the authentication methods requires a bind DN. There is no default value.

`proxyPassword`
Specify client proxy password. This option is required if the credential level is `proxy`, and at least one of the authentication methods requires a bind DN. There is no default.

`searchTimeLimit`
Specify maximum number of seconds allowed for an LDAP search operation. The default is 30 seconds. The server may have its own search time limit.

`serviceAuthenticationMethod`
Specify authentication methods to be used by a service in the form *servicename*:`authenticationmethod`, for example:

```
pam_ldap:tls:simple
```

For multiple authentication methods, use a semicolon-separated list. The default value is no service authentication methods, in which case, each service would default to the `authenticationMethod` value. The supported authentications are described above.

Three services support this feature: `passwd-cmd`, `keyserv`, and `pam_ldap`. The `passwd-cmd` service is used to define the authentication method to be used by `passwd(1)` to change the user's password and other attributes. The `keyserv` service is used to identify the authentication method to be used by the `chkey(1)` and `newkey(1M)` utilities. The `pam_ldap` service defines the authentication method to be used for authenticating users when `pam_ldap(5)` is configured. If this attribute is not set for any of these services, the `authenticationMethod` attribute is used to define the authentication method. This is a multivalued attribute.

`serviceCredentialLevel`
Specify credential level to be used by a service. Multiple values can be specified in a space-separated list. The default value for all services is `NULL`. The supported credential levels are: `anonymous` or `proxy`. At present, no service uses this attribute. This is a multivalued attribute.

`serviceSearchDescriptor`
Override the default base DN for LDAP searches for a given service. The format of the descriptors also allow overriding the default search scope and search filter for each service. The syntax of `serviceSearchDescriptor` is defined in the profile IETF draft. The default value for all services is `NULL`. This is a multivalued attribute. In the example,

```
serviceSearchDescriptor=passwd:ou=people,dc=a1,dc=acme,dc=com?one
```

the LDAP client would do a one level search in `ou=people,dc=a1,dc=acme,dc=com` rather than `ou=people,`*defaultSearchBase* for the `passwd` service.

Options The following options are supported:

`-a`
Specify `attrName` and its value.

`-q`
Quiet mode. No output is generated.

`-v`
Verbose output.

`-z` *adminrPasswdFile*
Specify a file containing the password for the `adminDN`. To protect the password, use this option in scripts and place the password in a secure file. This option is mutually exclusive of the `-a` *adminPassword* option.

Operands The following operand is supported:

LDAP_server
An address or a name for the LDAP server from which the profile will be loaded. The current naming service specified in the `nsswitch.conf` file is used. Once the profile is loaded, the`preferredServerList` and `defaultServerList` specified in the profile are used.

Examples **EXAMPLE 1** Setting Up a Client By Using the Default Profile Stored on a Specified LDAP Server

The following example shows how to set up a client using the default profile stored on the specified LDAP server. This command will only be successful if either the credential level in the profile is set to `anonymous` or the authentication method is set to `none`.

```
example# ldapclient init 172.16.100.1
```

EXAMPLE 2 Setting Up a Client By Using the `simple` Profile Stored on a Specified LDAP Server

The following example shows how to set up a client using the `simple` profile stored on the specified LDAP server. The domainname is set to `xyz.mycompany.com` and the proxyPassword is `secret`.

```
example# ldapclient init -a profileName=simple \
-a domainName=xyz.mycompany.com \
-a proxyDN=cn=proxyagent,ou=profile,dc=xyz,dc=mycompany,dc=com \
-a proxyPassword=secret '['fe80::a00:20ff:fea3:388']':386
```

EXAMPLE 3 Setting Up a Client Using Only One Server

The following example shows how to set up a client using only one server. The authentication method is set to `none`, and the search base is `dc=mycompany,dc=com`.

```
example# ldapclient manual -a authenticationMethod=none \
-a defaultSearchBase=dc=mycompany,dc=com \
-a defaultServerList=172.16.100.1
```

EXAMPLE 4 Setting Up a Client Using Only One Server That Does Not Follow Referrals

The following example shows how to set up a client using only one server. The credential level is set to `proxy`. The authentication method of is `sasl/CRAM-MD5`, with the option not to follow referrals. The domain name is `xyz.mycompany.com`, and the LDAP server is running on port number 386 at IP address `172.16.100.1`.

```
example# ldapclient manual \
-a credentialLevel=proxy \
-a authenticationMethod=sasl/CRAM-MD5 \
-a proxyPassword=secret \
-a proxyDN=cn=proxyagent,ou=profile,dc=xyz,dc=mycompany,dc=com \
-a defaultSearchBase=dc=xyz,dc=mycompany,dc=com \
-a domainName=xyz.mycompany.com \
-a followReferrals=false \
-a defaultServerList=172.16.100.1:386
```

EXAMPLE 5 Using `genprofile` to Set Only the `defaultSearchBase` and the Server Addresses

The following example shows how to use the `genprofile` command to set the `defaultSearchBase` and the server addresses.

```
example# ldapclient genprofile -a profileName=myprofile \
-a defaultSearchBase=dc=eng,dc=sun,dc=com \
-a "defaultServerList=172.16.100.1 172.16.234.15:386" \
> myprofile.ldif
```

EXAMPLE 6 Creating a Profile on IPv6 servers

The following example creates a profile on IPv6 servers

```
example# ldapclient genprofile -a profileName=eng \
-a credentialLevel=proxy \
-a authenticationMethod=sasl/DIGEST-MD5 \
-a defaultSearchBase=dc=eng,dc=acme,dc=com \
-a "serviceSearchDescriptor=passwd:ou=people,dc=a1,dc=acme,dc=com?one"\
-a preferredServerList= '['fe80::a00:20ff:fea3:388']' \
-a "defaultServerList='['fec0::111:a00:20ff:fea3:edcf']' \
    '['fec0::111:a00:20ff:feb5:e41']'" > eng.ldif
```

EXAMPLE 7 Creating a Profile That Overrides Every Default Value

The following example shows a profile that overrides every default value.

```
example# ldapclient genprofile -a profileName=eng \
-a credentialLevel=proxy -a authenticationMethod=sasl/DIGEST-MD5 \
-a bindTimeLimit=20 \
-a defaultSearchBase=dc=eng,dc=acme,dc=com \
-a "serviceSearchDescriptor=passwd:ou=people,dc=a1,dc=acme,dc=com?one"\
-a serviceAuthenticationMethod=pam_ldap:tls:simple \
-a defaultSearchScope=sub \
```

EXAMPLE 7 Creating a Profile That Overrides Every Default Value *(Continued)*

```
-a attributeMap=passwd:uid=employeeNumber \
-a objectclassMap=passwd:posixAccount=unixAccount \
-a followReferrals=false -a profileTTL=6000 \
-a preferredServerList=172.16.100.30 -a searchTimeLimit=30 \
-a "defaultServerList=172.16.200.1 172.16.100.1 192.168.5.6" > eng.ldif
```

Exit Status

The following exit values are returned:

0 The command successfully executed.

1 An error occurred. An error message is output.

2 `proxyDN` and `proxyPassword` attributes are required, but they are not provided.

Files

`/var/ldap/ldap_client_cred`
`/var/ldap/ldap_client_file`
Contain the LDAP configuration of the client. These files are not to be modified manually. Their content is not guaranteed to be human readable. Use `ldapclient` to update them.

`/etc/defaultdomain`
System default domain name, matching the domain name of the data in the LDAP servers. See `defaultdomain(4)`.

`/etc/nsswitch.conf`
Configuration file for the name-service switch. See `nsswitch.conf(4)`.

`/etc/nsswitch.ldap`
Sample configuration file for the name-service switch configured with LDAP and files.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu
Interface Stability	Evolving

See Also

`chkey(1)`, `ldap(1)`, `ldapadd(1)`, `ldapdelete(1)`, `ldaplist(1)`, `ldapmodify(1)`, `ldapmodrdn(1)`, `ldapsearch(1)`, `idsconfig(1M)`, `ldapaddent(1M)`, `ldap_cachemgr(1M)`, `suninstall(1M)`, `defaultdomain(4)`, `nsswitch.conf(4)`, `resolv.conf(4)`, `attributes(5)`

Name ldmad – Logical Domains Agents daemon

Synopsis

```
/usr/lib/ldoms/ldmad
```

Description The `ldmad` daemon is part of the framework that enables Logical Domain agents to run on a Logical Domain. A Logical Domain agent is a component that interacts with the control domain to provide features or information.

`ldmad` is responsible for running agents on a Logical Domain. `ldmad` must be enabled to ensure the proper functionality of all features provided by the domain manager on the control domain. The daemon is started at boot time and has no configuration options.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWldomu
Interface Stability	Unstable

See Also `svcs(1)`, `svcadm(1M)`, `syslog(3C)`, `syslog.conf(4)`, `attributes(5)`, `smf(5)`

Logical Domains 1.2 Administration Guide

Errors `ldmad` uses `syslog(3C)` to report status and error messages. Error messages are logged with the `LOG_ERR` and `LOG_NOTICE` priorities. Informational messages are logged with the `LOG_INFO` priority. The default entries in the `/etc/syslog.conf` file specify that all `ldmad` error messages are written to the `/var/adm/messages` log.

Notes The `ldmad` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/ldoms/agents:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name link, unlink – link and unlink files and directories

Synopsis

```
/usr/sbin/link  existing-file  new-file

/usr/xpg4/bin/link  existing-file  new-file

/usr/sbin/unlink file
```

Description The `link` and `unlink` commands link and unlink files and directories. Only super-users can use these commands on directories.

Use `link` to create a new file that points to an existing file. The *existing-file* and *new-file* operands specify the existing file and newly-created files. See `OPERANDS`.

`link` and `unlink` directly invoke the `link(2)` and `unlink(2)` system calls, performing exactly what they are told to do and abandoning all error checking. This differs from the `ln(1)` command. See `ln(1)`.

While linked files and directories can be removed using `unlink`, it is safer to use `rm(1)` and `rmdir(1)` instead. See `rm(1)` and `rmdir(1)`.

/usr/xpg4/bin/link If the existing file being hard linked is itself a symbolic link, then the newly created file (*new-file*) will be a hard link to the file referenced by the symbolic link, not to the symbolic link object itself (*existing-file*).

Operands The following operands are supported:

existing-file Specifies the name of the existing file to be linked.

file Specifies the name of the file to be unlinked.

new-file Specifies the name of newly created (linked) file.

Environment Variables See `environ(5)` for descriptions of the following environment variables that affect the execution of `link`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

/usr/xpg4/bin/link

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxcu4
Interface Stability	Standard

See Also ln(1), rm(1), link(2), unlink(2), attributes(5), environ(5), standards(5)

Name listdgrp – lists members of a device group

Synopsis `/usr/bin/listdgrp` *dgroup*...

Description `listdgrp` displays the members of the device groups specified by the *dgroup* list.

Examples **EXAMPLE 1** An example of `listdgrp`.

The following example lists the devices that belong to group `partitions`:

```
example% listdgrp partitions
         root
         swap
         usr
```

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Command was syntax incorrect, an invalid option used, or an internal error occurred.

`2` A device group table could not be opened for reading.

`3` A device group *dgroup* could not be found in the device group table.

Files `/etc/dgroup.tab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `putdgrp(1M)`, `attributes(5)`

Name listen – network listener daemon

Synopsis `/usr/lib/saf/listen [-m` *devstem*`]` *net_spec*

Description The `listen` process "listens" to a network for service requests, accepts requests when they arrive, and invokes servers in response to those service requests. The network listener process may be used with any connection-oriented network (more precisely, with any connection-oriented transport provider) that conforms to the Transport Layer Interface (TLI) Specification.

The listener internally generates a pathname for the minor device for each connection; it is this pathname that is used in the `utmpx` entry for a service, if one is created. By default, this pathname is the concatenation of the prefix `/dev/`*netspec* with the decimal representation of the minor device number. In either case, the representation of the minor device number will be at least two digits (for example, 05 or 27), or longer when it is necessary to accommodate minor device numbers larger than 99.

Server Invocation When a connection indication is received, the listener creates a new transport endpoint and accepts the connection on that endpoint. Before giving the file descriptor for this new connection to the server, any designated STREAMS modules are pushed and the configuration script is executed, (if one exists). This file descriptor is appropriate for use with either TLI (see `t_sync(3NSL)`) or the sockets interface library.

By default, a new instance of the server is invoked for each connection. When the server is invoked, file descriptor 0 refers to the transport endpoint, and is open for reading and writing. File descriptors 1 and 2 are copies of file descriptor 0; no other file descriptors are open. The service is invoked with the user and group IDs of the user name under which the service was registered with the listener, and with the current directory set to the HOME directory of that user.

Alternatively, a service may be registered so that the listener will pass connections to a standing server process through a FIFO or a named stream, instead of invoking the server anew for each connection. In this case, the connection is passed in the form of a file descriptor that refers to the new transport endpoint. Before the file descriptor is sent to the server, the listener interprets any configuration script registered for that service using `doconfig(3NSL)`, although `doconfig` is invoked with both the `NORUN` and `NOASSIGN` flags. The server receives the file descriptor for the connection in a `strrecvfd` structure using an `I_RECVFD` `ioctl(2)`.

For more details about the listener and its administration, see `nlsadmin(1M)`.

Options `-m`*devstem* The listener will use *devstem* as the prefix for the pathname.

Files `/etc/saf/`*pmtag*`/*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `nlsadmin`(1M), `pmadm`(1M), `sac`(1M), `sacadm`(1M), `ioctl`(2), `doconfig`(3NSL), `nlsgetcall`(3NSL), `nlsprovider`(3NSL), `t_sync`(3NSL), `attributes`(5), `streamio`(7I)

System Administration Guide: Basic Administration

Notes When passing a connection to a standing server, the user and group IDs contained in the `strrecvfd` structure will be those for the listener (that is, they will both be 0); the user name under which the service was registered with the listener is not reflected in these IDs.

When operating multiple instances of the listener on a single transport provider, there is a potential race condition in the binding of addresses during initialization of the listeners, if any of their services have dynamically assigned addresses. This condition would appear as an inability of the listener to bind a static-address service to its otherwise valid address, and would result from a dynamic-address service having been bound to that address by a different instance of the listener.

Name llc2_loop – loopback diagnostics to test the driver, adapter and network.

Synopsis

```
/usr/lib/llc2/llc2_loop2 [-v] ppa
/usr/lib/llc2/llc2_loop3 ppa sap frames
/usr/lib/llc2/llc2_loop3 ppa type frames
/usr/lib/llc2/llc2_loop4 [-v] ppa
```

Description

Loop 2 The `loop2` test sends a NULL XID frame to the broadcast (all 1's) destination MAC address. The source SAP (Service Access Point) value used is 0x04 (SNA's SAP). Therefore, if SNA is running on the system, the `loop2` test will fail. The destination SAP value is the NULL SAP (0x00). This test finds out who is listening and can receive frames sent out from a node. The verbose (`-v`) option displays the MAC address of responding nodes. All possible responders may not be displayed, since the `loop2` test only waits for responses for 2 seconds, but during this time 50-200 nodes may be displayed. The most likely error is:

```
Unexpected DLPI primitive x, expected y.
```

where $x = 5$ and $y = 6$. From `/usr/include/sys/dlpi.h`, the expected return value from one of the DLPI primitives is 6 (`DL_OK_ACK`), but instead a 5 (`DL_ERROR_ACK`) was received. This can occur for two reasons:

- The `loop2` command was issued to a non-existent PPA (Physical Point of Attachment).
- The SAP (0x04) is already in use (for example, the SNA subsystem is up).

Loop 3 The `loop3` test sends 1,495 byte Unnumbered Information (UI) frames to the NULL (all 0's) destination MAC address. This should be used along with data capture either on the local node or another node on the same LAN to verify the transmission of data. The *ppa* argument specifies the adapter on which to run the test. The *ppa* is the relative physical position of the adapter and may be ascertained by viewing the adapter configuration (see `llc2_config(1)`). For Token Ring or Ethernet, specify an even *sap* value from 2 through 254, or, for Ethernet only, any *type* value from 1519 (0x05ef) through 65535 (0xffff). It is advised to pick a value that is easily recognized when the data capture output is viewed. *frames* is the decimal number of 1,495 bytes packets to transmit. The test will only display a message if a failure occurs.

Loop 4 The `loop4` test sends a TEST frame (no information field) to the broadcast (all 1's) destination MAC address. The source SAP value used is 0x04 (SNA's SAP). Therefore, if SNA is running on the system, the `loop4` test will fail. The destination SAP value is the NULL SAP (0x00). This test finds out who is listening and can receive frames sent out from a node. The verbose (`-v`) option displays the MAC address of responding nodes. All possible responders may not be displayed since the `loop4` test only waits for responses for 2 seconds, but during this time 50-200 nodes may be displayed. The `loop4` test displays information similar to the following example if other nodes are listening and respond (verbose mode):

```
-Attaching
    -Binding
```

```
    -Sending TEST
    -Responders
       1-0000c0c12449
       2-08000e142990
       3-08000e142a51
       4-0000c0450044
       5-0000c0199e46
    -Unbinding
-Detaching
5 nodes responding
```

The errors displayed are the same as for `loop2`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWllc

See Also `llc2_config(1)`, `llc2(4)`, `attributes(5)`, `llc2(7D)`

Notes For information about how to start the service, see `llc2(7D)`

Name localeadm – query and configure locales

Synopsis

```
localeadm [-lcst] [-q locale | region] [-h] [-d device]...

localeadm -r locale | region [-t] [-v] [-m] [-R root_path]

localeadm -a locale | region [-t] [-v] [-m] [-d device]...
     [-R root_path]

localeadm -f locale | region [-t] [-v] [-m] [-d device]...
     [-R root_path]

localeadm -h

localeadm -C
```

Description The `localeadm` utility queries and configures Solaris locales through a command line interface.

In query (`-q`) or list (`-l`) modes, `localeadm` displays information about locale packages that are installed on the system or that reside on a particular device or directory.

To make it easier for users to pick out locales, the output from `localeadm` consists of a list of country or region names rather than a list of packages. Users can use the output to determine which locales or regions to add or remove.

When the user specifies a locale or region to add or remove using the name given by the output of the list mode, `localeadm` calculates which locale packages need to be changed and add or remove them. `localeadm` uses `pkgadd(1M)` or `pkgrm(1M)` to add or remove packages.

If the locales changed were Asian locales, then extra processes such as input method server daemons might need to be started before the new locales work properly. Once the locales are installed, the user is prompted to either reboot the machine or manually start the daemons. The user is also given a list of daemons which need to be started.

All locales are part of a set geographic region. A locale is an indivisible part of a region. You cannot have a locale which doesn't exist in a region, or a region without locales. If you choose to add or remove a particular locale, all of the locales in the region to which it belongs will be added or removed. Likewise, if you query a locale, `localeadm` checks the system for the region of which the local is part.

Options The following options are supported:

`-a` *locale* | *region* Add the packages for locale (*locale*) or geographic region *region* to the system.

Specify *locale* or *region* as the short name displayed by the `-l` option. For example, the `-l` option outputs `Australasia (aua)`, therefore, the argument for `-a` is `aua`.

This option requires the `-d` option with arguments. If necessary packages are already installed, `localeadm` does not overwrite them. It simply skips such packages.

If you use the `-a` and `-m` options with a locale that has already been added without desktop translated message packages, it adds the desktop translated message packages for that locale to the system.

Only superusers or others who have been assigned package administration access through role-based access control can use this option. See `rbac(5)` for information on adding and removing packages. See `smc(1M)` for information on setting up and adding users to a `package manager` role.

`-c` Display the locale name with codeset in `locale(1)` format.

Use this option in conjunction with the `-l` option to display the locale name with codeset in the format shown by the `locale(1)` command. For example, it displays `fr_FR.ISO8859-1` as opposed to `french`.

`-C` Enable user to recreate the `Locale_config` file based on the images available to them. This attempts to ensure that any inaccuracies (more likely for non-GA versions) in the bundled configuration file are addressed.

`-d` *device* Install or list locales from packages located in *device*. Specify *device* as a full path name to a directory containing Solaris packages or the path to the top directory of a complete Solaris image. You can also specify *device* as a device alias such as `/cdrom/cdrom0`, a device nickname as defined by `eject(1)`, or an alternative device nickname such as `cdrom`, `dvd` or `dvdrom`. If the packages are to be installed from a series of CDROM images, then multiple images can be specified in a comma separated list. The same device or nickname can be repeated to indicate multiple loadings of different media at the same device.

`-f` Check the `pkgs` modified by a previous add or remove operation to ensure all `pkgs` were added or removed properly. If a `pkg` was incorrectly added due to a `pkgadd` or `pkgrm` failure, the `pkg` is backed out and reinstalled.

Only superusers or others who have been assigned package administration access through role-based access control can use this option. See `rbac(5)` for information on adding and removing packages. See `smc(1M)` for information on setting up and adding users to a `package manager` role.

`-h` Print a short help message. The help message lists all the flags and their usage.

`-l` List all the locales that are installed on the system or available on an install media. The list is sorted by geographic region.

When you specify the `-d` option with `-l`, `localeadm` lists all of the locales or regions available on the device pointed to by the `-d` option arguments.

When you do not specify the `-d` option, `localeadm -l` lists all of the locales or regions installed on the current system.

When you specify the `-t` option with `-l`, `localeadm` lists all of the locales or regions that could possibly be added to the system.

`-m` Deselect translated message packages.

By default, with the `-a` option, `localeadm` adds the translated desktop message packages for the locale or region specified in the `-a` option argument. If you use the `-a` option with the `-m` option, the desktop translated message packages for the locale or region will not be added, thus effectively disabling the desktop translated messages support for that locale or region. If used with the `-r` option, *localeadm* will remove only the translated desktop message packages for the locale or region specified in the `-r` option argument.

If you use the `-m` option with a locale that has already been added without the translated desktop message packages it adds the translated desktop message packages for that locale to the system.

`-q` *locale* | *region* Query the system to see if the locale (*locale*) or geographic region *region* are already installed. The expected input for a locale or region name is the name displayed by the `-l` option.

`-r` *locale* | *region* Remove the packages for locale (*locale*) or geographic region (*region*) from the system.

Specify *locale* or *region* as the short name displayed by the `-l` option. For example, the `-l` option outputs `Australasia (aua)`, therefore, the argument for `-a` is `aua`.

Only superusers or others who have been assigned package administration access through role-based access control can use this option. See `rbac(5)` for information on adding and removing packages. See `smc(1M)` for information on setting up and adding users to a `package manager` role.

`-R` *root_path*
: Define the full path name of a directory to use as the root path. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. You can specify *root_path* when you install to a client from a server.

  Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-s`
: Display only the geographic regions of specific locales or regions.

  Use this option in conjunction with the `-l` option to display listed regions or locales.

`-t`
: Test mode.

  Use this option with `-a`, `-f` or `-r` to list all operations to be done. It will not actually add or remove packages.

  Use the this option with `-l` to list all of the locales or regions that could possibly be added to the system.

`-v`
: Print out messages produced during a `pkgadd` or `pkgrm` command.

  This option works on `localeadm` add and remove commands. It does not work on individual `pkgadd` or `pkgrm` commands. It displays additional information, but only as part of the larger program.

Examples

EXAMPLE 1 Listing All of the Locales and Codesets

The following example lists all of the geographic regions installed on the machine. All locales in the regions are listed by their codesets:

```
example% localeadm -lc
```

EXAMPLE 2 Listing the Regions Available on a Solaris CD or DVD

The following example command checks the `Solaris_10/Product` directory of the CD or DVD mounted on `/cdrom/cdrom0`. It also lists the names of the regions that can be installed from packages in that directory. The `-s` option displays the region names without any locales.

```
example% localeadm -ls -d /cdrom/cdrom0
```

EXAMPLE 3 Querying for a Locale

The following example queries whether the Central European region called `ceu` on the current machine.

```
example% localeadm -q ceu
```

EXAMPLE 4 Removing Western European Locales

The following example removes all packages associated with the Western Europe region from the system, except for those packages needed by other regions.

```
example% localeadm -r weu
```

EXAMPLE 5 Adding Russian Locales

The following example installs the Eastern Europe region, of which Russian locale is a part, from packages located in `/net/sparc_images/export/pkgs`.

```
example# localeadm -a ru_RU -d /net/sparc_images/export/pkgs
```

EXAMPLE 6 Adding the Traditional Chinese Locale

The following example adds the Traditional Chinese region to the system. This differs from the previous example in that Traditional Chinese is installed as a geographic region rather than just a locale. This is the case for all Asian languages, for example, `zh_TW`, `zh_CN`, `zh_HK`, `hi_IN`, `th_TH`, `ko_KR`, `ja`.

```
# localeadm -a zh_TW -d /net/sparc_images/export/pkgs
```

Exit Status The following exit values are returned when you invoke `localeadmin` without the `-q` (query) option:

`0` Successful completion.

`1` An error occurred.

The following exit values are returned when you invoke `localeadmin` with the `-q` (query) option:

`0` Successful search. The locale or region was found.

`1` Unsuccessful search. The locale or region was not found.

`2` An error occurred.

Files `/var/sadm/install/logs/localeadmin_install.`*date*
`/var/sadm/install/logs/localeadmin_uninstall.`*date*
Log files for installation and removal operations.

date is specified in YYYY_MM_DD format. If a particular day has multiple installs, *date* has a period (`.`) followed by a number appended to it, for example, `2003_10_20.1`, `2003_10_20.2`.

`/tmp/locales.list`
File that contains the output of the `-l` option.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWladm
Interface Stability	Evolving

See Also `eject(1)`, `locale(1)`, `pkgadd(1M)`, `pkgrm(1M)`, `smc(1M)`, `attributes(5)`, `rbac(5)`

International Language Environments Guide

Name localectr – customize and build new locales

Synopsis

```
/usr/bin/localectr

/usr/bin/localectr -l locale1,locale2... [-d destination_path]
     [-c compiler_path] [-r 0 | 1 | 2] [-i pkginfo_template]
     [-p pkgname_prefix]

/usr/bin/localectr -h

/usr/bin/localectr -q

/usr/bin/localectr -V
```

Description The `localectr` utility allows new locales to be customized and built. The output of `localectr` is an installable package containing the compiled shared object binary which contains the locale data information as well as a number of other supporting files that are required to have a fully working locale on the system.

Once generated, the package can be added to the system by using the `pkgadd(1M)` command and removed with `pkgrm(1M)`.

Depending on your default system login, you might have to reset your user environment after you add a locale. If `dtlogin(1X)` (for the CDE user environment) is the default system login, you need to restart `dtlogin`. No action is required if `gdm(1)` (for the Gnome user environment) is the default login.

There are two interfaces to `localectr`, command line (CLI) and graphical user interface (GUI). To customize the locale data, you must use the GUI. To create locales with standard locale data according to Unicode's Common Locale Data Repository (CLDR), the CLI is sufficient.

With the CLI it is also possible to generate several locales in a single step, with a separate package being generated for each locale. With the GUI, a single locale is processed at a time.

To launch the GUI, run the `localectr` command with no options. To run from the CLI, use the appropriate options as described below.

`localectr` uses the `localedef(1)` utility to build the locale data binary shared object. Therefore, access to a C compiler is required in order to run `localectr` successfully, as this is also required by `localedef`.

`localectr` is mainly concerned with locale data. However, in order to create a fully working locale on the system with `localectr`, many features, such as fonts, translations and input methods, are also required. Depending on what locales are already installed on the system, the relevant features might or might not be present on the system. If they are not present, then `localeadm(1M)` should be used to add the relevant packages before adding packages created by `localectr`. `localectr` bundles locale data for the latest set of locales available in Unicode's CLDR. A user can also create a locale not available in CLDR by supplying her own data in the `localectr` GUI.

Options The following options are supported:

`-c` *compiler_path*
Specify the path to the C compiler that is used to compile the locale data into a shared object binary. Not required if the compiler is already in the user's `PATH`.

`-d` *destination_path*
Specify the path to the directory where the created package is to be stored.

`-h`
Displays the usage message.

`-i` *template_file*
The full path to user defined `pkginfo(4)` template file.

`-l` *locale1,locale2...*
Specify a comma-separated list of locale(s) to generate. Locale names are in the form: *locale.codeset@variant*, where *codeset* and *variant* are optional. The default and only allowed *codeset* is `UTF-8`. The default *variant* is `localectr`.

`-p`
The package name prefix

`-q`
Queries `localectr` for a complete list of locales for which locale data is defined in `localectr`. When `localectr` is run from the CLI, the locale(s) specified with the `-l` option must be on this list in order for an installable locale package to be generated.

`-r`
Specify the range of Unicode characters for which locale data rules in the `LC_CTYPE` and `LC_COLLATE` categories are to be generated. There are three valid options:

`0`
Locale data rules are restricted to the exemplar or to commonly used characters of the locale in question.

`1`
Locale data rules are restricted to the Unicode plane 0 characters, whose codepoints fall in the range `u0000-uFFFF`.

`2`
Locale data rules are generated for all codepoints defined in the latest version of Unicode that is supported by the system on which `localectr` is being run.

`-V`
Shows the version of this software.

Examples **EXAMPLE 1** Launching the GUI

The following example launches the `localectr` GUI.

```
example% localectr
```

EXAMPLE 2 Generating Locale for Afrikaans (South Africa) with Default Locale Data

The following example generates a package in the specified destination directory, which can be used to install the Afrikaans (`af_ZA.UTF-8`) locale on the system. The package name is composed of a prefix followed by the hyphen separated ISO-639 language code, the ISO-3166 country code, the locale encoding and an optional user-defined tag. The resulting package can then be added to the system using `pkgadd(1M)`.

```
example% localectr -l af_ZA -d /tmp
```

EXAMPLE 3 Generating Several South Africa Locales with Full Unicode Range of Characters

The following example generates an installable package for each of the specified locales.

```
example% localectr -l af_ZA,en_ZA,xh_ZA,zu_ZA -d /tmp -r 2
```

EXAMPLE 4 Generating the Irish Locale with a User-Specified Tag

The following example will generate a locale whose full name is `ga_IE.UTF-8@mycompanyname`.

```
example% localectr -l ga_IE@mycompanyname -d /tmp
```

Exit Status

The following exit codes are returned:

`0`
: Successful completion

`>0`
: An error occurred.

Files

`/usr/bin/localectr`
: Wrapper script that launches locale creator.

`/usr/lib/localectr`
: Jar files, scripts, and locale data repository needed to run the application.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlocalecreator
Interface Stability	See below.

Interface stability is Committed for command-line options and is Uncommitted for other interfaces.

See Also `locale(1)`, `localedef(1)`, `localeadm(1M)`, `pkgadd(1M)`, `pkgrm(1M)`, `pkginfo(4)`, `attributes(5)`

`dtlogin`(1X), `gdm`(1)(these are not SunOS man pages)

International Language Environments Guide

Name locator – location indicator control

Synopsis

```
/usr/sbin/locator [-f | -n]
```

Description The locator command sets or queries the state of the system locator if such a device exists.

Without options, the locator command reports the current state of the system.

The privileges required to use this command are hardware dependent. Typically, only the super user can get or set a locator.

Options The following options are supported:

`-f` Turns the locator off.

`-n` Turns the locator on.

Examples **EXAMPLE 1** Using the `locator` Command on a Platform Which Has a System Locator LED

When issued on a platform which has a system locator LED, the following command turns the locator on:

```
# locator -n
# locator
The 'system' locator is on
```

EXAMPLE 2 Using the `locator` Command on a Platform Which Does Not Have a System Locator LED

When issued on a platform which does not have a system locator LED, the following command attempts to turn the locator on. The command returns an error message.

```
# locator -n
'system' locator not found
```

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Invalid command line input.

`2` The requested operation failed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `attributes(5)`

Name lockd – network lock daemon

Synopsis

```
/usr/lib/nfs/lockd [-g graceperiod] [-l listen_min_backlog]
     [-t timeout] [nthreads]
```

Description The `lockd` utility is part of the NFS lock manager, which supports record locking operations on NFS files. See `fcntl(2)` and `lockf(3C)`. The lock manager provides the following two functions:

- It forwards `fcntl(2)` locking requests for NFS mounted file systems to the lock manager on the NFS server.
- It generates local file locking operations in response to requests forwarded from lock managers running on NFS client machines.

State information kept by the lock manager about these locking requests can be lost if the `lockd` is killed or the operating system is rebooted. Some of this information can be recovered as follows. When the server lock manager restarts, it waits for a grace period for all client-site lock managers to submit reclaim requests. Client-site lock managers, on the other hand, are notified by the status monitor daemon, `statd(1M)`, of the restart and promptly resubmit previously granted lock requests. If the lock daemon fails to secure a previously granted lock at the server site, then it sends `SIGLOST` to a process.

Administrators can make changes to the startup parameters for `lockd` by logging in as root and editing the `/etc/default/nfs` file (See `nfs(4)`).

Options The following options are supported:

`-g` *graceperiod*	Deprecated in favor of `GRACE_PERIOD`. Specify the number of seconds that all clients (both NLM and NFSv4) have to reclaim locks after the server reboots. It also controls the NFSv4 lease interval. This option is equivalent to the `LOCKD_GRACE_PERIOD` parameter.
`-l` *listen_min_backlog*	Specify the listener backlog (*listen_min_backlog*). *listen_min_backlog* is the number connect requests that are queued and waiting to be processed before new connect requests start to get dropped.
`-t` *timeout*	Specify the number of seconds to wait before retransmitting a lock request to the remote server. The default value is 5 seconds. Equivalent of the `LOCKD_RETRANSMIT_TIMEOUT` parameter in the `nfs` file.

Operands

nthreads	Specify the maximum number of concurrent threads that the server can handle. This concurrency is achieved by up to *nthreads* threads created as needed in the kernel. *nthreads* should be based on the load expected on this server. If *nthreads* is not specified, the maximum number of concurrent threads will default to 20. Equivalent of the `LOCKD_SERVERS` parameter in the `nfs` file.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfscu

See Also svcs(1), automountd(1M), clear_locks(1M), mount_nfs(1M), share(1M), share_nfs(1M), statd(1M), svcadm(1M), fcntl(2), lockf(3C), nfs(4), attributes(5), smf(5)

Notes The lockd service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/nfs/nlockmgr
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

If it is disabled, it will be enabled by mount_nfs(1M), share_nfs(1M), and automountd(1M) unless its application/auto_enable property is set to false.

This daemon might not exist in a future release of Solaris.

Name lockfs – change or report file system locks

Synopsis

```
/usr/sbin/lockfs [-adefhnuw] [-c string] [file-system]...
```

Description `lockfs` is used to change and report the status of file system locks. `lockfs` reports the lock status and unlocks the file systems that were improperly left locked.

Using `lockfs` to lock a file system is discouraged because this requires extensive knowledge of SunOS internals to be used effectively and correctly.

When invoked with no arguments, `lockfs` lists the UFS file systems that are locked. If *file-system* is not specified, and `-a` is specified, `lockfs` is run on all mounted, UFS type file systems.

Options The options are mutually exclusive: `wndheuf`. If you do specify more than one of these options on a `lockfs` command line, the utility does not protest and invokes only the last option specified. In particular, you cannot specify a flush (`-f`) and a lock (for example, `-w`) on the same command line. However, all locking operations implicitly perform a flush, so the `-f` is superfluous when specifying a lock.

You must be super-user to use any of the following options, with the exception of `-a`, `-f` and `-v`.

The following options are supported.

`-a`
Apply command to all mounted, UFS type file systems. *file-system* is ignored when `-a` is specified.

`-c` *string*
Accept a string that is passed as the comment field. The `-c` only takes affect when the lock is being set using the `-d`, `-h`, `-n`, `-u`, or `-w` options.

`-d`
Delete-lock (`dlock`) the specified *file-system*. dlock suspends access that could remove directory entries.

`-e`
Error-lock (`elock`) the specified *file-system*. elock blocks all local access to the locked file system and returns `EWOULDBLOCK` on all remote access. File systems are elocked by UFS on detection of internal inconsistency. They may only be unlocked after successful repair by `fsck`, which is usually done automatically (see `mount_ufs(1M)`). elocked file systems can be unmounted.

`-f`
Force a synchronous flush of all data that is dirty at the time `fsflush` is run to its backing store for the named file system (or for all file systems.)

It is a more reliable method than using sync(1M) because it does not return until all possible data has been pushed. In the case of UFS filesystems with logging enabled, the log is also rolled before returning. Additional data can be modified by the time `fsflush` exits, so using one of the locking options is more likely to be of general use.

`-h`
Hard-lock (`hlock`) the specified *file-system*. hlock returns an error on every access to the locked file system, and cannot be unlocked. hlocked file systems can be unmounted.

`-n`
Name-lock (`nlock`) the specified *file-system*. nlock suspends accesses that could change or remove existing directories entries.

`-u`
Unlock (`ulock`) the specified *file-system*. ulock awakens suspended accesses.

`-v`
Enable verbose output.

`-w`
Write-lock (`wlock`) the specified *file-system*. wlock suspends writes that would modify the file system. Access times are not kept while a file system is write-locked.

Operands The following operands are supported.

file-system
A list of path names separated by whitespace. Note that *file-system* can be a directory rather than the specific name of a file system, such as `/` or `/usr`. For example, if you specify `/export/home` as an argument to a `lockfs` command and `/export/home` is mounted on the root (`/`) file system, the `lockfs` command will take effect on the root file system.

Usage See largefile(5) for the description of the behavior of `lockfs` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Using `lockfs -a`

In the following examples, *filesystem* is the pathname of the mounted-on directory (mount point). `Locktype` is one of "`write`," "*name*," "*delete*," "*hard*," or "*unlock*". When enclosed in parenthesis, the lock is being set. `Comment` is a string set by the process that last issued a lock command.

The following example shows the `lockfs` output when only the `-a` option is specified.

```
example#  /usr/sbin/lockfs -a
```

Filesystem	Locktype	Comment
/	unlock	

EXAMPLE 1 Using `lockfs -a` *(Continued)*

/var	unlock

```
example#
```

EXAMPLE 2 Using `lockfs -w`

The following example shows the `lockfs` output when the `-w` option is used to write lock the `/var` file system and the comment string is set using the `-c` option. The `-a` option is then specified on a separate command line.

```
example#  /usr/sbin/lockfs -w -c "lockfs: write lock example" /var
example#  /usr/sbin/lockfs -a
```

Filesystem	Locktype	Comment
/	unlock	
/var	write	lockfs: write lock example

```
example#
```

EXAMPLE 3 Using `lockfs -u`

The following example shows the `lockfs` output when the `-u` option is used to unlock the `/var` file system and the comment string is set using the `-c` option.

```
example#  /usr/sbin/lockfs -uc "lockfs: unlock example" /var
example#  /usr/sbin/lockfs /var
```

Filesystem	Locktype	Comment
/var	unlock	lockfs: unlock example

```
example#
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`kill(1)`, `mount_ufs(1M)`, `sync(1M)`, `attributes(5)`, `largefile(5)`, `ufs(7FS)`,

System Administration Guide: Basic Administration

Diagnostics *file system*`: Not owner`
You must be root to use this command.

file system `:Deadlock condition detected/avoided`
A file is enabled for accounting or swapping, on *file system*.

file system`: Device busy`
Another process is setting the lock on *file system*.

Name lockstat – report kernel lock and profiling statistics

Synopsis

```
lockstat [-ACEHI] [-e event_list] [-i rate]
     [-b | -t | -h | -s depth] [-n nrecords]
     [-l lock [, size]] [-d duration]
     [-f function [, size]] [-T] [-ckgwWRpP] [-D count]
     [-o filename] [-x opt [=val]] command [args]
```

Description The `lockstat` utility gathers and displays kernel locking and profiling statistics. `lockstat` allows you to specify which events to watch (for example, spin on adaptive mutex, block on read access to rwlock due to waiting writers, and so forth) how much data to gather for each event, and how to display the data. By default, `lockstat` monitors all lock contention events, gathers frequency and timing data about those events, and displays the data in decreasing frequency order, so that the most common events appear first.

`lockstat` gathers data until the specified command completes. For example, to gather statistics for a fixed-time interval, use `sleep(1)` as the command, as follows:

```
example# lockstat sleep 5
```

When the `-I` option is specified, `lockstat` establishes a per-processor high-level periodic interrupt source to gather profiling data. The interrupt handler simply generates a `lockstat` event whose caller is the interrupted PC (program counter). The profiling event is just like any other `lockstat` event, so all of the normal `lockstat` options are applicable.

`lockstat` relies on DTrace to modify the running kernel's text to intercept events of interest. This imposes a small but measurable overhead on all system activity, so access to `lockstat` is restricted to super-user by default. The system administrator can permit other users to use `lockstat` by granting them additional DTrace privileges. Refer to the *Solaris Dynamic Tracing Guide* for more information about DTrace security features.

Options The following options are supported:

Event Selection If no event selection options are specified, the default is `-C`.

`-A`
Watch all lock events. `-A` is equivalent to `-CH`.

`-C`
Watch contention events.

`-E`
Watch error events.

`-e` *event_list*
Only watch the specified events. *event list* is a comma-separated list of events or ranges of events such as 1,4-7,35. Run `lockstat` with no arguments to get a brief description of all events.

`-H`
Watch hold events.

`-I`
Watch profiling interrupt events.

`-i` *rate*
Interrupt rate (per second) for `-I`. The default is 97 Hz, so that profiling doesn't run in lockstep with the clock interrupt (which runs at 100 Hz).

Data Gathering

`-x` *arg*[=*val*]
Enable or modify a DTrace runtime option or D compiler option. The list of options is found in the *Solaris Dynamic Tracing Guide*. Boolean options are enabled by specifying their name. Options with values are set by separating the option name and value with an equals sign (=).

Data Gathering (Mutually Exclusive)

`-b`
Basic statistics: lock, caller, number of events.

`-h`
Histogram: Timing plus time-distribution histograms.

`-s` *depth*
Stack trace: Histogram plus stack traces up to *depth* frames deep.

`-t`
Timing: Basic plus timing for all events [default].

Data Filtering

`-d` *duration*
Only watch events longer than *duration*.

`-f` *func[,size]*
Only watch events generated by *func*, which can be specified as a symbolic name or hex address. *size* defaults to the ELF symbol size if available, or `1` if not.

`-l` *lock[,size]*
Only watch *lock*, which can be specified as a symbolic name or hex address. `size` defaults to the ELF symbol size or `1` if the symbol size is not available.

`-n` *nrecords*
Maximum number of data records.

`-T`
Trace (rather than sample) events [off by default].

Data Reporting

`-c`
Coalesce lock data for lock arrays (for example, `pse_mutex[]`).

`-D` *count*
Only display the top *count* events of each type.

`-g`
Show total events generated by function. For example, if `foo()` calls `bar()` in a loop, the work done by `bar()` counts as work generated by `foo()` (along with any work done by `foo()` itself). The `-g` option works by counting the total number of stack frames in which each function appears. This implies two things: (1) the data reported by `-g` can be misleading if the stack traces are not deep enough, and (2) functions that are called recursively might show greater than 100% activity. In light of issue (1), the default data gathering mode when using `-g` is `-s 50`.

`-k`
Coalesce PCs within functions.

`-o` *filename*
Direct output to *filename*.

`-P`
Sort data by (*count* * *time*) product.

`-p`
Parsable output format.

`-R`
Display rates (events per second) rather than counts.

`-W`
Whichever: distinguish events only by caller, not by lock.

`-w`
Wherever: distinguish events only by lock, not by caller.

Display Formats The following headers appear over various columns of data.

`Count` or `ops/s`
Number of times this event occurred, or the rate (times per second) if `-R` was specified.

`indv`
Percentage of all events represented by this individual event.

`genr`
Percentage of all events generated by this function.

`cuml`
Cumulative percentage; a running total of the individuals.

`rcnt`
Average reference count. This will always be `1` for exclusive locks (mutexes, spin locks, rwlocks held as writer) but can be greater than `1` for shared locks (rwlocks held as reader).

`nsec`
Average duration of the events in nanoseconds, as appropriate for the event. For the profiling event, duration means interrupt latency.

`Lock`
Address of the lock; displayed symbolically if possible.

`CPU+PIL`
CPU plus processor interrupt level (PIL). For example, if CPU 4 is interrupted while at PIL 6, this will be reported as `cpu[4]+6`.

`Caller`
Address of the caller; displayed symbolically if possible.

Examples

EXAMPLE 1 Measuring Kernel Lock Contention

```
example# lockstat sleep 5
Adaptive mutex spin: 2210 events in 5.055 seconds (437 events/sec)

Count indv cuml rcnt     nsec Lock                 Caller
-------------------------------------------------------------------------
  269  12%  12% 1.00     2160 service_queue        background+0xdc
  249  11%  23% 1.00       86 service_queue        qenable_locked+0x64
  228  10%  34% 1.00      131 service_queue        background+0x15c
   68   3%  37% 1.00       79 0x30000024070        untimeout+0x1c
   59   3%  40% 1.00      384 0x300066fa8e0        background+0xb0
   43   2%  41% 1.00       30 rqcred_lock          svc_getreq+0x3c
   42   2%  43% 1.00      341 0x30006834eb8        background+0xb0
   41   2%  45% 1.00      135 0x30000021058        untimeout+0x1c
   40   2%  47% 1.00       39 rqcred_lock          svc_getreq+0x260
   37   2%  49% 1.00     2372 0x300068e83d0        hmestart+0x1c4
   36   2%  50% 1.00       77 0x30000021058        timeout_common+0x4
   36   2%  52% 1.00      354 0x300066fa120        background+0xb0
   32   1%  53% 1.00       97 0x30000024070        timeout_common+0x4
   31   1%  55% 1.00     2923 0x300069883d0        hmestart+0x1c4
   29   1%  56% 1.00      366 0x300066fb290        background+0xb0
   28   1%  57% 1.00      117 0x3000001e040        untimeout+0x1c
   25   1%  59% 1.00       93 0x3000001e040        timeout_common+0x4
   22   1%  60% 1.00       25 0x30005161110        sync_stream_buf+0xdc
   21   1%  60% 1.00      291 0x30006834eb8        putq+0xa4
   19   1%  61% 1.00       43 0x3000515dcb0        mdf_alloc+0xc
   18   1%  62% 1.00      456 0x30006834eb8        qenable+0x8
   18   1%  63% 1.00       61 service_queue        queuerun+0x168
   17   1%  64% 1.00      268 0x30005418ee8        vmem_free+0x3c
[...]

R/W reader blocked by writer: 76 events in 5.055 seconds (15 events/sec)

Count indv cuml rcnt     nsec Lock                 Caller
-------------------------------------------------------------------------
   23  30%  30% 1.00 22590137 0x300098ba358        ufs_dirlook+0xd0
   17  22%  53% 1.00  5820995 0x3000ad815e8        find_bp+0x10
   13  17%  70% 1.00  2639918 0x300098ba360        ufs_iget+0x198
    4   5%  75% 1.00  3193015 0x300098ba360        ufs_getattr+0x54
```

EXAMPLE 1 Measuring Kernel Lock Contention *(Continued)*

```
    3   4%  79% 1.00  7953418 0x3000ad817c0       find_bp+0x10
    3   4%  83% 1.00   935211 0x3000ad815e8       find_read_lof+0x14
    2   3%  86% 1.00 16357310 0x300073a4720       find_bp+0x10
    2   3%  88% 1.00  2072433 0x300073a4720       find_read_lof+0x14
    2   3%  91% 1.00  1606153 0x300073a4370       find_bp+0x10
    1   1%  92% 1.00  2656909 0x300107e7400       ufs_iget+0x198
[...]
```

EXAMPLE 2 Measuring Hold Times

```
example# lockstat -H -D 10 sleep 1
Adaptive mutex spin: 513 events

Count indv cuml rcnt     nsec Lock                 Caller
-------------------------------------------------------------------------------
  480   5%   5% 1.00     1136 0x300007718e8       putnext+0x40
  286   3%   9% 1.00      666 0x3000077b430       getf+0xd8
  271   3%  12% 1.00      537 0x3000077b430       msgio32+0x2fc
  270   3%  15% 1.00     3670 0x300007718e8       strgetmsg+0x3d4
  270   3%  18% 1.00     1016 0x300007c38b0       getq_noenab+0x200
  264   3%  20% 1.00     1649 0x300007718e8       strgetmsg+0xa70
  216   2%  23% 1.00     6251 tcp_mi_lock         tcp_snmp_get+0xfc
  206   2%  25% 1.00      602 thread_free_lock    clock+0x250
  138   2%  27% 1.00      485 0x300007c3998       putnext+0xb8
  138   2%  28% 1.00     3706 0x300007718e8       strrput+0x5b8
-------------------------------------------------------------------------------
[...]
```

EXAMPLE 3 Measuring Hold Times for Stack Traces Containing a Specific Function

```
example# lockstat -H -f tcp_rput_data -s 50 -D 10 sleep 1
Adaptive mutex spin: 11 events in 1.023 seconds (11
events/sec)

-------------------------------------------------------------------------------
Count indv cuml rcnt     nsec Lock                   Caller
    9  82%  82% 1.00     2540 0x30000031380          tcp_rput_data+0x2b90

      nsec ------ Time Distribution ------ count     Stack
       256 |@@@@@@@@@@@@@@@@@              5         tcp_rput_data+0x2b90
       512 |@@@@@@                         2         putnext+0x78
      1024 |@@@                            1         ip_rput+0xec4
      2048 |                               0         _c_putnext+0x148
      4096 |                               0         hmeread+0x31c
      8192 |                               0         hmeintr+0x36c
     16384 |@@@                            1
sbus_intr_wrapper+0x30
[...]
```

EXAMPLE 3 Measuring Hold Times for Stack Traces Containing a Specific Function *(Continued)*

```
Count indv cuml rcnt     nsec Lock                   Caller
    1   9%  91% 1.00     1036 0x30000055380          freemsg+0x44

      nsec ------ Time Distribution ------ count     Stack
      1024 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 1        freemsg+0x44
                                                     tcp_rput_data+0x2fd0
                                                     putnext+0x78
                                                     ip_rput+0xec4
                                                     _c_putnext+0x148
                                                     hmeread+0x31c
                                                     hmeintr+0x36c

sbus_intr_wrapper+0x30
-------------------------------------------------------------------------------
[...]
```

EXAMPLE 4 Basic Kernel Profiling

For basic profiling, we don't care whether the profiling interrupt sampled `foo()+0x4c` or `foo()+0x78`; we care only that it sampled somewhere in `foo()`, so we use `-k`. The CPU and PIL aren't relevant to basic profiling because we are measuring the system as a whole, not a particular CPU or interrupt level, so we use `-W`.

```
example# lockstat -kIW -D 20 ./polltest
Profiling interrupt: 82 events in 0.424 seconds (194
events/sec)

Count indv cuml rcnt     nsec Hottest CPU+PIL     Caller
-------------------------------------------------------------------------
    8  10%  10% 1.00      698 cpu[1]              utl0
    6   7%  17% 1.00      299 cpu[0]              read
    5   6%  23% 1.00      124 cpu[1]              getf
    4   5%  28% 1.00      327 cpu[0]              fifo_read
    4   5%  33% 1.00      112 cpu[1]              poll
    4   5%  38% 1.00      212 cpu[1]              uiomove
    4   5%  43% 1.00      361 cpu[1]              mutex_tryenter
    3   4%  46% 1.00      682 cpu[0]              write
    3   4%  50% 1.00       89 cpu[0]              pcache_poll
    3   4%  54% 1.00      118 cpu[1]              set_active_fd
    3   4%  57% 1.00      105 cpu[0]              syscall_trap32
    3   4%  61% 1.00      640 cpu[1]              (usermode)
    2   2%  63% 1.00      127 cpu[1]              fifo_poll
    2   2%  66% 1.00      300 cpu[1]              fifo_write
    2   2%  68% 1.00      669 cpu[0]              releasef
    2   2%  71% 1.00      112 cpu[1]              bt_getlowbit
    2   2%  73% 1.00      247 cpu[1]              splx
```

EXAMPLE 4 Basic Kernel Profiling *(Continued)*

```
    2   2%  76% 1.00      503 cpu[0]              mutex_enter
    2   2%  78% 1.00      467 cpu[0]+10           disp_lock_enter
    2   2%  80% 1.00      139 cpu[1]              default_copyin
-------------------------------------------------------------------------
[...]
```

EXAMPLE 5 Generated-load Profiling

In the example above, 5% of the samples were in `poll()`. This tells us how much time was spent inside `poll()` itself, but tells us nothing about how much work was *generated* by `poll()`; that is, how much time we spent in functions called by `poll()`. To determine that, we use the `-g` option. The example below shows that although `polltest` spends only 5% of its time in `poll()` itself, `poll()`-induced work accounts for 34% of the load.

Note that the functions that generate the profiling interrupt (`lockstat_intr()`, `cyclic_fire()`, and so forth) appear in every stack trace, and therefore are considered to have generated 100% of the load. This illustrates an important point: the generated load percentages do *not* add up to 100% because they are not independent. If 72% of all stack traces contain both `foo()` and `bar()`, then both `foo()` and `bar()` are 72% load generators.

```
example# lockstat -kgIW -D 20 ./polltest
Profiling interrupt: 80 events in 0.412 seconds (194 events/sec)

Count genr cuml rcnt     nsec Hottest CPU+PIL     Caller
-------------------------------------------------------------------------
   80 100% ---- 1.00      310 cpu[1]              lockstat_intr
   80 100% ---- 1.00      310 cpu[1]              cyclic_fire
   80 100% ---- 1.00      310 cpu[1]              cbe_level14
   80 100% ---- 1.00      310 cpu[1]              current_thread
   27  34% ---- 1.00      176 cpu[1]              poll
   20  25% ---- 1.00      221 cpu[0]              write
   19  24% ---- 1.00      249 cpu[1]              read
   17  21% ---- 1.00      232 cpu[0]              write32
   17  21% ---- 1.00      207 cpu[1]              pcache_poll
   14  18% ---- 1.00      319 cpu[0]              fifo_write
   13  16% ---- 1.00      214 cpu[1]              read32
   10  12% ---- 1.00      208 cpu[1]              fifo_read
   10  12% ---- 1.00      787 cpu[1]              utl0
    9  11% ---- 1.00      178 cpu[0]              pcacheset_resolve
    9  11% ---- 1.00      262 cpu[0]              uiomove
    7   9% ---- 1.00      506 cpu[1]              (usermode)
    5   6% ---- 1.00      195 cpu[1]              fifo_poll
    5   6% ---- 1.00      136 cpu[1]              syscall_trap32
    4   5% ---- 1.00      139 cpu[0]              releasef
    3   4% ---- 1.00      277 cpu[1]              polllock
-------------------------------------------------------------------------
[...]
```

EXAMPLE 6 Gathering Lock Contention and Profiling Data for a Specific Module

In this example we use the `-f` option not to specify a single function, but rather to specify the entire text space of the `sbus` module. We gather both lock contention and profiling statistics so that contention can be correlated with overall load on the module.

```
example# modinfo | grep sbus
 24 102a8b6f   b8b4  59   1  sbus (SBus (sysio) nexus driver)

example# lockstat -kICE -f 0x102a8b6f,0xb8b4 sleep 10
Adaptive mutex spin: 39 events in 10.042 seconds (4 events/sec)

Count indv cuml rcnt     nsec Lock               Caller
-------------------------------------------------------------------------------
   15  38%  38% 1.00      206 0x30005160528      sync_stream_buf
    7  18%  56% 1.00       14 0x30005160d18      sync_stream_buf
    6  15%  72% 1.00       27 0x300060c3118      sync_stream_buf
    5  13%  85% 1.00       24 0x300060c3510      sync_stream_buf
    2   5%  90% 1.00       29 0x300060c2d20      sync_stream_buf
    2   5%  95% 1.00       24 0x30005161cf8      sync_stream_buf
    1   3%  97% 1.00       21 0x30005161110      sync_stream_buf
    1   3% 100% 1.00       23 0x30005160130      sync_stream_buf
[...]

Adaptive mutex block: 9 events in 10.042 seconds (1 events/sec)

Count indv cuml rcnt     nsec Lock               Caller
-------------------------------------------------------------------------------
    4  44%  44% 1.00   156539 0x30005160528      sync_stream_buf
    2  22%  67% 1.00   763516 0x30005160d18      sync_stream_buf
    1  11%  78% 1.00   462130 0x300060c3510      sync_stream_buf
    1  11%  89% 1.00   288749 0x30005161110      sync_stream_buf
    1  11% 100% 1.00  1015374 0x30005160130      sync_stream_buf
[...]

Profiling interrupt: 229 events in 10.042 seconds (23 events/sec)

Count indv cuml rcnt     nsec Hottest CPU+PIL    Caller

-------------------------------------------------------------------------------
   89  39%  39% 1.00      426 cpu[0]+6           sync_stream_buf
   64  28%  67% 1.00      398 cpu[0]+6           sbus_intr_wrapper
   23  10%  77% 1.00      324 cpu[0]+6           iommu_dvma_kaddr_load
   21   9%  86% 1.00      512 cpu[0]+6           iommu_tlb_flush
   14   6%  92% 1.00      342 cpu[0]+6           iommu_dvma_unload
   13   6%  98% 1.00      306 cpu[1]             iommu_dvma_sync
    5   2% 100% 1.00      389 cpu[1]             iommu_dma_bindhdl
-------------------------------------------------------------------------------
[...]
```

EXAMPLE 7 Determining the Average PIL (processor interrupt level) for a CPU

```
example# lockstat -Iw -l cpu[3] ./testprog

Profiling interrupt: 14791 events in 152.463 seconds (97 events/sec)

Count indv cuml rcnt     nsec CPU+PIL             Hottest Caller

-------------------------------------------------------------------------
13641  92%  92% 1.00      253 cpu[3]              (usermode)
  579   4%  96% 1.00      325 cpu[3]+6            ip_ocsum+0xe8
  375   3%  99% 1.00      411 cpu[3]+10           splx
  154   1% 100% 1.00      527 cpu[3]+4            fas_intr_svc+0x80
   41   0% 100% 1.00      293 cpu[3]+13           send_mondo+0x18
    1   0% 100% 1.00      266 cpu[3]+12           zsa_rxint+0x400
-------------------------------------------------------------------------
[...]
```

EXAMPLE 8 Determining which Subsystem is Causing the System to be Busy

```
example# lockstat -s 10 -I sleep 20

Profiling interrupt: 4863 events in 47.375 seconds (103 events/sec)

Count indv cuml rcnt     nsec CPU+PIL          Caller

-------------------------------------------------------------------------
1929   40%  40% 0.00     3215 cpu[0]           usec_delay+0x78
  nsec ------ Time Distribution ------ count   Stack
  4096 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1872    ata_wait+0x90
  8192 |                                 27      acersb_get_intr_status+0x34
 16384 |                                 29      ata_set_feature+0x124
 32768 |                                 1       ata_disk_start+0x15c
                                                 ata_hba_start+0xbc
                                                 ghd_waitq_process_and \
                                                 _mutex_hold+0x70
                                                 ghd_waitq_process_and \
                                                 _mutex_exit+0x4
                                                 ghd_transport+0x12c
                                                 ata_disk_tran_start+0x108
-------------------------------------------------------------------------
[...]
```

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdtrc

See Also dtrace(1M), plockstat(1M), attributes(5), lockstat(7D), mutex(9F), rwlock(9F)

Solaris Dynamic Tracing Guide

Notes The profiling support provided by `lockstat -I` replaces the old (and undocumented) `/usr/bin/kgmon` and `/dev/profile`.

Tail-call elimination can affect call sites. For example, if `foo()+0x50` calls `bar()` and the last thing `bar()` does is call `mutex_exit()`, the compiler can arrange for `bar()` to branch to `mutex_exit()` with a return address of `foo()+0x58`. Thus, the `mutex_exit()` in `bar()` will appear as though it occurred at `foo()+0x58`.

The PC in the stack frame in which an interrupt occurs can be bogus because, between function calls, the compiler is free to use the return address register for local storage.

When using the `-I` and `-s` options together, the interrupted PC will usually not appear anywhere in the stack since the interrupt handler is entered asynchronously, not by a function call from that PC.

The `lockstat` technology is provided on an as-is basis. The format and content of `lockstat` output reflect the current Solaris kernel implementation and are therefore subject to change in future releases.

Name lofiadm – administer files available as block devices through lofi

Synopsis

```
/usr/sbin/lofiadm -a file [device]

/usr/sbin/lofiadm -d file | device

/usr/sbin/lofiadm [ file | device]
```

Description `lofiadm` administers `lofi(7D)`, the loopback file driver. `lofi`(7D) allows a file to be associated with a block device. That file can then be accessed through the block device. This is useful when the file contains an image of some filesystem (such as a floppy or CD-ROM image), because the block device can then be used with the normal system utilities for mounting, checking or repairing filesystems. See `fsck(1M)` and `mount(1M)`.

Use `lofiadm` to add a file as a loopback device, remove such an association, or print information about the current associations.

The `lofi(7D)` driver is not available and will not work inside a zone.

Options The following options are supported:

`-a` *file* [*device*]
Add *file* as a block device.

If *device* is not specified, an available device is picked.

If *device* is specified, `lofiadm` attempts to assign it to *file*. *device* must be available or `lofiadm` will fail. The ability to specify a device is provided for use in scripts that wish to re-establish a particular set of associations.

`-d` *file* | *device*
Remove an association by *file* or *device* name, if the associated block device is not busy, and deallocates the block device.

Operands The following operands are supported:

file
Print the block device associated with *file*.

device
Print the file name associated with the block device *device*.

Without arguments, print a list of the current associations. Filenames must be valid absolute pathnames.

When a file is added, it is opened for reading or writing by root. Any restrictions apply (such as restricted root access over NFS). The file is held open until the association is removed. It is not actually accessed until the block device is used, so it will never be written to if the block device is only opened read-only.

Examples

EXAMPLE 1 Mounting an Existing CD-ROM Image

You should ensure that Solaris understands the image before creating the CD. `lofi` allows you to mount the image and see if it works.

This example mounts an existing CD-ROM image (`sparc.iso`), of the `Red Hat 6.0 CD` which was downloaded from the Internet. It was created with the `mkisofs` utility from the Internet.

Use `lofiadm` to attach a block device to it:

```
# lofiadm -a /home/mike_s/RH6.0/sparc.iso
/dev/lofi/1
```

`lofiadm` picks the device and prints the device name to the standard output. You can run `lofiadm` again by issuing the following command:

```
# lofiadm
Block Device     File
/dev/lofi/1      /home/mike_s/RH6.0/sparc.iso
```

Or, you can give it one name and ask for the other, by issuing the following command:

```
# lofiadm /dev/lofi/1
/home/mike_s/RH6.0/sparc.iso
```

Use the `mount` command to mount the image:

```
# mount -F hsfs -o ro /dev/lofi/1 /mnt
```

Check to ensure that Solaris understands the image:

```
# df -k /mnt
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/lofi/1           512418  512418       0   100%    /mnt
# ls /mnt
./            RedHat/       doc/          ls-lR         rr_moved/
../           TRANS.TBL     dosutils/     ls-lR.gz      sbin@
.buildlog     bin@          etc@          misc/         tmp/
COPYING       boot/         images/       mnt/          usr@
README        boot.cat*     kernels/      modules/
RPM-PGP-KEY   dev@          lib@          proc/
```

Solaris can mount the CD-ROM image, and understand the filenames. The image was created properly, and you can now create the CD-ROM with confidence.

As a final step, unmount and detach the images:

```
# umount /mnt
# lofiadm -d /dev/lofi/1
# lofiadm
Block Device             File
```

EXAMPLE 2 Mounting a Floppy Image

This is similar to Example 1.

Using `lofi` to help you mount files that contain floppy images is helpful if a floppy disk contains a file that you need, but the machine which you are on does not have a floppy drive. It is also helpful if you do not want to take the time to use the `dd` command to copy the image to a floppy.

This is an example of getting to MDB floppy for Solaris on an x86 platform:

```
# lofiadm -a /export/s28/MDB_s28x_wos/latest/boot.3
/dev/lofi/1
# mount -F pcfs /dev/lofi/1 /mnt
# ls /mnt
./            COMMENT.BAT*  RC.D/         SOLARIS.MAP*
../           IDENT*        REPLACE.BAT*  X/
APPEND.BAT*   MAKEDIR.BAT*  SOLARIS/
# umount /mnt
# lofiadm -d /export/s28/MDB_s28x_wos/latest/boot.3
```

EXAMPLE 3 Making a UFS Filesystem on a File

Making a UFS filesystm on a file can be useful, particularly if a test suite requires a scratch filesystem. It can be painful (or annoying) to have to re-partition a disk just for the test suite, but you do not have to. You can `newfs` a file with `lofi`

Create the file:

```
# mkfile 35m /export/home/test
```

Attach it to a block device. You also get the character device that `newfs` requires, so `newfs` that:

```
# lofiadm -a /export/home/test
/dev/lofi/1
# newfs /dev/rlofi/1
newfs: construct a new file system /dev/rlofi/1: (y/n)? y
/dev/rlofi/1:   71638 sectors in 119 cylinders of 1 tracks, 602 sectors
        35.0MB in 8 cyl groups (16 c/g, 4.70MB/g, 2240 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 9664, 19296, 28928, 38560, 48192, 57824, 67456,
```

Note that `ufs` might not be able to use the entire file. Mount and use the filesystem:

```
# mount /dev/lofi/1 /mnt
# df -k /mnt
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/lofi/1            33455       9   30101     1%    /mnt
# ls /mnt
./           ../          lost+found/
# umount /mnt
```

EXAMPLE 3 Making a UFS Filesystem on a File *(Continued)*

```
# lofiadm -d /dev/lofi/1
```

EXAMPLE 4 Creating a PC (FAT) File System on a Unix File

The following series of commands creates a FAT file system on a Unix file. The file is associated with a block device created by `lofiadm`.

```
# mkfile 10M /export/test/testfs
# lofiadm -a /export/test testfs
/dev/lofi/1
Note use of rlofi, not lofi, in following command.
# mkfs -F pcfs -o nofdisk,size=20480 /dev/rlofi/1
Construct a new FAT file system on /dev/rlofi/1: (y/n)? y
# mount -F pcfs /dev/lofi/1 /mnt
# cd /mnt
# df -k .
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/lofi/1            10142       0   10142     0%    /mnt
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `lofiadm`: `LC_CTYPE`, `LC_MESSAGES` and `NLSPATH`.

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`fsck(1M)`, `mount(1M)`, `mount_ufs(1M)`, `newfs(1M)`, `attributes(5)`, `lofi(7D)`, `lofs(7FS)`

Notes

Just as you would not directly access a disk device that has mounted file systems, you should not access a file associated with a block device except through the `lofi` file driver. It might also be appropriate to ensure that the file has appropriate permissions to prevent such access.

Associations are not persistant across reboots. A script can be used to re-establish them if required.

The abilities of `lofiadm`, and who can use them, are controlled by the permissions of `/dev/lofictl`. Read-access allows query operations, such as listing all the associations. Write-access is required to do any state-changing operations, like adding an association. As

shipped, `/dev/lofictl` is owned by `root`, in group `sys`, and mode `0644`, so all users can do query operations but only root can change anything. The administrator can give users write-access, allowing them to add or delete associations, but that is very likely a security hole and should probably only be given to a trusted group.

When mounting a filesystem image, take care to use appropriate mount options. In particular, the `nosuid` mount option might be appropriate for UFS images whose origin is unknown. Also, some options might not be useful or appropriate, like `logging` or `forcedirectio` for UFS. For compatibility purposes, a raw device is also exported along with the block device. For example, `newfs(1M)` requires one.

The output of `lofiadm` (without arguments) might change in future releases.

Name logadm – manage endlessly growing log files

Synopsis

```
logadm
logadm [-options] logname...
```

Description `logadm` is a general log rotation tool that is suitable for running from `cron(1M)`.

Without arguments, `logadm` reads the `/etc/logadm.conf` file, and, for every entry found in that file, checks the corresponding log file to see if it should be rotated. Typically this check is done each morning by an entry in the root's `crontab(1)`.

If the *logname* argument is specified, `logadm` renames the corresponding log file by adding a suffix so that the most recent log file ends with `.0` (that is, *logfile*`.0`), the next most recent ends with `.1` (that is, *logfile*`.1`), and so forth. By default, ten versions of old log files are kept (that is, *logfile*`.0` through *logfile*`.9`). At the point when what would be the eleventh file is logged,`logadm` automatically deletes the oldest version to keep the count of files at ten.

`logadm` takes a number of *options*. You can specify these options on the command line or in the `/etc/logadm.conf` file. The `logadm` command searches `/etc/logadm.conf` for lines of the form *logname options*

logname
: Identifies an entry in `/etc/logadm.conf`. This can be a name or the pathname of the log file. If you specify a log file, rather than a name, for this field, it must be a fully qualified pathname.

options
: Identifies command line options exactly as they would be entered on the command line. This allows commonly used log rotation policies to be stored in the `/etc/logadm.conf` file. See `EXAMPLES`.

  If *options* are specified both in `/etc/logadm.conf` and on the command line, those in the `/etc/logadm.conf` file are applied first. Therefore, the command line options override those in `/etc/logadm.conf`.

  Log file names specified in `/etc/logadm.conf` can contain filename substitution characters such as * and `?`, that are supported by `csh(1)`.

Two options control when a log file is rotated. They are: `-s` size `-p` period.

When using more than one of these options at a time, there is an implied *and* between them. This means that all conditions must be met before the log is rotated.

If neither of these two options are specified, the default conditions for rotating a log file are: `-s 1b -p 1w`, which means the log file is only rotated if the size is non-zero and if at least 1 week has passed since the last time it was rotated.

By specifying `-p never` as a rotation condition, any other rotation conditions are ignored and `logadm` moves on to the expiration of old log files. By specifying `-p now` as a rotation condition, a log rotation is forced.

Unless specified by the `-o`, `-g`, or `-m` options, `logadm` replaces the log file (after renaming it) by creating an empty file whose owner, group ID, and permissions match the original file.

Three options control when old log files are expired: `-A` age `-C` count `-S`size. These options expire the oldest log files until a particular condition or conditions are met. For example, the combination `-C 5` and the `-S 10m` options expires old log files until there are no more than 5 of the *and* their combined disk usage is no more than 10 megabytes. If none of these options are specified, the default expiration s `-C 10` which keeps ten old log files. If no files are to be expired, use `-C 0` to prevent expiration by default.

Options The following options are supported:

`-a` *post_command*
Execute the *post_command* after renaming the log file. *post_command* is passed to `sh -c`.

Specify *post_command* as a valid shell command. Use quotes to protect spaces or shell metacharacters in *post_command*.

This option can be used to restart a daemon that is writing to the file. When rotating multiple logs with one `logadm` command, *post_command* is executed only once after all the logs are rotated, not once per rotated log.

`-A` *age*
Delete any versions that have not been modified for the amount of time specified by *age*.

Specify *age* as a number followed by an `h` (hours), `d` (days), `w`(weeks), `m` (months), or `y` (years).

`-b` *pre_command*
Execute *pre_command* before renaming the log file. *pre_command* is passed to `sh -c`.

Specify *pre_command* as a valid shell command. Use quotes to protect spaces or shell metacharacters in the *pre_command*.

This option can be used to stop a daemon that is writing to the file. When rotating multiple logs with one `logadm` command, *pre_command* is executed only once before all the logs are rotated, not once per rotated log.

`-c`
Rotate the log file by copying it and truncating the original logfile to zero length, rather than renaming the file.

`-C` *count*
Delete the oldest versions until there are not more than *count* files left.

If no expire options (`-A`, `-C`, or `-S`) are specified, `-C 10` is the default. To prevent the default expire rule from being added automatically, specify `-C 0`.

`-e` *mail_addr*
Send error messages by email to *mail_addr*.

As `logadm` is typically run from `cron(1M)`, error messages are captured by `cron` and mailed to the owner of the `crontab`.

This option is useful if you want the mail regarding error messages to go to another address instead. If no errors are encountered, no mail message is generated.

`-E` *cmd*
Execute *cmd* to expire the file, rather than deleting the old log file to expire it.

cmd is passed it to `sh -c`. The file is considered expired after *cmd* completes. If the old log file is not removed or renamed by the *cmd*, `logadm` considers it for expiration the next time that it runs on the specified log file. If present, the keyword `$file` is expanded in the specified *cmd*to the name of the file being expired.

This option is useful for tasks such as mailing old log files to administrators, or copying old log files to long term storage.

`-f` *conf_file*
Use *conf_file* instead of `/etc/logadm.conf`.

This option allows non-root users to keep their own `logadm` configuration files.

`-g` *group*
Create a new empty file with the ID specified by *group*, instead of preserving the group ID of the log file.

Specify *group* by name or by numeric group ID, as accepted by `chgrp`(1).

This option requires the ability to change file group ownership using the `chgrp(1)` command.

`-h`
Print a help message that describes `logadm`'s options.

`-l`
Use local time rather than the Coordinated Universal Time (UTC) when naming rotated log files (see the discussion of percent sequences in the templates supplied with the `-t` option).

`-m` *mode*
Create a new empty file with the mode specified by *mode*, instead of preserving the mode of the log file.

Specify *mode* in any form that is accepted by the `chmod(1)` command.

`-M` *cmd*
Use *cmd* to rename the log file. If the keyword `$file` is specified, it is expanded to the name of the log file. Similarly, the keyword `$nfile` is expanded to the new name of the log file. The `$nfile` keyword is only available with commands provided with the `-M` option. After the command completes, the log file is replaced by the rotate file. The default *cmd* is "`/bin/mv $file $nfile`".

`-n`
Print the actions that the `logadm` command will perform without actually performing them.

This option is useful for checking arguments before making any changes to the system.

It is important to remember, however, that since log rotating actions are only printed with this option, `logadm` might not find files that need expiring, but if run without the `-n` `logadm` might create a file that needs expiring by performing the log rotating actions. Therefore, if you see no files being expired with the `-n` option, files still might be expired without it.

`-N`
Prevent an error message if the specified logfile does not exist. Normally, `logadm` produces an error message if the log file is not found. With `-N`, if the log file doesn't exist `logadm` moves on to the expire rules (if any) and then to the next log file (if any), without creating the empty replacement log file.

`-o` *owner*
Create the new empty file with *owner*, instead of preserving the owner of the log file.

Specify *owner* in any form that is accepted by the `chown(1)` command.

`-p` *period*
Rotate a log file after the specified time period (*period*) .

Specify *period* as a number followed by `d` for days, `h` for hours, `w` for weeks, `m` for months (really 30 days) or `y` for years. There are also two special values for period: `now` and `never`. `-p now` forces log rotation. `-p never` forces no log rotation.

`-P` *timestamp*
Used by `logadm` to record the last time the log was rotated in `/etc/logadm.conf`.

This option uses *timestamp* to determine if the log rotation period has passed. The format of *timestamp* matches the format generated by `ctime(3C)`, with quotes around it to protect embedded spaces. *timestamp* is always recorded in the Coordinated Universal Time (UTC) timezone.

`-r`
Remove any entries corresponding to the specified *logname* from the `/etc/logadm.conf`.

`-R` *cmd*
Run the *cmd* when an old log file is created by a log rotation. If the keyword `$file` is embedded in the specified command, it is expanded to the name of the old log file just created by log rotation.

This option is useful for processing log file contents after rotating the log. *cmd* is executed by passing it to `sh -c`. When rotating multiple logs with one logadm command, the command supplied with `-R` is executed once every time a log is rotated. This is useful for post-processing a log file (that is, sorting it, removing uninteresting lines, etc.). The `-a` option is a better choice for restarting daemons after log rotation.

`-s` *size*
Rotate the log file only if its size is greater than or equal to *size*.

Specify *size* as a number followed by the letter `b` for bytes, `k` for kilobytes, `m` for megabytes, or `g` for gigabytes.

`-S` *size*
Delete the oldest versions until the total disk space used by the old log files is less than the specified size.

Specify *size* as a number followed by the letter `b` for bytes, `k` for kilobytes, `m` for megabytes, or `g` for gigabytes.

`-t` *template*
Specify the template to use when renaming log files.

template can be a simple name, such as `/var/adm/oldfile`, or it can contain special keywords which are expanded by `logadm` and are in the form `$`*word*. Allowed sequences are:

$*file*
The full path name of the file to be rotated

$*dirname*
The directory of the file to be rotated

$*basename*
The log file name, without the directory name

$*n*
The version number, `0` is most recent, `1` is next most recent, and so forth

$*N*
The same as $*n*, but starts at `1` instead of zero

$*secs*
The number of seconds since `00:00:00 UTC, January 1,1970`

$*nodename*
Expands to the output of `uname -n`

$*platform*
Expands to the output of `uname -i`

$*isa*
Expands to the output of `uname -p`

$*release*
Expands to the output of `uname -r`

$*machine*
Expands to the output of `uname -m`

$*domain*
Expands to the output of `domainname`

To actually have the dollar sign character in the file name, use `$$`. Any percent sequences allowed by `strftime(3C)` are also allowed, for example, `%d` expands to the day of the month. To actually have a percent sign character in the file name, use `%%`. Both dollar-sign keywords and percent sequences can appear anywhere in the template. If the template results in a pathname with non-existent directories, they are created as necessary when rotating the log file.

If no `-t`option is specified, the default template is `$file.$n`. Actual *rotation* of log files, where each version is shifted up until it expires is done using the `$n` keyword. If the template does not contain the `$n` keyword, the log file is simply renamed to the new name and then the expire rules, if any, are applied.

`-T` *pattern*
Normally `logadm` looks for a list of old log files by turning the template (specified with the `-t` option) into a pattern and finding existing files whose names match that pattern. The `-T` option causes the given pattern to be used instead.

This option is useful if another program fiddles with the old log file names, like a `cron` job to compress them over time. The pattern is in the form of a pathname with special characters such as `*` and `?` as supported by `csh(1)` filename substitution.

`-v`
Print information about the actions being executed in verbose mode.

`-V`
Validate the configuration file.

This option validates that an entry for the specified *logname* exists in the `/etc/logadm.conf` file and is syntactically correct. If *logname* is not specified, all entries in the configuration file are validated. If a `logname` argument is specified, the command validates the syntax of that entry. If the entry is found, it is printed and the exit value of the command is true. Otherwise the exit value is false.

`-w` *entryname*
Write an entry into the config file (that is, `/etc/logadm.conf`) that corresponds to the current command line arguments. If an entry already existed for the specified entryname, it is removed first. This is the preferred method for updating `/etc/logadm.conf`, because it prevents syntax errors in that file. The *entryname* is an argument to an invocation of `logadm`. The *entryname* might be chosen as something easy to remember or it can be the pathname ofthe log file.

If no log file name is provided on a `logadm` command line, the entry name is assumed to be the same as the log file name. For example, the following two lines achieve the same result, keeping two copies of rotated log files:

```
% logadm -C2 -w mylog /my/really/long/log/file/name
% logadm -C2 -w /my/really/long/log/file/name
```

`-z` *count*
Compress old log files as they are created. *count* of the most recent log files are left uncompressed, therefore making the *count* most recent files easier to peruse. Use *count* of zero to compress all old logs.

The compression is done with `gzip`(1) and the resulting log file has the suffix of `.gz`.

Operands The following operands are supported:

logname
Identifies the name of the entry in `/etc/logadm.conf`. If the log file name is specified in the *logname* field, it is assumed that *logname* is the same as the actual log file name.

Examples **EXAMPLE 1** Rotating a File and Keeping Previous Versios

The following example rotates the `/var/adm/exacct/proc` file, keeping ten previous versions in `/var/adm/exacct/proc.0` through `/var/adm/exacct/proc.9`.

Tell `logadm` to copy the file and truncate it.

```
 % logadm -c /var/adm/exacct/proc
```

EXAMPLE 2 Rotating syslog

The following example rotates `syslog` and keeps eight log files. Old log files are put in the directory `/var/oldlogs` instead of `/var/log`:

```
% logadm -C8 -t'/var/oldlogs/syslog.$n' /var/log/syslog
```

EXAMPLE 3 Rotating /var/adm/sulog and Expiring Based on Age

The following entry in the `/etc/logadm.conf` file rotates the `/var/adm/sulog` file and expires any copies older than 30 days.

```
/var/adm/sulog -A 30d
```

EXAMPLE 4 Rotating Files and Expiring Based on Disk Usage

The following entry in the `/etc/logadm.conf` file rotates the `/var/adm/sulog` file and expires old log files when more than 100 megabytes are used by the sum of all the rotated log files.

```
/var/adm/sulog -S 100m
```

EXAMPLE 5 Creating an Entry that Stores the Logfile Name

This example creates an entry storing the log file name and the fact that we want to keep 20 copies in `/etc/logadm.conf`, but the `-p never` means the entry is ignored by the normal `logadm` run from root's crontab every morning.

```
% logadm -w locallog /usr/local/logfile -C20 -p never
```

Use the following entry on the command line to override the `-p never` option:

```
 % logadm -p now locallog
```

EXAMPLE 6 Rotating the apache Error and Access Logs

The following example rotates the apache error and access logs monthly to filenames based on current year and month. It keeps the 24 most recent copies and tells apache to restart after renaming the logs.

This command is run once, and since the `-w` option is specified, an entry is made in `/etc/logadm.conf` so the apache logs are rotated from now on.

```
 % logadm -w apache -p 1m -C 24\
     -t '/var/apache/old-logs/$basename.%Y-%m'\
     -a '/usr/apache/bin/apachectl graceful'\
     '/var/apache/logs/*{access,error}_log'
```

This example also illustrates that the entry name supplied with the `-w` option doesn't have to match the log file name. In this example, the entry name is `apache` and once the line has been run, the entry in `/etc/logadm.conf` can be forced to run by executing the following command:

```
% logadm -p now apache
```

Because the expression matching the apache log file names was enclosed in quotes, the expression is stored in `/etc/logadm.conf`, rather than the list of files that it expands to. This means that each time `logadm` runs from `cron` it expands that expression and checks all the log files in the resulting list to see if they need rotating.

The following command is an example without the quotes around the log name expression. The shell expands the last argument into a list of log files that exist at the time the command is entered, and writes an entry to `/etc/logadm.conf` that rotates the files.

```
logadm -w apache /var/apache/logs/*_log
```

Files `/etc/logadm.conf`
configuration file for logadm command

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `chgrp(1)`, `chmod(1)`, `chown(1)`, `crontab(1)`, `csh(1)`, `gzip(1)`, `cron(1M)`, `ctime(3C)`, `strftime(3C)`, `logadm.conf(4)`, `attributes(5)`

Notes When `logadm` applies expire conditions (supplied by the `-A`, `-C`, and `-S` options), it deletes files, the oldest first, until the conditions are satisfied. If the template used for naming the old logs contained `$n` or `$N`, `logadm` picks the highest value of `$n` or `$N` found in the old log file names first. If the template used is something else, `logadm` uses the modification time to determine which files to expire first. This might not be the expected behavior if an old log file has been modified since it was rotated.

Note that, depending on log file sizes and number of log files, log file rotations can be very time-consuming.

By default, `logadm` works in GMT. Therefore, all entries written to the `/etc/logadm.conf` file (see `logadm.conf(4)`) will have a GMT timestamp. Users can use the `-l` option to set `logadm` to local time.

Name logins – list user and system login information

Synopsis `/usr/bin/logins [-admopstux] [-g` *group...*`] [-l` *login_name...*`]`

Description This command displays information on user and system logins known to the system. Contents of the output is controlled by the command options and can include the following: user or system login, user id number, `passwd` account field value (user name or other information), primary group name, primary group id, multiple group names, multiple group ids, home directory, login shell, and four password aging parameters. The default information is the following: login id, user id, primary group name, primary group id and the account field value. Output is sorted by user id, system logins, followed by user logins.

Options Options may be used together. If so, any login that matches any criteria are displayed.

The following options are supported:

`-a` Add two password expiration fields to the display. The fields show how many days a password can remain unused before it automatically becomes inactive, and the date that the password expires.

`-d` Selects logins with duplicate uids.

`-g` *group* Selects all users belonging to `group`, sorted by login. Multiple groups can be specified as a comma-separated list. When the `-l` and `-g` options are combined, a user is only listed once, even if the user belongs to more than one of the selected groups.

`-l` *login_name...* Selects the requested login. Multiple logins can be specified as a comma-separated list. Depending on the nameservice lookup types set in `/etc/nsswitch.conf`, the information can come from the `/etc/passwd` and `/etc/shadow` files and other nameservices. When the `-l` and `-g` options are combined, a user is only listed once, even if the user belongs to more than one of the selected groups.

`-m` Displays multiple group membership information.

`-o` Formats output into one line of colon-separated fields.

`-p` Selects logins with no passwords.

`-s` Selects all system logins.

`-t` Sorts output by login instead of by uid.

`-u` Selects all user logins.

`-x` Prints an extended set of information about each selected user. The extended information includes home directory, login shell and password aging information, each displayed on a separate line. The password information consists of password status (`PS` for password, `NP` for no password or `LK` for locked). If the login is passworded, status is followed

by the date the password was last changed, the number of days required between changes, and the number of days allowed before a change is required. The password aging information shows the time interval that the user receives a password expiration warning message (when logging on) before the password expires.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also attributes(5)

Name lpadmin – configure the LP print service

Synopsis

```
lpadmin -p printer {options}

lpadmin -x dest

lpadmin -d [dest]

lpadmin -S print-wheel -T [-A alert-type] [-W minutes]
     [-Q requests]
```

Description `lpadmin` configures the LP print service by defining printers and devices. It is used to add and change printers, to remove printers from service, to set or change the system default destination, to define alerts for printer faults, and to mount print wheels.

Options The `lpadmin` command has options for:

- Adding or changing a printer
- Removing a printer destination
- Setting or changing the system default destination
- Setting an alert for a print wheel

The options for each of the above categories are specified in the following subsections.

Several options support the use of lists. A list might contain, for example, user names, printers, printer forms, or content types. A list of multiple items can have the form of either comma-separated names or have the entire list enclosed by double quotes with a space between each name. For example, both lists below are acceptable:

```
one,two,three
"one two three"
```

Adding or Changing a Printer

The first form of the `lpadmin` command (`lpadmin -p` *printer* {*options*}) configures a new printer or changes the configuration of an existing printer. It also starts the print scheduler.

When creating a new printer, one of three options (`-v`, `-U`, or `-s`) must be supplied. In addition, only one of the following can be supplied: `-e`, `-i`, or `-m`; if none of these three options is supplied, the model standard is used. The `-h` and `-l` options are mutually exclusive. Printer and class names must be no longer than 14 characters and must consist entirely of the characters `A-Z`, `a-z`, `0-9`, dash (`-`) and underscore (_). If `-s` is specified, the following options are invalid: `-A`, `-e`, `-F`, `-h`, `-i`, `-l`, `-M`, `-m`, `-o`, `-U`, `-v`, and `-W`.

The following options can appear in any order.

`-A` *alert-type* [`-W` *minutes*]
The `-A` option is used to define an alert that informs the administrator when a printer fault is detected, and periodically thereafter, until the printer fault is cleared by the administrator. The *alert-types* are:

`mail` Send the alert message using mail (see `mail(1)`) to the administrator.

`write`	Write the message to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is chosen arbitrarily.
`quiet`	Do not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the fault has been cleared and printing resumes, messages are sent again when another fault occurs with the printer.
`showfault`	Attempt to execute a fault handler on each system that has a print job in the queue. The fault handler is `/etc/lp/alerts/printer`. It is invoked with three parameters: *printer_name*, `date`, *file_name*. The *file_name* is the name of a file containing the fault message.
`none`	Do not send messages; any existing alert definition for the printer is removed. No alert is sent when the printer faults until a different alert-type (except `quiet`) is used.
shell-command	Run the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blank spaces embedded in the command, enclose the command in quotes. Notice that the `mail` and `write` values for this option are equivalent to the values `mail` *user-name* and `write` *user-name* respectively, where *user-name* is the current name for the administrator. This is the login name of the person submitting this command unless he or she has used the `su` command to change to another user ID. If the `su` command has been used to change the user ID, then the *user-name* for the new ID is used.
`list`	Display the type of the alert for the printer fault. No change is made to the alert.

When a fault occurs, the printing subsystem displays a message indicating that printing for a specified printer has stopped and the reason for the stoppage. The message also indicates that printing will restart in a few minutes and that you can enter an `enable` command if you want to restart sooner than that.

Following a fault that occurs in the middle of a print job, the job is reprinted from the beginning. An exception to this occurs when you enter a command, such as the one shown below, that changes the page list to be printed.

```
% lp -i request-id -P ...
```

For a given print request, the presence of multiple reasons for failure indicate multiple attempts at printing.

The LP print service can detect printer faults only through an adequate fast filter and only when the standard interface program or a suitable customized interface program is used. Furthermore, the level of recovery after a fault depends on the capabilities of the filter.

If, instead of a single printer, the keyword `all` is displayed in an alert, the alert applies to all printers.

If the `-W` option is not used to arrange fault alerting for *printer*, the default procedure is to mail one message to the administrator of *printer* per fault. This is equivalent to specifying `-W once` or `-W 0`. If *minutes* is a number greater than zero, an alert is sent at intervals specified by *minutes*.

`-c` *class*
Insert *printer* into the specified *class*. *class* is created if it does not already exist. This option requires the `-U` *dial-info* or `-v` *device* options.

`-D` *comment*
Save this *comment* for display whenever a user asks for a full description of *printer* (see `lpstat(1)`). The LP print service does not interpret this comment.

`-e` *printer*
Copy the interface program of an existing *printer* to be the interface program for *printer*. (Options `-i` and `-m` must not be specified with this option.)

`-f allow:`*form-list*
`-f deny:`*form-list*
Allow or deny the forms in *form-list* to be printed on *printer*. By default no forms are allowed on a new printer.

For each printer, the LP print service keeps two lists of forms: an "allow-list" of forms that can be used with the printer, and a "deny-list" of forms that cannot be used with the printer. With the `-f allow` option, the forms listed are added to the allow-list and removed from the deny-list. With the `-f deny` option, the forms listed are added to the deny-list and removed from the allow-list.

If the allow-list is not empty, only the forms in the list can be used on the printer, regardless of the contents of the deny-list. If the allow-list is empty, but the deny-list is not, the forms in the deny-list cannot be used with the printer. All forms can be excluded from a printer by specifying `-f deny:all`. All forms can be used on a printer (provided the printer can handle all the characteristics of each form) by specifying `-f allow:all`.

The LP print service uses this information as a set of guidelines for determining where a form can be mounted. Administrators, however, are not restricted from mounting a form on any printer. If mounting a form on a particular printer is in disagreement with the information in the allow-list or deny-list, the administrator is warned but the mount is accepted. Nonetheless, if a user attempts to issue a print or change request for a form and printer combination that is in disagreement with the information, the request is accepted

only if the form is currently mounted on the printer. If the form is later unmounted before the request can print, the request is canceled and the user is notified by mail.

If the administrator tries to specify a form as acceptable for use on a printer that does not have the capabilities needed by the form, the command is rejected.

Notice the other use of `-f`, with the `-M` option, below.

The `-T` option must be invoked first with `lpadmin` to identify the printer type before the `-f` option can be used.

`-F` *fault-recovery*
This option specifies the recovery to be used for any print request that is stopped because of a printer fault, according to the value of *fault-recovery*:

`continue`	Continue printing on the top of the page where printing stopped. This requires a filter to wait for the fault to clear before automatically continuing.
`beginning`	Start printing the request again from the beginning.
`wait`	Disable printing on *printer* and wait for the administrator or a user to enable printing again. During the wait, the administrator or the user who submitted the stopped print request can issue a change request that specifies where printing should resume. (See the `-i` option of the `lp` command.) If no change request is made before printing is enabled, printing resumes at the top of the page where stopped, if the filter allows; otherwise, the request is printed from the beginning.

`-h`
Indicate that the device associated with the printer is hardwired. If neither of the mutually exclusive options, `-h` and `-l`, is specified, `-h` is assumed.

`-i` *interface*
Establish a new interface program for *printer*. *interface* is the pathname of the new program. (The `-e` and `-m` options must not be specified with this option.)

`-I` *content-type-list*
Allow *printer* to handle print requests with the content types listed in a *content-type-list*.

The type `simple` is recognized as the default content type for files in the UNIX system. A `simple` type of file is a data stream containing only printable ASCII characters and the following control characters:

Control Char	Octal Value	Meaning
BACKSPACE	10	Move back one char, except

Control Char	Octal Value	Meaning
		at beginning of line
TAB	11	Move to next tab stop
LINEFEED	12	Move to beginning of
(newline)		next line
FORMFEED	14	Move to beginning of
		next page
RETURN	15	Move to beginning of
		current line

To prevent the print service from considering `simple` a valid type for the printer, specify either an explicit value (such as the printer type) in the *content-type-list*, or an empty list. If you do want `simple` included along with other types, you must include `simple` in the *content-type-list*.

In addition to content types defined by the print administrator, the type `PostScript` is recognized and supported by the Solaris print subsystem. This includes filters to support `PostScript` as the printer content type.

The type `any` is recognized as a special content type for files. When declared as the input type for a printer, it signals the print sub-system not to do any filtering on the file before sending it to the printer.

Except for `simple` and `any`, each *content-type* name is determined by the administrator. If the printer type is specified by the `-T` option, then the printer type is implicitly considered to be also a valid content type.

`-l`
Indicate that the device associated with *printer* is a login terminal. The LP scheduler (`lpsched`) disables all login terminals automatically each time it is started. (The `-h` option must not be specified with this option.)

`-m` *model*
Select *model* interface program, provided with the LP print service, for the printer. (Options `-e` and `-i` must not be specified with this option.)

`-M -f` *form-name* [`-a` [`-o filebreak`]] [`-t` *tray-number*]]
Mount the form *form-name* on *printer*. Print requests that need the pre-printed form *form-name* is printed on *printer*. If more than one printer has the form mounted and the user has specified `any` (with the `-d` option of the `lp` command) as the printer destination, then the print request is printed on the one printer that also meets the other needs of the request.

The page length and width, and character and line pitches needed by the form are compared with those allowed for the printer, by checking the capabilities in the `terminfo` database for the type of printer. If the form requires attributes that are not available with the printer, the administrator is warned but the mount is accepted. If the form lists a print wheel as mandatory, but the print wheel mounted on the printer is different, the administrator is also warned but the mount is accepted.

If the `-a` option is given, an alignment pattern is printed, preceded by the same initialization of the physical printer that precedes a normal print request, with one exception: no banner page is printed. Printing is assumed to start at the top of the first page of the form. After the pattern is printed, the administrator can adjust the mounted form in the printer and press return for another alignment pattern (no initialization this time), and can continue printing as many alignment patterns as desired. The administrator can quit the printing of alignment patterns by typing `q`.

If the `-o filebreak` option is given, a formfeed is inserted between each copy of the alignment pattern. By default, the alignment pattern is assumed to correctly fill a form, so no formfeed is added.

If the `-t` *tray-number* option is specified, printer tray *tray-number* is used.

A form is "unmounted" either by mounting a new form in its place or by using the `-f none` option. By default, a new printer has no form mounted.

Notice the other use of `-f` without the `-M` option above.

`-M -S` *print-wheel*
Mount the *print-wheel* on *printer*. Print requests that need the *print-wheel* are printed on *printer*. If more than one printer has *print-wheel* mounted and the user has specified `any` (with the `-d` option of the `lp` command) as the printer destination, then the print request is printed on the one printer that also meets the other needs of the request.

If the *print-wheel* is not listed as acceptable for the printer, the administrator is warned but the mount is accepted. If the printer does not take print wheels, the command is rejected.

A print wheel is "unmounted" either by mounting a new print wheel in its place or by using the option `-S none`. By default, a new printer has no print wheel mounted.

Notice the other uses of the `-S` option without the `-M` option described below.

`-n` *ppdfilename*
Specify a PPD file for creating and modifying printer queues. *ppdfilename* is the full path and file name to the PPD file. Used in conjunction with the `-p`, `-d`, `-x`, or `-S` options.

`-o` *option*
The `-o` option defines default printer configuration values given to an interface program. The default can be explicitly overwritten for individual requests by the user (see `lp(1)`), or taken from a preprinted form description (see `lpforms(1M)` and `lp(1)`).

There are several options which are predefined by the system. In addition, any number of key-value pairs can be defined. See the section "Predefined Options Used with the `-o` Option", below.

`-P` *paper-name*
Specify a paper type list that the printer supports.

`-r` *class*
Remove *printer* from the specified *class*. If *printer* is the last member of *class*, then *class* is removed.

`-S` *list*
Allow either the print wheels or aliases for character sets named in *list* to be used on the printer.

If the printer is a type that takes print wheels, then *list* is a comma or space separated list of print wheel names. These are the only print wheels considered mountable on the printer. (You can always force a different print wheel to be mounted.) Until the option is used to specify a list, no print wheels are considered mountable on the printer, and print requests that ask for a particular print wheel with this printer are rejected.

If the printer is a type that has selectable character sets, then *list* is a list of character set name "mappings" or aliases. Each "mapping" is of the form *known-name=alias* The *known-name* is a character set number preceded by `cs` (such as `cs3` for character set three) or a character set name from the `terminfo` database entry `csnm`. See `terminfo(4)`. If this option is not used to specify a list, only the names already known from the `terminfo` database or numbers with a prefix of `cs` is acceptable for the printer. If *list* is the word `none`, any existing print wheel lists or character set aliases are removed.

Notice the other uses of the `-S` with the `-M` option described above.

The `-T` option must be invoked first with `lpadmin` to identify the printer type before the `-S` option can be used.

`-s` *system-name*
The `-s` option can be used for both remote or local printers. For remote printers:

`-s` *system-name*[`!`*printer-name*] (UUCP format)
`-s` *printer-name*`@`*system-name* (RCMD format)

Make a remote printer (one that must be accessed through another system) accessible to users on your system. *system-name* is the name of the remote system on which the remote printer is located it. *printer-name* is the name used on the remote system for that printer. For example, if you

want to access *printer1* on *system1* and you want it called *printer2* on your system:

`-p` *printer2* `-s` *system1*`!`*printer1*

`-p` *printer2* `-s` *printer1*`@`*system1*

`-s` *scheme*`://`*end-point* (URI format)

Make a remote printer (one that must be accessed through another system) accessible to users on your system. The supported schemes include `lpd` and `ipp`. Specify URI's using the `lpd` format as follows:

`lpd://`*server*`/`*printers*`/`*queue*`[#Solaris]`

URI's using the `ipp` format are defined by the remote print server. They are generally of the format:

`ipp://`*server*`/`*printers*`/`*queue*

In either case, *server* specifies the hostname or IP address of the remote print server, *queue* specifies the name of the print queue on the remote print server, and the optional `#Solaris`specifies that the remote print server is a Solaris server when `lpd` URI format is being used.

For example:

`-p` *printer* `-s lpd://`*server*`/`*printers*`/`*queue*`#Solaris`
`-p` *printer* `-s ipp://`*server*`/`*printers*`/`*queue*

For local printers:

`-s "localhost"` Use `localhost` for the *system-name* to be used by the print service. In an environment where the nodename is variable, print queues are invalidated when the nodename changes. Using `localhost` as the *system-name* allows print queues to be maintained across changing nodenames. The *system-name*, as used by the print service, is only set to `localhost` when explicitely set with this option; by default, `lpadmin` sets *system-name* to *nodename*. For example, if you want to configure a new printer on the local system, and want it called *printer3*:

`-p` *printer3* `-s` *localhost* `-v` *device*

This option should never be used when creating name service maps.

`-T` *printer-type-list*
Identify the printer as being of one or more *printer-type*s. Each *printer-type* is used to extract data from the `terminfo` database; this information is used to initialize the printer before printing each user's request. Some filters might also use a *printer-type* to convert content for the printer. If this option is not used, the default *printer-type* is `unknown`. No information is extracted from `terminfo` so each user request is printed without first initializing the printer. Also, this option must be used if the following are to work: `-o cpi`, `-o lpi`, `-o` width, and `-o length` options of the `lpadmin` and `lp` commands, and the `-S` and `-f` options of the `lpadmin` command.

If the *printer-type-list* contains more than one type, then the *content-type-list* of the `-I` option must either be specified as `simple`, as empty (`-I ""`), or not specified at all.

`-t`*number-of-trays*
Specify the number of trays when creating the printer.

`-u allow:`*login-ID-list*
`-u deny:`*login-ID-list*
Allow or deny the users in *login-ID-list* access to the printer. By default all users are allowed on a new printer. The *login-ID-list* argument can include any or all of the following constructs:

login-ID	a user on any system
system-name`!`*login-ID*	a user on system *system-name*
system-name`!all`	all users on system *system-name*
`all!`*login-ID*	a user on all systems
`all`	all users on all systems

For each printer, the LP print service keeps two lists of users: an "allow-list" of people allowed to use the printer, and a "deny-list" of people denied access to the printer. With the `-u allow` option, the users listed are added to the allow-list and removed from the deny-list. With the `-u deny` option, the users listed are added to the deny-list and removed from the allow-list.

If the allow-list is not empty, only the users in the list can use the printer, regardless of the contents of the deny-list. If the allow-list is empty, but the deny-list is not, the users in the deny-list cannot use the printer. All users can be denied access to the printer by specifying `-u deny:all`. All users can use the printer by specifying `-u allow:all`.

The `-U` option allows your print service to access a remote printer. (It does not enable your print service to access a remote printer service.) Specifically, `-U` assigns the "dialing" information *dial-info* to the printer. *dial-info* is used with the `dial` routine to call the printer. Any network connection supported by the Basic Networking Utilities works.

dial-info can be either a phone number for a modem connection, or a system name for other kinds of connections. Or, if `-U direct` is given, no dialing takes place, because the name `direct` is reserved for a printer that is directly connected. If a system name is given, it is used to search for connection details from the file `/etc/uucp/Systems` or related files. The Basic Networking Utilities are required to support this option. By default, `-U direct` is assumed.

`-v` *device*
Associate a *device* with *printer*. *device* is the path name of a file that is writable by `lp`. Notice that the same *device* can be associated with more than one printer.

`-v` *scheme*`://`*end-point*
Associate a network attached device with printer.

scheme is the method or protocol used to access the network attached device and *end-point* is the information necessary to contact that network attached device. Use of this device format requires the use of the `uri` interface script and can only be used with the `smb` scheme at this time.

For example:

```
# lpadmin -p queue -v smb://smb-service/printer -m uri
```

See the `/usr/sfw/man/man1m/smbspool.1m` man page for details.

Removing a Printer Destination

The `-x` *dest* option removes the destination *dest* (a printer or a class), from the LP print service. If *dest* is a printer and is the only member of a class, then the class is deleted, too. If *dest* is `all`, all printers and classes are removed. If there are no remaining local printers and the scheduler is still running, the scheduler is shut down.

No other *options* are allowed with `-x`.

Setting/Changing the System Default Destination

The `-d` [*dest*] option makes *dest* (an existing printer or class) the new system default destination. If *dest* is not supplied, then there is no system default destination. No other *options* are allowed with `-d`.

Setting an Alert for a Print Wheel

`-S` *print-wheel* [`-A` *alert-type*] [`-W` *minutes*] [`-Q` *requests*] `-T`
The `-S` *print-wheel* option is used with the `-A` *alert-type* option to define an alert to mount the print wheel when there are jobs queued for it. If this command is not used to arrange alerting for a print wheel, no alert is sent for the print wheel. Notice the other use of `-A`, with the `-p` option, above.

The *alert-types* are:

`mail`	Send the alert message using the `mail` command to the administrator.
`write`	Write the message, using the `write` command, to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is arbitrarily chosen.

`quiet` Do not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the *print-wheel* has been mounted and subsequently unmounted, messages are sent againwhen the number of print requests reaches the threshold specified by the `-Q` option.

`none` Do not send messages until the `-A` option is given again with a different *alert-type* (other than `quiet`).

shell-command Run the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blanks embedded in the command, enclose the command in quotes. Notice that the `mail` and `write` values for this option are equivalent to the values `mail` *user-name* and `write` *user-name* respectively, where *user-name* is the current name for the administrator. This is the login name of the person submitting this command unless he or she has used the `su` command to change to another user ID. If the `su` command has been used to change the user ID, then the *user-name* for the new ID is used.

`list` Display the type of the alert for the print wheel on standard output. No change is made to the alert.

The message sent appears as follows:

```
The print wheel print-wheel needs to be mounted
on the printer(s):
printer(integer1requests) integer2 print requests
await this print wheel.
```

The printers listed are those that the administrator had earlier specified were candidates for this print wheel. The number *integer1* listed next to each printer is the number of requests eligible for the printer. The number *integer2* shown after the printer list is the total number of requests awaiting the print wheel. It is less than the sum of the other numbers if some requests can be handled by more than one printer.

If the *print-wheel* is `all`, the alerting defined in this command applies to all print wheels already defined to have an alert.

If the `-W` option is not given, the default procedure is that only one message is sent per need to mount the print wheel. Not specifying the `-W` option is equivalent to specifying `-W once` or `-W 0`. If *minutes* is a number greater than zero, an alert is sent at intervals specified by *minutes*.

If the `-Q` option is also given, the alert is sent when a certain number (specified by the argument *requests*) of print requests that need the print wheel are waiting. If the `-Q` option is not given, or *requests* is 1 or `any` (which are both the default), a message is sent as soon as

anyone submits a print request for the print wheel when it is not mounted.

Predefined Options Used with the -o Option

A number of options, described below, are predefined for use with `-o`. These options are used for adjusting printer capabilities, adjusting printer port characteristics, configuring network printers, and controlling the use of banner. The `-o` also supports an arbitrary *keyword=value* format, which is referred to below as an undefined option.

Adjusting Printer Capabilities

The `length`, `width`, `cpi`, and `lpi` parameters can be used in conjunction with the `-o` option to adjust printer capabilities. The format of the parameters and their values is as follows:

`length=`*scaled-decimal-number*
`width=`*scaled-decimal-number*
`cpi=`*scaled-decimal-number*
`lpi=`*scaled-decimal-number*

The term *scaled-decimal-number* refers to a non-negative number used to indicate a unit of size. The type of unit is shown by a "trailing" letter attached to the number. Three types of *scaled-decimal-numbers* can be used with the LP print service: numbers that show sizes in centimeters (marked with a trailing `c`); numbers that show sizes in inches (marked with a trailing `i`); and numbers that show sizes in units appropriate to use (without a trailing letter), that is, lines, characters, lines per inch, or characters per inch.

The option values must agree with the capabilities of the type of physical printer, as defined in the terminfo database for the printer type. If they do not, the command is rejected.

The defaults are defined in the `terminfo` entry for the specified printer type. The defaults can be reset by:

`lpadmin -p` *printername* **`-o length=`**
`lpadmin -p` *printername* **`-o width=`**
`lpadmin -p` *printername* **`-o cpi=`**
`lpadmin -p` *printername* **`-o lpi=`**

Adjusting Printer Port Characteristics

You use the `stty` keyword in conjunction with the o option to adjust printer port characteristics. The general form of the `stty` portion of the command is:

`stty="'`*stty-option-list*`'"`

The *stty-option-list* is not checked for allowed values, but is passed directly to the `stty` program by the standard interface program. Any error messages produced by `stty` when a request is processed (by the standard interface program) are mailed to the user submitting the request.

The default for `stty` is:

```
stty="'9600 cs8 -cstopb -parenb ixon
     -ixany opost -olcuc onlcr
     -ocrnl -onocr
     -onlret -ofill nl0 cr0 tab0 bs0 vt0 ff0'"
```

The default can be reset by:

```
lpadmin -p printername -o stty=
```

Configuring Network Printers

The `dest`, `protocol`, `bsdctrl`, and `timeout` parameters are used in conjunction with the `-o` option to configure network printers. The format of these keywords and their assigned values is as follows:

```
dest=string protocol=string bsdctrl=string \
     timeout=non-negative-integer-seconds
```

These four options are provided to support network printing. Each option is passed directly to the interface program; any checking for allowed values is done there.

The value of `dest` is the name of the destination for the network printer; the semantics for value `dest` are dependent on the printer and the configuration. There is no default.

The value of option `protocol` sets the over-the-wire protocol to the printer. The default for option `protocol` is `bsd`. The value of option `bsdctrl` sets the print order of control and data files (BSD protocol only); the default for this option is `control file first`. The value of option `timeout` sets the seed value for backoff time when the printer is busy. The default value for the `timeout` option is `10` seconds. The defaults can be reset by:

```
lpadmin -p printername -o protocol=
lpadmin -p printername -o bsdctrl=
lpadmin -p printername -o timeout=
```

Controlling the Use of the Banner Page

Use the following commands to control the use of the banner page:

```
lpadmin -p printer -o nobanner
lpadmin -p printer -o banner
lpadmin -p printer -o banner=always
lpadmin -p printer -o banner=never
lpadmin -p printer -o banner=optional
```

The first and fifth commands (`-o nobanner` and `-o banner=optional`) are equivalent. The default is to print the banner page, unless a user specifies `-o nobanner` on an `lp` command line.

The second and third commands (`-o banner` and `-o banner=always`) are equivalent. Both cause a banner page to be printed always, even if a user specifies `lp -o nobanner`. The root user can override this command.

The fourth command (`-o banner=never`) causes a banner page never to be printed, even if a user specifies `lp -o banner`. The root user can override this command.

Undefined Options

The `-o` option supports the use of arbitrary, user-defined options with the following format:

key=*value*	Each *key*=*value* is passed directly to the interface program. Any checking for allowed values is done in the interface program. Any default values for a given *key*=*value* option are defined in the interface program. If a default is provided, it can be reset by typing the key without any value: `lpadmin -p` *printername* `-o` *key*=
`lpadmin -p` *printer* `-o foo | nofoo`	Sets boolean values `foo=true | foo=false`.

Examples In the following examples, *prtr* can be any name up to 14 characters and can be the same name as the `ping(1M)` name.

EXAMPLE 1 Configuring an HP Postscript Printer with a Jet Direct Network Interface

The following example configures an HP postscript printer with a jet direct network interface:

```
example# lpadmin -p prtr -v /dev/null -m netstandard \
          -o dest=ping_name_of_prtr:9100 -o protocol=tcp -T PS -I \
           postscript
example# enable prtr
example# accept prtr
```

EXAMPLE 2 Configuring a Standard Postscript Network Printer

The following example configures a standard postscript network printer:

```
example# lpadmin -p prtr -v /dev/null -m netstandard \
         -o dest=ping_name_of_prtr -T PS -I postscript
example# enable prtr
example# accept prtr
```

Exit Status The following exit values are returned:

`0`	Successful completion.
non-zero	An error occurred.

Files

`/var/spool/lp/*`	
`/etc/lp`	
`/etc/lp/alerts/printer`	Fault handler for `lpadmin`
`/etc/printers.conf`	System printer configuration database

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu
Interface Stability	Obsolete

See Also enable(1), lp(1), lpstat(1), mail(1), stty(1), accept(1M), lpforms(1M), lpsched(1M), lpsystem(1M), ping(1M), dial(3NSL), terminfo(4), attributes(5)

System Administration Guide: Basic Administration

Notes When using lpadmin to provide access to a remote printer, remote configuration data is stored in `/etc/printers.conf`. This data includes a `bsdaddr` and a `printer-uri-supported` attribute. The data in this file can be shared through the use of a network name service or replicated across multiple systems. If the data is shared, it is important to make sure that the `bsdaddr` and `printer-uri-supported` contain hostname information that is correctly resolved on all hosts sharing this data. Also, the `printer-uri-supported` is the preferred means of accessing remote print service. The `bsdaddr` is supplied for backward compatability with Solaris 2.6-10 systems.

Name lpfilter – administer filters used with the LP print service

Synopsis

```
/usr/sbin/lpfilter -f filter-name
     {- | -i | -l | -x | -F pathname}
```

Description The `lpfilter` command is used to add, change, delete, or list a filter used with the LP print service. These filters convert the content of a file to have a content type acceptable to a printer.

Options Arguments consist of the `-f`*filter-name* option and exactly one of the arguments appearing within braces (`{ }`) in the SYNOPSIS.

–	Adds or changes a filter as specified from standard input. The format of the input is specified below. If `-f all` is specified with the – option, the specified change is made to all existing filters. This is not useful.
`-f` *filter-name*	Specifies the *filter-name* of the filter to be added, changed, reset, deleted, or listed. The filter name `all` is a special filter name defined below. The `-f` option is required.
`-F` *pathname*	Adds or changes a filter as specified by the contents of the file *pathname*. The format of the file's contents is specified below. If `-f all` is specified with the `-F` option, the specified change is made to all existing filters. This is not useful.
`-i`	Resets a filter to its default settings. Using `-f all` with the `-i` option restores all filters for which predefined settings are available to their original settings.
`-l`	Lists a filter description. Using `-f all` with the `-l` option produces a list of all filters.
`-x`	Deletes a filter. Using `-f all` with the `-x` option results in all filters being deleted.

Usage

Adding or Changing a Filter

The filter named in the `-f` option is added to the filter table. If the filter already exists, its description is changed to reflect the new information in the input.

When – is specified, standard input supplies the filter description. When `-F` is specified, the file *pathname* supplies the filter description. One of these two options must be specified to add or change a filter.

When an existing filter is changed with the `-F` or – option, lines in the filter description that are not specified in the new information are not changed. When a new filter is added with this command, unspecified lines receive default values. See below.

Filters are used to convert the content of a request from its initial type into a type acceptable to a printer. For a given print request, the LP print service knows the following:

- The content type of the request (specified by `lp -T` or determined implicitly).
- The name of the printer (specified by `lp -d`).
- The printer type (specified by `lpadmin -T`).

  The printer type is intended to be a printer model, but some people specify it with a content type even though `lpadmin -I` is intended for this purpose.
- The content types acceptable to the printer (specified by `lpadmin -I`).

  The values specified by the `lpadmin -T` are treated as if they were specified by the `-I` option as well.
- The modes of printing asked for by the originator of the request (specified by various options to `lp`).

The system uses the above information to construct a list of one or more filters that converts the document's content type into a content type acceptable to the printer and consumes all `lp` arguments that invoke filters (`-y` and `-P`).

The contents of the file (specified by the `-F` option) and the input stream from standard input (specified by –) must consist of a series of lines, such that each line conforms to the syntax specified by one of the seven lines below. All lists are comma or space separated. Each item contains a description.

`Input types:` *content-type-list*
`Output types:` *content-type-list*
`Printer types:` *printer-type-list*
`Printers:` *printer-list*
`Filter type:` *filter-type*
`Command:` *shell-command*
`Options:` *template-list*

`Input types` — This gives the content types that can be accepted by the filter. The default is `any`. The document content type must be a member of this list for the initial filter in the sequence.

`Output types` — This gives the content types that the filter can produce from any of the input (content) types. The default is `any`. The intersection of the output types of this list and the content types acceptable to the printer (from `lpadmin -I` and `lpadmin -T`) must be non-null for the last filter in the sequence. For adjacent filters in the sequence, the intersection of output types of one and the input types of the next must be non-null.

`Printer types` — This gives the printer types for which this printer can be used. The LP print service will restrict the use of the filter to these printer types (from `lpadmin -T`). The default is `any`.

`Printers` This gives the names of the printers for which the filter can be used. The LP print service will restrict the use of the filter to just the printers named. The default is `any`.

`Filter type` This marks the filter as a `slow` filter or a `fast` filter. Slow filters are generally those that take a long time to convert their input (that is, minutes or hours). They are run before the job is scheduled for a printer, to keep the printers from being tied up while the filter is running. If a listed printer is on a remote system, the filter type for it must have the value `slow`. That is, if a client defines a filter, it must be a slow filter. Fast filters are generally those that convert their input quickly (that is, faster than the printer can process the data), or those that must be connected to the printer when run. Fast filters will be given to the interface program to run while connected to the physical printer.

`Command` This specifies which program to run to invoke the filter. The full program pathname as well as fixed options must be included in the *shell-command*; additional options are constructed, based on the characteristics of each print request and on the `Options` field. A command must be given for each filter. The command must accept a data stream as standard input and produce the converted data stream on its standard output. This allows filter pipelines to be constructed to convert data not handled by a single filter.

`Options` This is a comma-separated list of templates used by the LP print service to construct options to the filter from the characteristics of each print request listed in the table later. The `-y` and `-P` arguments to the `lp` command cause a filter sequence to be built even if there is no need for a conversion of content types.

In general, each template is of the following form:

keyword pattern = *replacement*

The *keyword* names the characteristic that the template attempts to map into a filter-specific option; each valid *keyword* is listed in the table below.

A *pattern* is one of the following: a literal pattern of one of the forms listed in the table, a single asterisk (`*`), or a regular expression. If *pattern* matches the value of the characteristic, the template fits and is used to generate a filter-specific option. The *replacement* is what will be used as the option.

Regular expressions are the same as those found on the `regexp(5)` manual page. This includes the `\(...\)` and `\`*n* constructions, which can be used to extract portions of the *pattern* for copying into the *replacement*, and the `&`, which can be used to copy the entire *pattern* into the *replacement*.

The *replacement* can also contain a *; it too, is replaced with the entire *pattern*, just like the & of regexp(5).

The keywords are:

lp Option	Characteristic	*keyword*	Possible *patterns*
-T	Content type (input)	INPUT	content-type
Not applicable	Content type (output)	OUTPUT	content-type
not applicable	Printer type	TERM	printer-type
-d	Printer name	PRINTER	*printer-name*
-f, -o cpi=	Character pitch	CPI	integer
-f, -o lpi=	Line pitch	LPI	integer
-f, -o length=	Page length	LENGTH	integer
-f, -o width=	Page width	WIDTH	integer
-P	Pages to print	PAGES	page-list
-S	Character set Print wheel	CHARSET CHARSET	character-set-name print-wheel-name
-f	Form name	FORM	form-name
-y	Modes	MODES	mode
-n	Number of copies	COPIES	*integer*

Resetting a Filter to Defaults If the filter named is one originally delivered with the LP print service, the -i option restores the original filter description.

Deleting a Filter The -x option is used to delete the filter specified in filter-name from the LP filter table.

Listing a Filter Description The -l option is used to list the description of the filter named in filter-name. If the command is successful, the following message is sent to standard output:

```
Input types: content-type-list
Output types: content-type-list
Printer types: printer-type-list
```

```
Printers: printer-list
Filter type: filter-type
Command: shell-command
Options: template-list
```

If the command fails, an error message is sent to standard error.

Large File Behavior

See largefile(5) for the description of the behavior of `lpfilter` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples

EXAMPLE 1 Printing with the landscape option

For example, the template

```
MODES landscape = -l
```

shows that if a print request is submitted with the `-y landscape` option, the filter will be given the option `-l`.

EXAMPLE 2 Selecting the printer type

As another example, the template

```
TERM * = -T *
```

shows that the filter will be given the option `-T` *printer-type* for whichever *printer-type* is associated with a print request using the filter.

EXAMPLE 3 Using the keywords table

Consider the template

```
MODES prwidth\=\(.*\) = -w\1
```

Suppose a user gives the command

```
lp -y prwidth=10
```

From the table above, the LP print service determines that the `-y` option is handled by a `MODES` template. The `MODES` template here works because the pattern prwidth=) matches the prwidth=10 given by the user. The replacement -w1 causes the LP print service to generate the filter option `-w10`. If necessary, the LP print service will construct a filter pipeline by concatenating several filters to handle the user's file and all the print options. See sh(1) for a description of a pipeline. If the print service constructs a filter pipeline, the `INPUT` and `OUTPUT` values used for each filter in the pipeline are the types of input and output for that filter, not for the entire pipeline.

Exit Status

The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpsu

See Also `lp(1)`, `sh(1)`, `lpadmin(1M)`, `attributes(5)`, `largefile(5)`, `regexp(5)`

System Administration Guide: Basic Administration

Notes If the `lp` command specifies more than one document, the filtering chain is determined by the first document. Other documents may have a different format, but they will print correctly only if the filter chain is able to handle their format.

Name lpforms – administer forms used with the LP print service

Synopsis

```
lpforms -f form-name option

lpforms -f form-name -A alert-type [-P paper-name [-d]]
     [-Q requests] [-W minutes]
```

Description The `lpforms` command administers the use of preprinted forms, such as company letterhead paper, with the LP print service. A form is specified by its *form-name*. Users may specify a form when submitting a print request (see `lp(1)`). The argument `all` can be used instead of *form-name* with either of the command lines shown above. The first command line allows the administrator to add, change, and delete forms, to list the attributes of an existing form, and to allow and deny users access to particular forms. The second command line is used to establish the method by which the administrator is alerted that the form *form-name* must be mounted on a printer.

Options The following options are supported:

`-f` *form-name* Specify a form.

The first form of `lpforms` requires that one of the following *option*s (—, `-l`, `-F`, `-x`) must be used:

`-F` *pathname* To add or change form *form-name*, as specified by the information in *pathname*.

— To add or change form *form-name*, as specified by the information from standard input.

`-l` To list the attributes of form *form-name*.

`-x` To delete form *form-name* (this option must be used separately; it may not be used with any other option).

The second form of the `lpforms` command requires the `-A` *alert-type* option. The other options are optional.

`-A` *alert-type* Defines an alert to mount the form when there are queued jobs which need it.

`-P` *paper-name* [`-d`] Specify the paper name when creating the form. If `-d` is specified, this paper is the default.

`-Q` *requests* An alert will be sent when a certain number of print requests that need the form are waiting.

`-W` *minutes* An alert will be sent at intervals specified by minutes.

Usage

Adding or Changing a Form

The `-F` *pathname* option is used to add a new form, *form-name*, to the LP print service, or to change the attributes of an existing form. The form description is taken from *pathname* if the `-F` option is given, or from the standard input if the – option is used. One of these two options must be used to define or change a form.

pathname is the path name of a file that contains all or any subset of the following information about the form.

`Page length:` *scaled-decimal-number1*
`Page width:` *scaled-decimal-number2*
`Number of pages:` *integer*
`Line pitch:` *scaled-decimal-number3*
`Character pitch:` *scaled-decimal-number4*
`Character set choice:` *character-set/print-wheel* `[mandatory]`
`Ribbon color:` *ribbon-color*
`Comment:`
comment
`Alignment pattern: [`*content-type*`]`
content

The term "scaled-decimal-number" refers to a non-negative number used to indicate a unit of size. The type of unit is shown by a "trailing" letter attached to the number. Three types of scaled decimal numbers can be used with the LP print service: numbers that show sizes in centimeters (marked with a trailing `c`); numbers that show sizes in inches (marked with a trailing `i`); and numbers that show sizes in units appropriate to use (without a trailing letter); lines, characters, lines per inch, or characters per inch.

Except for the last two lines, the above lines may appear in any order. The `Comment:` and *comment* items must appear in consecutive order but may appear before the other items, and the `Alignment pattern:` and the *content* items must appear in consecutive order at the end of the file. Also, the *comment* item may not contain a line that begins with any of the key phrases above, unless the key phrase is preceded with a > sign. Any leading > sign found in the *comment* will be removed when the comment is displayed. There is no case distinction among the key phrases.

When this command is issued, the form specified by *form-name* is added to the list of forms. If the form already exists, its description is changed to reflect the new information. Once added, a form is available for use in a print request, except where access to the form has been restricted, as described under the `-u` option. A form may also be allowed to be used on certain printers only.

A description of each form attribute is below:

`Page length` and `Page Width`
: Before printing the content of a print request needing this form, the generic interface program provided with the LP print service will initialize the physical printer to handle pages *scaled-decimal-number1* long, and

	scaled-decimal-number2 wide using the printer type as a key into the terminfo(4) database. The page length and page width will also be passed, if possible, to each filter used in a request needing this form.
`Number of pages`	Each time the alignment pattern is printed, the LP print service will attempt to truncate the *content* to a single form by, if possible, passing to each filter the page subset of 1-*integer*.
`Line pitch` and `Character pitch`	Before printing the content of a print request needing this form, the interface program provided with the LP print service will initialize the physical printer to handle these pitches, using the printer type as a key into the terminfo(4) database. Also, the pitches will be passed, if possible, to each filter used in a request needing this form. *scaled-decimal-number3* is in lines-per-centimeter if a `c` is appended, and lines-per-inch otherwise; similarly, *scaled-decimal-number4* is in characters-per-centimeter if a `c` is appended, and characters-per-inch otherwise. The character pitch can also be given as `elite` (12 characters-per-inch), `pica` (10 characters-per-inch), or `compressed` (as many characters-per-inch as possible).
`Character set choice`	When the LP print service alerts an administrator to mount this form, it will also mention that the print wheel *print-wheel* should be used on those printers that take print wheels. If printing with this form is to be done on a printer that has selectable or loadable character sets instead of print wheels, the interface programs provided with the LP print service will automatically select or load the correct character set. If `mandatory` is appended, a user is not allowed to select a different character set for use with the form; otherwise, the character set or print wheel named is a suggestion and a default only.
`Ribbon color`	When the LP print service alerts an administrator to mount this form, it will also mention that the color of the ribbon should be *ribbon-color*.
`Comment`	The LP print service will display the *comment* unaltered when a user asks about this form (see lpstat(1)).

`Alignment pattern`	When mounting this form, an administrator can ask for the *content* to be printed repeatedly, as an aid in correctly positioning the preprinted form. The optional *content-type* defines the type of printer for which *content* had been generated. If *content-type* is not given, `simple` is assumed. Note that the *content* is stored as given, and will be readable only by the user `lp`.

When an existing form is changed with this command, items missing in the new information are left as they were. When a new form is added with this command, missing items will get the following defaults:

```
Page Length: 66
Page Width: 80
Number of Pages: 1
Line Pitch: 6
Character Pitch: 10
Character Set Choice: any
Ribbon Color: any
```

Deleting a Form

LP print service: The `-x` option is used to delete the form *form-name* from the LP print service.

Listing Form Attributes

The `-l` option is used to list the attributes of the existing form *form-name*. The attributes listed are those described under `Adding and Changing a Form,` above. Because of the potentially sensitive nature of the alignment pattern, only the administrator can examine the form with this command. Other people may use the `lpstat(1)` command to examine the non-sensitive part of the form description.

Allowing and Denying Access to a Form

The `-u` option, followed by the argument `allow:`*login-ID-list* or `-u deny:`*login-ID-list* lets you determine which users will be allowed to specify a particular form with a print request. This option can be used with the `-F` or – option, each of which is described above under `Adding or Changing a Form`.

The *login-ID-list* argument may include any or all of the following constructs:

login-ID	A user on any system
system_name`!`*login-ID*	A user on system *system_name*
system_name`!all`	All users on system *system_name*
`all!`*login-ID*	A user on all systems
`all`	All users on all systems

The LP print service keeps two lists of users for each form: an "allow-list" of people allowed to use the form, and a "deny-list" of people that may not use the form. With the `-u allow` option, the users listed are added to the allow-list and removed from the deny-list. With the `-u deny`

option, the users listed are added to the deny-list and removed from the allow-list. (Both forms of the `-u` option can be run together with the `-F` or the – option.)

If the allow-list is not empty, only the users in the list are allowed access to the form, regardless of the content of the deny-list. If the allow-list is empty but the deny-list is not, the users in the deny-list may not use the form, (but all others may use it). All users can be denied access to a form by specifying `-f deny:all`. All users can be allowed access to a form by specifying `-f allow:all`. (This is the default.)

Setting an Alert to Mount a Form

The `-f` *form-name* option is used with the `-A` *alert-type* option to define an alert to mount the form when there are queued jobs which need it. If this option is not used to arrange alerting for a form, no alert will be sent for that form.

The method by which the alert is sent depends on the value of the *alert-type* argument specified with the `-A` option. The *alert-types* are:

`mail` Send the alert message using the `mail` command to the administrator.

`write` Write the message, using the `write` command, to the terminal on which the administrator is logged in. If the administrator is logged in on several terminals, one is arbitrarily chosen.

`quiet` Do not send messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the form *form-name* has been mounted and subsequently unmounted, messages will again be sent when the number of print requests reaches the threshold specified by the `-Q` option.

`showfault` Attempt to execute a form alert handler on each system that has a print job for that form in the queue. The fault handler is `/etc/lp/alerts/form`. It is invoked with three parameters: *form_name*, `date`, *file_name*. *file_name* is the name of a file containing the form alert message.

`none` Do not send messages until the `-A` option is given again with a different *alert-type* (other than `quiet`).

shell-command Run the *shell-command* each time the alert needs to be sent. The shell command should expect the message in standard input. If there are blank spaces embedded in the command, enclose the command in quotes. Note that the `mail` and `write` values for this option are equivalent to the values `mail` *login-ID* and `write` *login-ID* respectively, where *login-ID* is the current name for the administrator. This will be the login name of the person submitting this command unless he or she has used the `su` command to change to another login-ID. If the `su` command has been used to change the user ID, then the *user-name* for the new ID is used.

`list` Display the type of the alert for the form on standard output. No change is made to the alert.

The message sent appears as follows:

```
The form form-name
needs to be mounted on the printer(s): printer (integer1 requests).
integer2 print requests await this form.
Use the ribbon-color ribbon. Use the print-wheel print wheel, if appropriate.
```

The printers listed are those that the administrator has specified as candidates for this form. The number *integer1* listed next to each printer is the number of requests eligible for the printer. The number *integer2* shown after the list of printers is the total number of requests awaiting the form. It will be less than the sum of the other numbers if some requests can be handled by more than one printer. The *ribbon-color* and *print-wheel* are those specified in the form description. The last line in the message is always sent, even if none of the printers listed use print wheels, because the administrator may choose to mount the form on a printer that does use a print wheel.

Where any color ribbon or any print wheel can be used, the statements above will read:

```
Use any ribbon.
Use any print-wheel.
```

If *form-name* is `any`, the *alert-type* defined in this command applies to any form for which an alert has not yet been defined. If *form-name* is `all`, the *alert-type* defined in this command applies to all forms.

If the `-W` *minutes* option is not given, the default procedure is that only one message will be sent per need to mount the form. Not specifying the `-W` option is equivalent to specifying `-W once` or `-W 0`. If *minutes* is a number greater than `0`, an alert will be sent at intervals specified by *minutes*.

If the `-Q` *requests* option is also given, the alert will be sent when a certain number (specified by the argument *requests*) of print requests that need the form are waiting. If the `-Q` option is not given, or the value of *requests* is `1` or `any` (which are both the default), a message is sent as soon as anyone submits a print request for the form when it is not mounted.

Listing the Current Alert

The `-f` option, followed by the `-A` option and the argument `list` is used to list the *alert-type* that has been defined for the specified form *form-name*. No change is made to the alert. If *form-name* is recognized by the LP print service, one of the following lines is sent to the standard output, depending on the type of alert for the form.

- `When` *requests* `requests are queued: alert with` *shell-command* `every` *minutes* `minutes`
- `When` *requests* `requests are queued: write to` *user-name* `every` *minutes* `minutes`
- `When` *requests* `requests are queued: mail to` *user-name* `every` *minutes* `minutes`
- `No alert`

The phrase `every` *minutes* `minutes` is replaced with `once` if *minutes* (`-W`*minutes*) is 0.

Terminating an Active Alert

The `-A quiet` option is used to stop messages for the current condition. An administrator can use this option to temporarily stop receiving further messages about a known problem. Once the form has been mounted and then unmounted, messages will again be sent when the number of print requests reaches the threshold *requests*.

Removing an Alert Definition

No messages will be sent after the `-A none` option is used until the `-A` option is given again with a different *alert-type*. This can be used to permanently stop further messages from being sent as any existing alert definition for the form will be removed.

Large File Behavior

See `largefile(5)` for the description of the behavior of `lpforms` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status

The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Files

`/etc/lp/alerts/form` Fault handler for `lpform`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpsu

See Also

`lp(1)`, `lpstat(1)`, `lpadmin(1M)`, `terminfo(4)`, `attributes(5)`, `largefile(5)`

System Administration Guide: Basic Administration

Name lpget – get printing configuration

Synopsis

```
lpget [-k key] [destination... | list]
```

Description The `lpget` utility reads printing configuration information from the configuration databases in `$HOME/.printers`, `/etc/printers.conf`, `printers.conf.byname`, and `printers.org_dir` printer. This information, called a *configuration report*, is displayed to the standard output. See `printers(4)` and `printers.conf(4)` for information about the printer configuration databases.

`lpget` displays a configuration report for all keys for the specified destination or destinations by default. Use the `-k` option to display a configuration report for specific keys. Use the `list` operand to display a configuration report for all configured destinations.

Options The following option is supported:

`-k` *key* Displays a configuration report for *key*. See `printers.conf(4)` for information about specifying *key*.

Operands The following operands are supported:

destination Displays a configuration report for *destination*. Destination can be either a printer of a class of printers. See `lpadmin(1M)`. Specify *destination* using atomic or POSIX-style (*server*:*destination*) names. See `printers.conf(4)` for information regarding the naming conventions for atomic names and `standards(5)` for information concerning POSIX.

`list` Displays a configuration report for all configured destinations.

Examples

EXAMPLE 1 Displaying a Configuration Report for the `bsdaddr` Key

The following example displays a configuration report for the `bsdaddr` key for printer `catalpa`.

```
example% lpget -k bsdaddr catalpa
```

EXAMPLE 2 A Configuration Report for all Keys for all Configured Destinations

The following example displays a configuration report for all keys for all configured destinations.

```
example% lpget list
```

Exit Status The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Files

`/etc/printers.conf`	System printer configuration database.
`$HOME/.printers`	User-configurable printer database.
`printers.conf.byname`	NIS version of `/etc/printers.conf`.
`printers.org_dir`	NIS+ version of `/etc/printers.conf`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu
Stability Level	Stable

See Also `ldap(1)`, `lp(1)`, `lpc(1B)`, `lpq(1B)`, `lpr(1B)`, `lpstat(1)`, `lpadmin(1M)`, `lpset(1M)`, `printers(4)`, `printers.conf(4)`, `attributes(5)`, `standards(5)`

System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)

Notes Be mindful of the following if the LDAP database is used as the name service. If the `ldapclient(1M)` server is a replica LDAP server, LDAP printer database updates may not appear immediately, as the replica server may not not have been updated by the master server and can be out of sync. For example, a printer that you deleted by using `lpset(1M)` may still appear in the printer list you display with `lpget` until the replica is updated from the master. Replica servers vary as to how often they are updated from the master. Refer to the *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for more information on LDAP replication.

Name lpmove – move print requests

Synopsis

```
lpmove [request-ID] destination
lpmove source destination
```

Description The `lpmove` command moves print requests queued by `lp(1)` or `lpr(1B)` between destinations.

The first form of `lpmove` moves specific print requests (*request-ID*) to a specific *destination*.

The second form of the `lpmove` command moves all print requests from one destination (*destination1*) to another (*destination2*). This form of `lpmove` also rejects new print requests for *destination1*.

`lpmove` moves individual requests or entire queues only within an instance of a print service, not between a local and a remote queues or local queues on different instances of a print service. Requests can only be moved if the print service or protocol supports it. The LP print server and IPP print protocol both support moving requests between queues. The BSD print protocol does not.

When moving requests, `lpmove` does not check the acceptance status of the destination to which the print requests are being moved (see `accept(1M)`). `lpmove` does not move requests that have options (for example, content type or requiring a special form) that cannot be handled by the new destination.

Operands The following operands are supported:

request-ID The specific print request to be moved. Specify *request-ID* as the identifier associated with a print request as reported by `lpstat`. See `lpstat(1)`.

destination The name of the printer or class of printers (see `lpadmin(1M)`) to which `lpmove` moves a *specified* print request. Specify *destination* using atomic, URI-style (*scheme*://*endpoint*), POSIX-style (*server*:*destination*) syntax.

source The name of the destination from which `lpmove` moves *all* print requests. Specify *destination* using atomic, URI-style (*scheme*://*endpoint*), POSIX-style (*server*:*destination*) syntax.

See `printers.conf(4)` for information regarding the naming conventions for atomic names and `standards(5)` for information regarding POSIX.

Exit Status The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Files

`/etc/printers.conf`	System printer configuration database
`$HOME/.printers`	User-configurable printer database
`ou=printers`	LDAP version of `/etc/printers.conf`

`printers.conf.byname` NIS version of `/etc/printers.conf`

`printers.org_dir` NIS+ version of `/etc/printers.conf`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlp-cmds
Interface Stability	Obsolete

See Also `lp(1)`, `lpr(1B)`, `lpstat(1)`, `accept(1M)`, `lpadmin(1M)`, `lpsched(1M)`, `printers.conf(4)`, `attributes(5)`, `standards(5)`

System Administration Guide: Printing

Notes When IPP is in use, the user is prompted for a passphsrase if the remote print service is configured to require authentication.

Name lpsched – start the LP print service

Synopsis

```
lpsched [-f num_filters] [-n num_notifiers] [-p fd_limit]
     [-r reserved_fds]
```

Description The `lpsched` command starts or restarts the LP print service.

The `lpshut` command stops the LP print service. Printers that are restarted using `lpsched` reprint (in their entirety) print requests that were stopped by `lpshut`. See `lpshut(1M)`.

It is recommended that you start and stop the LP print service using `svcadm(1M)`. See `NOTES`.

Options The following options are supported:

`-f` *num_filters* Specifies the number of concurrent slow filters that may be run on a print server. A default value of `1` is used if none is specified. Depending on server configuration, a value of `1` may cause printers to remain idle while there are jobs queued to them.

`-n` *num_notifiers* Specifies the number of concurrent notification processes that can run on a print server. A default value of `1` is used when none is specified.

`-p` *fd_limit* Specifies the file descriptor resource limit for the `lpsched` process. A default value of `4096` is used if none is specified. On extremely large and active print servers, it may be necessary to increase this value.

`-r` *reserved_fds* Specifies the number of file descriptors that the scheduler reserves for internal communications under heavy load. A default value of `2` is used when none is specified. It should not be necessary to modify this value unless instructed to do so when troubleshooting problems under high load.

Exit Status The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Files `/var/spool/lp/*` LP print queue.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpsu

See Also `lp(1)`, `svcs(1)`, `lpstat(1)`, `lpadmin(1M)`, `lpmove(1M)`, `lpshut(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

System Administration Guide: Basic Administration

Notes The `lpsched` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/application/print/server
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name lpset – set printing configuration in /etc/printers.conf or other supported databases

Synopsis

```
lpset [-n system | nisplus | fnsldap] [-x]
     [ [-D binddn] [-w passwd] [-h ldaphost]]
     [-a key=value] [-d key] destination
```

Description The `lpset` utility sets printing configuration information in the system configuration databases. Use `lpset` to create and update printing configuration in `/etc/printers.conf`, or `printers.org_dir` (NIS+). See `nsswitch.conf(4)` and `printers.conf(4)`.

Only a superuser or a member of Group 14 may execute `lpset`.

Options The following options are supported:

`-n system|nisplus|ldap` Create or update the configuration information for the *destination* entry in `/etc/printers.conf`, `printers.org_dir` (NIS+), or LDAP printer contexts. `system` specifies that the information is created or updated in `/etc/printers.conf`. `nisplus` specifies that the information is created or updated in the `printers.org_dir` NIS+ table. `ldap` specifies that the information is written to an LDAP server. See Notes.

If `-n` is not specified, `system` is the default.

`-x` Remove all configuration for the *destination* entry from the database specified by the `-n` option.

`-a` *key*=*value* Configure the specified *key*=*value* pair for the *destination*. See `printers.conf(4)` for information regarding the specification of *key*=*value* pairs.

`-d` *key* Delete the configuration option specified by *key* for the *destination* entry. See `printers.conf(4)` for information regarding the specification of *key* and *key*=*value* pairs.

`-D binddn` Use the distinguished name (DN) `binddn` to bind to the LDAP directory server.

`-w passwd` Use `passwd` as the password for authentication to the LDAP directory server.

`-h ldaphost` Specify an alternate host on which the LDAP server is running. This option is only used when `ldap` is specified as the naming service. If this option is not specified, the default is the current host system.

Operands The following operand is supported:

destination Specifies the entry in `/etc/printers.conf`, `printers.org_dir`, or LDAP, in which to create or modify information. *destination* names a printer of class of

printers. See lpadmin(1M). Each entry in printers.conf describes one destination. Specify *destination* using atomic names. POSIX-style destination names are not acceptable. See printers.conf(4) for information regarding the naming conventions for atomic names and standards(5) for information regarding POSIX.

Examples

EXAMPLE 1 Removing All Existing Printing Configuration Information

The following example removes all existing printing configuration information for destination dogs from /etc/printers.conf:

```
example% lpset -x dogs
```

EXAMPLE 2 Setting a key=value Pair

The following example sets the user-equivalence=true *key=value* pair for destination tabloid in the NIS+ context:

```
example% lpset -n nisplus -a user-equivalence=true tabloid
```

EXAMPLE 3 Setting a key=value Pair in LDAP

```
example% lpset -n ldap -h ldapl.xyz.com -D "cn=Directory Manager" \
   -w passwd -a key1=value1 printer1
```

Exit Status

The following exit values are returned:

0	Successful completion.
non-zero	An error occurred.

Files

/etc/printers.conf	System configuration database.
printer.org_dir (NIS+)	NIS+ version of /etc/printers.conf.

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu
Stability Level	Stable

See Also

ldap(1), lp(1), lpc(1B), lpq(1B), lpr(1B), lpstat(1), ldapclient(1M), lpadmin(1M), lpget(1M), nsswitch.conf(4), printers(4), printers.conf(4), attributes(5), standards(5)

System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)

Notes If the `ldap` database is used, the printer administrator should be mindful of the following when updating printer information.

1. Because the domain information for the printer being updated is extracted from the `ldapclient(1M)` configuration, the LDAP server being updated must host the same domain that is used by the current `ldapclient(1M)` server.
2. If the LDAP server being updated is a replica LDAP server, the updates will be referred to the master LDAP server and completed there. The updates might be out of sync and not appear immediatedly, as the replica server may not have been updated by the master server. For example, a printer that you deleted by using `lpset` may still appear in the printer list you display with `lpget` until the replica is updated from the master. Replica servers vary as to how often they are updated from the master. See *System Administration Guide: Printing* for information on LDAP server replication.
3. Although users can use the LDAP command line utilities `ldapadd(1)` and `ldapmodify(1)` to update printer entries in the directory, the preferred method is to use `lpset`. Otherwise, if the `ldapadd` and `ldapmodify` utilities are used, the administrator must ensure that the `printer-name` attribute value is unique within the `ou=printers` container on the LDAP server. If the value is not unique, the result of modifications done using `lpset` or the Solaris Print Manager, `printmgr(1M)` may be unpredictable.

Name lpshut – stop the LP print service

Synopsis

```
lpshut
```

Description The `lpshut` command stops the LP print service.

Printers that are printing when `lpshut` is invoked stop printing. Start or restart printers using `lpsched(1M)`.

Exit Status The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

Files `/var/spool/lp/*` LP print queue.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpsu

See Also `lp(1)`, `lpstat(1)`, `lpadmin(1M)`, `lpmove(1M)`, `lpsched(1M)`, `attributes(5)`

System Administration Guide: Basic Administration

Name lpsystem – register remote systems with the print service

Description The `lpsystem` command is obsolete, and could be removed at any time. The print system no longer uses the information generated by `lpsystem`. See `lpadmin(1M)`, `lpusers(1M)` or `printers.conf(4)` for equivalent functionality.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu
Stability Level	Obsolete*

* This command could be removed at any time.

See Also `lpadmin(1M)`, `lpusers(1M)`, `printers.conf(4)`, `attributes(5)`

Name lpusers – set printing queue priorities

Synopsis

```
lpusers -d priority-level
lpusers -q priority-level -u login-ID-list
lpusers -u login-ID-list
lpusers -q priority-level
lpusers -l
```

Description The `lpusers` command sets limits to the queue priority level that can be assigned to jobs submitted by users of the LP print service.

The first form of the command (with `-d`) sets the system-wide priority default to *priority-level*, where *priority-level* is a value of 0 to 39, with 0 being the highest priority. If a user does not specify a priority level with a print request (see `lp(1)`), the default priority level is used. Initially, the default priority level is 20.

The second form of the command (with `-q` and `-u`) sets the default highest *priority-level* (0-39) that the users in *login-ID-list* can request when submitting a print request. The *login-ID-list* argument may include any or all of the following constructs:

login-ID	A user on any system
system_name`!`*login-ID*	A user on the system *system_name*
system_name`!all`	All users on system *system_name*
`all!`*login-ID*	A user on all systems
`all`	All users on all systems

Users that have been given a limit cannot submit a print request with a higher priority level than the one assigned, nor can they change a request that has already been submitted to have a higher priority. Any print requests submitted with priority levels higher than allowed will be given the highest priority allowed.

The third form of the command (with `-u`) removes any explicit priority level for the specified users.

The fourth form of the command (with `-q`) sets the default highest priority level for all users not explicitly covered by the use of the second form of this command.

The last form of the command (with `-l`) lists the default priority level and the priority limits assigned to users.

Options The following options are supported:

`-d` *priority-level*	Set the system-wide priority default to *priority-level*.

`-l`	List the default priority level and the priority limits assigned to users.
`-q` *priority-level*	Set the default highest priority level for all users not explicitly covered.
`-q` *priority-level* `-u` *login-ID-list*	Set the default highest *priority-level* that the users in *login-ID-list* can request when submitting a print request.
`-u` *login-ID-list*	Remove any explicit priority level for the specified users.

Exit Status The following exit values are returned:

`0`	Successful completion.
non-zero	An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpsu

See Also `lp(1)`, `attributes(5)`

Name lu – FMLI-based interface to Live Upgrade functions

Synopsis `/usr/sbin/lu`

Description Sun no longer recommends use of the `lu` command. The `lu` command displays a character user interface (CUI). The underlying command sequence for the CUI--typically `lucreate`, `luupgrade`, and `luactivate`--is straightforward to use. The `SEE ALSO` section below highlights the full Solaris Live Upgrade command set.

The `lu` program is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `lu` program is a Forms and Menu Language Interpreter-based user interface. (See `fmli(1)` for a description of the Forms and Menu Language Interpreter.) `lu` enables you to create and upgrade boot environments (BEs) and perform other administrative tasks on BEs. The `lu` program performs a subset of the functions provided by the Live Upgrade command-line utilities.

Users of `lu` should be aware of the following:

- `lu` is a deprecated interface. It will be replaced in the future and should not be depended on for critical functionality.
- All new Live Upgrade features are being implemented in the Live Upgrade command–line utilities. No new features are being made available in `lu`.
- The `lu` command is not internationalized. It will not be internationalized in a future release.

`lu` should be used for learning or experimenting only. For any production use or to use the full capabilities of Live Upgrade, use the Live Upgrade command-line utilities.

Invocation of the `lu` command requires root privileges.

The `lu` command accepts no arguments. After invoking `lu`, you receive a display with the following options:

Activate — Activate a boot environment. This option designates that the system boot from the specified BE upon next reboot. This option is equivalent to the command-line `luactivate(1M)` utility.

Cancel — Cancel a copy job. Live Upgrade allows you to schedule the copy, upgrade, and flash functions (all described below) at a later time. The cancel function enables you to cancel a scheduled job. This function is equivalent to the command-line `lucancel(1M)` utility.

Compare — Compare the contents of BEs. Enables you to obtain a detailed comparison of two BEs. Equivalent to the command-line `lucompare(1M)` utility.

Copy	Start/schedule a copy. Copies the contents of one BE to another. Equivalent of the command-line `lumake(1M)` utility. At any time, you can have only one Live Upgrade operation scheduled.
Create	Create a boot environment. Implements a subset of the functions performed by the command-line `lucreate(1M)` utility.
Current	Display the name of the current boot environment. Equivalent of the command-line `lucurr(1M)` utility.
Delete	Delete a boot environment. Equivalent of the command-line `ludelete(1M)` utility.
List	List the file systems of a boot environment. Equivalent of the command-line `lufslist(1M)` utility.
Rename	Change the name of a boot environment. Equivalent of the command-line `lurename(1M)` utility.
Status	List the status of all boot environments. Equivalent of the command-line `lustatus(1M)` utility.
Upgrade	Upgrade a boot environment or upgrade the OS on an inactive BE. This option enables you to upgrade to a new operating system or install new packages or patches on a specified BE. Implements a subset of the functions performed by the command-line `luupgrade(1M)` utility. Note that if you are performing an upgrade that requires more than one CD, you must use the `-i` option of `luupgrade`.
Flash	Flash a boot environment. This option enables you to install an operating system on a BE from a flash archive. You can perform the same function with `luupgrade(1M)`.
Help	Displays help information. There are also context-specific help screens for many of the options.
Exit	Exit `lu`.

Navigation You navigate through `lu`'s various screens using arrow keys and function keys (usually F2 through F9 on the keyboard of a Sun desktop system). Available key functions are displayed at the base of the `lu` screen. You can use Ctrl-F, plus a number key, to duplicate a function key. For example, press Ctrl-F and the number key `2` to duplicate the F2 key.

In a screen for a given option, you can press Esc to obtain context-specific help.

Display Issues When viewing the FMLI interface remotely, such as over a `tip` line, you might need to set the `TERM` environment variable to `VT220`. When using the FMLI interface in a CDE environment use `dtterm`, rather than `xterm`, as the value of the `TERM` variable.

The `lu` command supports only single-byte environments.

Common Functions

Most of the options listed above offer the following functions. These functions are accessible through function keys indicated at the base of the screen.

Choice — Available to you whenever you have a field that can be filled in. Pressing the Choice function key gives you a popup screen displaying a list of alternatives. For example, for options involving copying or upgrading BEs, you receive a list of available BEs. You can then use arrow and function keys to make a selection from this popup. The choice function is useful because it prevents you from selecting an invalid alternative. In our example, it prevents you from choosing a BE that is not available for a copy or upgrade operation. Such non-availability might occur when a BE is in the midst of an upgrade.

Cancel — Cancel an operation.

Save — Proceed with an operation.

Other Functions

The "Create" option, described above, offers the following functions:

Split — Split a file system. For example, you can split a `/` file system into `/`, `/usr`, and `/var`. To split a file system, you must have disk slices available on which to mount the separated file system(s). If you do not, `lu` invokes the `format(1M)` utility, in which you can use the `partition` option to create a new disk slice.

Merge — Join one or more file systems with its (or their) parent file system. For example, using a source BE that has separate `/`, `/usr`, and `/var` file systems, you can merge these file systems under `/` on a target BE.

Files

`/etc/lutab` — list of BEs on the system

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also

`luactivate(1M)`, `lucancel(1M)`, `lucompare(1M)`, `lucreate(1M)`, `lucurr(1M)`, `ludelete(1M)`, `ludesc(1M)`, `lufslist(1M)`, `lumake(1M)`, `lumount(1M)`, `lurename(1M)`, `lustatus(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`

Solaris Installation Guide

Warnings

The `lu` command is a deprecated interface. See DESCRIPTION.

Name luactivate – activate a boot environment

Synopsis

```
/usr/sbin/luactivate [-l error_log] [-o outfile] [-s]
     [BE_name] [-X]
```

Description The `luactivate` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `luactivate` command, with no arguments, displays the name of the boot environment (BE) that will be active upon the next reboot of the system. When an argument (a BE) is specified, `luactivate` activates the specified BE.

`luactivate` activates a BE by making the BE's root partition bootable. On an x86 machine, this might require that you take steps following the completion of `luactivate`. If so, `luactivate` displays the correct steps to take.

To successfully activate a BE, that BE must meet the following conditions:

- The BE must have a status of "complete," as reported by `lustatus(1M)`.
- If the BE is not the current BE, you cannot have mounted the partitions of that BE (using `lumount(1M)` or `mount(1M)`).
- The BE you want to activate cannot be involved in an `lucompare(1M)` operation.

After activating a specified BE, `luactivate` displays the steps to be taken for fallback in case of any problem on the next reboot. Make note of these instructions and follow them exactly, if necessary.

Note – Before booting a new BE, you must run `luactivate` to specify that BE as active. `luactivate` performs a number of tasks, described below, that ensure correct operation of the BE. In some cases, a BE is not bootable until after you have run the command.

The `luactivate` command performs the following tasks:

- The first time you boot from a newly created BE, Live Upgrade software synchronizes this BE with the BE that was last active. (This is not necessarily the BE that was the source for the newly created BE.) *Synchronize* here means that certain system files and directories are copied from the last-active BE to the BE being booted. (See `synclist(4)`.) Live Upgrade software does not perform this synchronization after a BE's initial boot, unless you use the `-s` option, described below.
- If `luactivate` detects conflicts between files that are subject to synchronization, it issues a warning and does not perform the synchronization for those files. Activation can complete successfully, in spite of such a conflict. A conflict can occur if you upgrade one BE or another to a new operating system version or if you modify system files (for example, `/etc/passwd`) on one of the BEs.

- `luactivate` checks to see whether upgrade problems occurred. For example, packages required for the correct operation of the operating system might be missing. This package check is done for the global zone as well as all non-global zones inside of the BE. The command can issue a warning or, if a BE is incomplete, can refuse activation.
- `luactivate` determines whether the bootstrap program requires updating and takes steps to update if necessary. If a bootstrap program changed from on operating release to another, an incorrect bootstrap program might render an upgraded BE unbootable. See `installboot(1M)`.
- `luactivate` modifies the root partition ID on a Solaris x86 disk to enable multiple BEs to reside on a single disk. In this configuration, if you do not run `luactivate`, booting of the BE will fail. See `fmthard(1M)` and `dkio(7I)`.

The `luactivate` command requires root or equivalent privileges.

Options The `luactivate` command has the following options:

`-l` *error_log* Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.

`-o` *outfile* All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-s` Causes synchronization to occur (see DESCRIPTION) even if next boot of a specified BE is not the first boot of that BE. Use this option with great caution, because you might not be aware or in control of changes that might have occurred in the last-active BE.

If using `-s`, take special care when booting to an earlier release of Solaris than what is installed on the last-active BE. For example, consider that the last-active BE contains Solaris 9 and you want to activate a BE that contains Solaris 2.6. If you forced synchronization with the `-s` option, the BE containing Solaris 2.6 might be synchronized with files that, while compatible with Solaris 9, might not work under Solaris 2.6.

`-X` Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Operands *BE_name* Name of the BE to be activated.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.<num>`	Live Upgrade DTD (see `-X` option)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlu

See Also `lucancel(1M)`, `lucompare(1M)`, `lucreate(1M)`, `lucurr(1M)`, `ludelete(1M)`, `ludesc(1M)`, `lufslist(1M)`, `lumake(1M)`, `lumount(1M)`, `lurename(1M)`, `lustatus(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`, `zones(5)`

Notes After entering `init 6`, if an activated BE fails to boot, consult `/var/svc/log/rc6.log` on the current BE for possible obstacles.

Name lucancel – cancel a scheduled Live Upgrade copy/create procedure

Synopsis

```
/usr/sbin/lucancel [-l error_log] [-o outfile] [-X]
```

Description The `lucancel` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `lucancel` command cancels a boot environment (BE) creation or upgrade that was scheduled in the FMLI-based interface, `lu(1M)`, or the repopulation of a BE, scheduled with `lumake(1M)`. Note that Sun recommends against the use of the `lu` command.

`lucancel` does not cancel a job that is active (that is, is in the process of creation or repopulation).

The `lucancel` command requires root privileges.

Options The `lucancel` command has the following options:

`-l` *error_log* Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.

`-o` *outfile* All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-X` Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/etc/lutab` list of BEs on the system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also `luactivate(1M)`, `lucompare(1M)`, `lucreate(1M)`, `lucurr(1M)`, `ludelete(1M)`, `ludesc(1M)`, `lufslist(1M)`, `lumake(1M)`, `lumount(1M)`, `lurename(1M)`, `lustatus(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`

Name lucompare – compare boot environments

Synopsis

```
/usr/sbin/lucompare [-i infile | -t] [-o outfile] BE_name
     [-X]

/usr/sbin/lucompare [-C file [-o outfile]] [-X]
```

Description The `lucompare` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `lucompare` command compares the contents of the current boot environment (BE) with the contents of another BE. With the `-C` option, `lucompare` compares file statistics so that you can determine which files have changed on a BE since a specified time, such as the creation time of a BE. A specified BE must be inactive and in the complete state, as reported by the `lustatus(1M)` command. Also, a BE cannot have a copy job scheduled, which is also reported by `lustatus(1M)`. A specified BE cannot have any partitions mounted with `lumount(1M)` or `mount(1M)`.

For each file system defined for a specified BE, `lucompare` compares all files with the files with the same pathnames in the current BE. The files present in the active BE, but not in the specified BE, and vice-versa, are reported. You also have the option to specify a list of files to be compared.

If you specify the `-C` option, instead of doing an absolute comparison of the current BE with a target BE, `lucompare` compares the files in a specified BE with the list of files recorded in a file. When a BE is created, `lucreate(1M)` creates a file named :<*BE_name*> in `/etc/lu/compare`. You can use the `-C` option to compare the files in a specified BE to this snapshot in `/etc/lu/compare` or you can compare the BE to a file previously created with the `-o` option. Comparing a BE to its own snapshot in `/etc/lu/compare` enables you to determine which files have changed on the BE since its creation.

By default, the output of `lucompare` is written to stdout. With the `-C` option, you must use the `-o` option to specify an output file. The output for `lucompare` is a list of files that differ in permissions, owner, group, or sum, along with the reason for difference. The output format is shown below:

```
> active BE
< BE_name
reason
> file_name:owner:group:number_of_links:mode:type: size
or major_minor number:checksum
< file_name:owner:group:number_of_links:mode:type: size
or major_minor number:checksum
```

The above fields are obtained from the `stat(2)` structure of the file.

The *type* field can be one of the following:

`SYMLINK` symbolic link

FIFO	FIFO file
CHRSPC	character special
BLKSPC	block special
DIR	directory
REGFIL	regular file
UNKNOW	unknown file type

`lucompare` computes checksums only if the file on the specified BE matches its counterpart on the active BE in all of the fields described above. If the checksums differ, `lucompare` appends the differing checksums to the entries for the compared files.

The `lucompare` command requires root privileges.

Options The `lucompare` command has the following options:

`-C` *file*	Compare file statistics of BE with those recorded in *file*. *file* can be the snapshot created at BE creation time, `/etc/lu/compare/:`<*BE_name*>, or a file previously created with the `-o` option. You must use the `-o` option with this option.
`-i` *infile*	Compare files listed in *infile*. The files to be compared should be an absolute filename. If the entry in the file is a directory, then comparison is recursive with respect to the directory. Mutually exclusive of `-t`.
`-o` *outfile*	Send output of differences to *outfile*. You must use this option if you use `-C`.
`-t`	Compare only nonbinary files. This is achieved by performing a `file(1)` command on each file in the tree walk and only comparing text files. Mutually exclusive of `-i`.
`-X`	Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Operands

BE_name	Name of the BE to which the active BE will be compared. You cannot specify a BE that is involved in another Live Upgrade operation, or specify a BE for which you have mounted partitions (using `lumount(1M)` or `mount(1M)`).

Examples **EXAMPLE 1** Checking Differences Since BE Creation

The following command lists the differences in the BE `s8u5` between its creation time and the present.

```
# lucompare -C /etc/lu/compare/:s8u5 -o /var/tmp/compare.out s8u5
```

EXAMPLE 1 Checking Differences Since BE Creation *(Continued)*

Note that `/etc/lu/compare/:s8u5` is the file created by `lucreate` upon creation of a BE. The list of differences is sent to `/var/tmp/compare.out`.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.`*<num>*	Live Upgrade DTD (see `-X` option)

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also

`luactivate(1M)`, `lucancel(1M)`, `lucreate(1M)`, `lucurr(1M)`, `ludelete(1M)`, `ludesc(1M)`, `lufslist(1M)`, `lumake(1M)`, `lumount(1M)`, `lurename(1M)`, `lustatus(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`

Notes

The `lucompare` command makes no attempt to reconcile any differences it detects between BEs.

Name lucreate – create a new boot environment

Synopsis

```
/usr/sbin/lucreate [-A BE_description] [-c BE_name]
     [-C ( boot_device | - )] -n BE_name
     [-f exclude_list_file] [-I] [-l error_log]
     [-o outfile] [-s ( - | source_BE_name )]
     [ [-M slice_list_file [-M]...]
     [-m mount_point:device [,volume]:fs_options[:zonename] [-m...]]] |
     [-p zfs_root_pool]
     [-x exclude [-x]...] [-X] [-y include [-y]...]
     [-Y include_list_file] [-z filter_list]
```

Description The `lucreate` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature and its associated terminology.

The `lucreate` command offers a set of command line options that enable you to perform the following functions:

- Create a new boot environment (BE), based on the current BE.
- Create a new BE, based on a BE other than the current BE.
- Join or separate the file systems of a BE onto a new BE. For example, join `/var` and `/opt` under `/`, or separate these directories to be mounted under different disk slices.
- Specify separate file systems belonging to a particular zone inside of the new BE. (See `zones(5)`.)
- Create the file systems for a BE, but leave those file systems unpopulated.

If `lucreate` is invoked without the `-m`, `-M`, or `-p` options (described below), it brings up an FMLI-based interface that provides curses-based screens for Live Upgrade administration. Note that the FMLI-based interface does not support all of the Live Upgrade features supported by the command-line version of `lucreate`. Also, Sun is not committed to ongoing development of the FMLI-based interface.

With the `-p` option, `lucreate` supports the creation of BEs on ZFS file systems. The source BE can be a UFS root file system on a disk slice or a ZFS file system in an existing ZFS storage pool. `lucreate` provides a convenient means of migrating a BE from a UFS root file system to a ZFS root file system. You cannot create a BE on a UFS file system from a source BE on a ZFS file system.

The creation of a BE includes selecting the disk or device slices for all the mount points of the BE. Slices can be physical disks or logical devices, such as Solaris Volume Manager volumes. You can also change the mount points of the BE using the SPLIT and MERGE functions of the FMLI-based configuration screen.

Upon successful creation of a BE, you can use lustatus(1M) to view the state of that BE and lufslist(1M) to view the BE's file systems. You can use luupgrade(1M) to upgrade the OS on that BE and luactivate(1M) to make a BE active, that is, designate it as the BE to boot from at the next reboot of the system.

Note – Before booting a new BE, you must run `luactivate` to specify that BE as active. `luactivate` performs a number of tasks that ensure correct operation of the BE. In some cases, a BE is not bootable until after you have run the command. See luactivate(1M) for a list of the operations performed by that command.

The `lucreate` command makes a distinction between the file systems that contain the OS—`/`, `/usr`, `/var`, and `/opt`—and those that do not, such as `/export`, `/home`, and other, user-defined file systems. The file systems in the first category cannot be shared between the source BE and the BE being created; they are always copied from the source BE to the target BE. By contrast, the user-defined file systems are shared by default. For Live Upgrade purposes, the file systems that contain the OS are referred to as non-shareable (or *critical*) file systems; other file systems are referred to as shareable. A non-shareable file system listed in the source BE's `vfstab` is copied to a new BE. For a shareable file system, if you specify a destination slice, the file system is copied. If you do not, the file system is shared.

When migrating from a UFS-based BE to a ZFS-based BE, you cannot migrate shared UFS file systems to ZFS. Also, when the source and destination BEs are both ZFS-based, you cannot copy shared file systems. Such file systems can only be shared.

The `lucreate` command copies all non-global zones from the current BE to the BE being created. For non-global zones residing in a non-shared file system, the new BE gets a copy of the zone in its non-shared file system. For non-global zones residing in a shared file system, `lucreate` makes a copy of the zone for the new BE in that shared file system and uses a different zonepath (see zoneadm(1M)) for the zone. The zonepath used is of the form *zonepath*`-newBE`. This prevents BEs from sharing the same non-global zone in the shared file system. When the new BE gets booted, the zone in the shared file system belonging to the new BE has its zonepath renamed to *zonepath* and the zone in the shared file system belonging to the original BE has its zonepath renamed to *zonepath*`-origBE`.

If a zone exists in a non-shared file system, the zone is automatically copied when the UFS root file system is migrated to a ZFS root file system. If a zone exists in a shared UFS file system, to migrate to a ZFS root file system, you must first upgrade the zone, as in previous Solaris releases. A zone in a non-shared file system within a ZFS BE is cloned when upgrading to a ZFS BE within the same ZFS pool.

The `lucreate` command supports a limited subset of Solaris Volume Manager functions. In particular, using `lucreate` with the `-m` option, you can:

- Create a mirror.
- Detach existing SVM concatenations from mirrors. Similarly, you can attach existing Solaris Volume Manager concatenations to mirrors. These can be mirrors that were created in Solaris Volume Manager or those created by `lucreate`.

- Create a single-slice concatenation and attach a single disk slice to it.
- Detach a single disk slice from a single-slice concatentation.
- Attach multiple single-slice concatenations to a mirror. `lucreate` can attach as many of these concatenations as are allowed by Solaris Volume Manager.

`lucreate` does not allow you to attach multiple disk slices or multiple storage devices to a concatenation. Similarly, it does not allow you to detach multiple slices or devices from a concatenation.

If you use Solaris Volume Manager volumes for boot environments, it is recommended that you use `lucreate` rather than Solaris Volume Manager commands to manipulate these volumes. The Solaris Volume Manager software has no knowledge of boot environments, whereas the `lucreate` command contains checks that prevent you from inadvertently destroying a boot environment by, for example, overwriting or deleting a Solaris Volume Manager volume.

If you have already used Solaris Volume Manager software to create complex Solaris Volume Manager volumes (for example, RAID-5 volumes), Live Upgrade will support the use of these. However, to create and manipulate these complex objects, you must use Solaris Volume Manager software. As described above, the use of Solaris Volume Manager software, rather than the `lucreate` command, entails the risk of destroying a boot environment. If you do use Solaris Volume Manager software, use `lufslist(1M)` to determine which devices are in use for boot environments.

Except for a special use of the `-s` option, described below, you must have a source BE for the creation of a new BE. By default, it is the current BE. You can use the `-s` option to specify a BE other than the current BE.

When creating a new BE on a UFS file system, `lucreate` enables you to exclude and include certain files from the source BE. You perform this inclusion or exclusion with the `-f`, `-x`, `-y`, `-Y`, and `-z` options, described below. See the subsection on combining these options, following OPTIONS, below.

By default, all swap partitions on a UFS-based source BE are shared with a UFS-based target BE. For UFS-based target BEs, you can use the `-m` option (see below) to specify an additional or new set of swap partitions on the source BE for sharing with the target. When a UFS-based source BE is copied to a ZFS target BE, `lucreate` creates in the new BE a swap area and a dump device on separate ZFS volumes. When both the source and target BEs are ZFS-based and are in the same pool, both BEs use the same swap volume. If source and target are in different pools, a new swap volume is created in the pool of the target BE.

The `lucreate` command allows you to assign a description to a BE. A description is an optional attribute of a BE that can be of any format or length. It might be, for example, a text string or binary data. After you create a BE, you can change a BE description with the `ludesc(1M)` utility.

The `lucreate` command requires root privileges or that you assume the Primary Administrator role.

Options The `lucreate` command has the options listed below. Note that a BE name must not exceed 30 characters in length and must consist only of alphanumeric characters and other ASCII characters that are not special to the Unix shell. See the "Quoting" section of `sh(1)`. The BE name can contain only single-byte, 8–bit characters; it cannot contain whitespace characters.

Omission of `-m`, `-M`, and `-p` options (described below) in an `lucreate` command line invokes the FMLI-based interface, which allows you to select disk or device slices for a UFS-based BE.

`-A` *BE_description*
Assigns the *BE_description* to a BE. *BE_description* can be a text string or other characters that can be entered on a Unix command line. See `ludesc(1M)` for additional information on BE descriptions.

`-c` *BE_name*
Assigns the name *BE_name* to the current BE. This option is not required and can be used only when the first BE is created. For the first time you run `lucreate`, for a UFS-based BE, if you omit `-c`, `lucreate` supplies a default name according to the following rules:

1. If the physical boot device can be determined, the base name of that device is used to name the new boot environment. For example, if the physical boot device is `/dev/dsk/c0t0d0s0`, `lucreate` names the new boot environment `c0t0d0s0`.
2. If the physical boot device cannot be determined, the operating system name (from `uname -s`) and operating system release level (from `uname -r`) are combined to produce the name of the new boot environment. For example, if `uname -s` returns `SunOS` and `uname -r` returns `5.9`, then `lucreate` assigns the name `SunOS5.9` to the new boot environment.
3. If `lucreate` can determine neither boot device nor operating system name, it assigns the name `current` to the new boot environment.

For a ZFS-based BE, the default BE name is the base name of the root file system.

If you use the `-c` option after the first boot environment is created, the option is ignored if the name specified is the same as the current boot environment name. If the name is different, `lucreate` displays an error message and exits.

`-C` (*boot_device* | –)
Provided for occasions when `lucreate` cannot figure out which physical storage device is your boot device. This might occur, for example, when you have a mirrored root device on the source BE on an x86 machine. The `-C` specifies the physical boot device from which the source BE is booted. Without this option, `lucreate` attempts to determine the physical device from which a BE boots. If the device on which the root file system is located is not a physical disk (for example, if root is on a Solaris Volume Manager volume) and `lucreate` is able to make a reasonable guess as to the physical device, you receive the query:

```
Is the physical device devname the boot device for
the logical device devname?
```

If you respond `y`, the command proceeds.

If you specify `-C` *boot_device*, `lucreate` skips the search for a physical device and uses the device you specify. The – (hyphen) with the `-C` option tells `lucreate` to proceed with whatever it determines is the boot device. If the command cannot find the device, you are prompted to enter it.

If you omit `-C` or specify `-C` *boot_device* and `lucreate` cannot find a boot device, you receive an error message.

Use of the `-C` – form is a safe choice, because `lucreate` either finds the correct boot device or gives you the opportunity to specify that device in response to a subsequent query.

`-f` *exclude_list_file*
Use the contents of *exclude_list_file* to exclude specific files (including directories) from the newly created BE. *exclude_list_file* contains a list of files and directories, one per line. If a line item is a file, only that file is excluded; if a directory, that directory and all files beneath that directory, including subdirectories, are excluded.

This option is not supported when the source BE is on a ZFS file system.

`-I`
Ignore integrity check. Prior to creating a new BE, `lucreate` performs an integrity check, to prevent you from excluding important system files from the BE. Use this option to override this integrity check. The trade-off in use of this option is faster BE creation (with `-I`) versus the risk of a BE that does not function as you expect.

`-l` *error_log*
Error messages and other status messages are sent to *error_log*, in addition to where they are sent in your current environment.

`-m` *mount_point*:*device*[,*volume*]:*fs_option*[:*zonename*]
[`-m` *mount_point*:*device*:*fs_option*[:*zonename*]] ...
Specifies the `vfstab(4)` information for a new UFS-based BE. The file systems specified as arguments to `-m` can be on the same disk or can be spread across multiple disks.

The `-m` option is not supported for BEs based on ZFS file systems. This option also does not support EFI-labeled disks.

mount_point can be any valid mount point or – (hyphen), indicating a swap partition. The *device* field can be one of the following:

- The name of a disk slice, of the form `/dev/dsk/c`*num*`t`*num*`d`*num*`s`*num*.
- The name of a Solaris Volume Manager volume, of the form `/dev/md/dsk/d`*num*.
- The name of a Solaris Volume Manager disk set, of the form `/dev/md/`*setname*`/dsk/d`*num*.

- The name of a Veritas volume, of the form `/dev/vx/dsk/`*dgname*`/`*volname*.
- The keyword `merged`, indicating that the file system at the specified mount point is to be merged with its parent.
- The keyword `shared`, indicating that all of the swap partitions in the source BE are to be shared with the new BE.

The `-m` option enables you to attach a physical disk device to a Solaris Volume Manager single–slice concatenation or attach a Solaris Volume Manager volume to a mirror. Both operations are accomplished with the `attach` keyword, described below. With this option, you have the choice of specifying a concatentation or mirror or allowing `lucreate` to select one for you. To specify a concatenation or mirror, append a comma and the name of the Solaris Volume Manager logical device to the device name to which the logical device is being attached. If you omit this specification, `lucreate` selects a concatenation or mirror from a list of free devices. See EXAMPLES.

The *fs_option* field can be one or more of the keywords listed below. The first two keywords specify types of file systems. The remaining keywords specify actions to be taken on a file system. When you specify multiple keywords, separate these with a comma.

`ufs` Create the file system as a UFS volume.

`vxfs` Create the file system as a Veritas device.

`preserve` Preserve the file system contents of the specified physical storage device. Use of this keyword presumes that the device's file system and its contents are appropriate for the specified mount point. For a given mount point, you can use preserve with only one device. This keyword enables you to bypass the default steps of creating a new file system on the specified storage device, then copying the file system contents from the source BE to the specified device. When you use `preserve`, `lucreate` checks that the storage device's contents is suitable for a specified file system. This check is limited and cannot guarantee suitability.

`mirror` Create a mirror on the specified storage device. The specified storage device must be a correctly named (for example, `/dev/md/dsk/d10`) logical device that can serve as a mirror. In subsequent `-m` options, you must specify `attach` (see below) to attach at least one physical device to the new mirror.

`attach` Attach a physical storage device, contained by a volume, to the mirror or single-slice concatenation associated with a specified mount point. When using `attach`, if you want to attach a disk to a specific mirror or concatenation, you append a comma and the name of that logical device to the device name. If you omit the comma and the concatentation name, `lucreate` selects a free mirror or single-slice concatenation as the container volume for the storage device. See EXAMPLES.

`lucreate` allows you to create only concatenations that contain a single physical drive and allows you to attach up to four such concatenations to a mirror.

`detach` Detach a physical storage device from the mirror or concatenation associated with a specified mount point.

The optional *zonename* field specifies the name of an installed non-global zone. It is used to specify a separate file system that belongs to the particular zone, named *zonename*, that exists in the new BE being created.

At minimum, you must specify one disk or device slice, for root. You can do this with `-m`, `-M` (described below), or in the FMLI-based interface. You must specify an `-m` argument for each file system you want to create on a new BE. For example, if you have three file systems on a source BE (say, `/`, `/usr`, and `/var`) and want these three entities as separate file systems on a new BE, you must specify three `-m` arguments. If you were to specify only one, in our example, `/`, `/usr`, and `/var` would be merged on the new BE into a single file system, under `/`.

When using the `-m` option to specify swap partition(s), you can designate device(s) currently used for swap on any BE and any unused devices. Regarding swap assignments, you have the following choices:

- Omit any specification of swap devices, in which case all swap devices associated with the source BE will be used by the new BE.
- Specify one or more swap devices, in which case the new BE will use only the specified swap devices and not automatically share the swap devices associated with the source BE.
- Specify one or more swap devices and use the syntax `-m –:shared:swap`, in which case the new BE will use the specified swap devices and will share swap devices with the source BE.

See EXAMPLES, below.

`-M` *slice_list*

List of `-m` options, collected in the file *slice_list*. Specify these arguments in the format specified for `-m`. Comment lines, beginning with a hash mark (#), are ignored. The `-M` option is useful where you have a long list of file systems for a BE. Note that you can combine `-m` and `-M` options. For example, you can store swap partitions in *slice_list* and specify `/` and `/usr` slices with `-m`.

The `-M` option is not supported for BEs based on ZFS file systems.

The `-m` and `-M` options support the listing of multiple slices for a given mount point. In processing these slices, `lucreate` skips any unavailable slices and selects the first available slice. See EXAMPLES.

`-n` *BE_name*
The name of the BE to be created. *BE_name* must be unique on a given system.

`-o` *outfile*
All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-p` *zfs_root_pool*
Specifies the ZFS pool in which a new BE will reside.

This option can be omitted if the source and target BEs are within the same pool.

The `-p` option does not support the splitting and merging of file systems in a target BE that is supported by the `-m` option.

`-s` (– | *BE_name*)
Source for the creation of the new BE. This option enables you to use a BE other than the current BE as the source for creation of a new BE.

If you specify a hyphen (`-`) as an argument to `-s`, `lucreate` creates the new BE, but does not populate it. This variation of the `-s` option is intended for the subsequent installation of a flash archive on the unpopulated BE using `luupgrade(1M)`. See `flar(1M)`.

`-x` *exclude*
Exclude the file or directory *exclude* from the newly created BE. If *exclude* is a directory, `lucreate` excludes that directory and all files beneath that directory, including subdirectories.

This option is not supported when the source BE is on a ZFS file system.

`-X`
Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

`-y` *include*
Include the file or directory *include* in the newly created BE. If *include* is a directory, `lucreate` includes that directory and all files beneath that directory, including subdirectories.

This option is not supported when the source BE is on a ZFS file system.

`-Y` *include_list_file*
Use the contents of *include_list_file* to include specific files (including directories) from the newly created BE. *include_list_file* contains a list of files and directories, one per line. If a line item is a file, only that file is included; if a directory, that directory and all files beneath that directory, including subdirectories, are included.

This option is not supported when the source BE is on a ZFS file system.

`-z` *filter_list_file*
filter_list_file contains a list of items, files and directories, one per line. Each item is preceded by either a +, indicating the item is to be included in the new BE, or -, indicating the item is to be excluded from the new BE.

This option is not supported when the source BE is on a ZFS file system.

Combining File Inclusion and Exclusion Options

When a source BE is on a UFS file system, the `lucreate` command allows you to include or exclude specific files and directories when creating a new BE. You can include files and directories with:

- the `-y` *include* option
- the `-Y` *include_list_file* option
- items with a leading + in the file used with the `-z` *filter_list* option

You can exclude files and directories with:

- the `-x` *exclude* option
- the `-f` *exclude_list_file* option
- items with a leading – in the file used with the `-z` *filter_list* option

If the parent directory of an excluded item is included with include options (for example, `-y` *include*), then only the specific file or directory specified by *exclude* is excluded. Conversely, if the parent directory of an included file is specified for exclusion, then only the file *include* is included. For example, if you specify:

```
-x /a -y /a/b
```

all of `/a` except for `/a/b` is excluded. If you specify:

```
-y /a -x /a/b
```

all of `/a` except for `/a/b` is included.

Examples

The `lucreate` command produces copious output. In the following examples, this output is not reproduced, except where it is needed for clarity.

EXAMPLE 1 Creating a New Boot Environment for the First Time

The following command sequence creates a new boot environment on a machine on which a BE has never been created. All non-shareable (critical) file systems are mounted under `/`.

```
# lucreate -c first_disk -m /:/dev/dsk/c0t4d0s0:ufs -n second_disk
many lines of output
lucreate: Creation of Boot Environment <second_disk> successful.
```

The following command, like the preceding, creates a new boot environment on a machine on which a BE has never been created. However, the following command differs in two respects: the `-c` option is omitted and the `/usr` file system is mounted on its own disk slice, separate from `/`.

EXAMPLE 1 Creating a New Boot Environment for the First Time *(Continued)*

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
-n second_disk
lucreate: Please wait while your system configuration is determined.
many lines of output
lucreate: Creation of Boot Environment c0t4d0s0 successful.
```

In the absence of the `-c` option, `lucreate` assigns the name `c0t4d0s0`, the base name of the root device, to the new boot environment.

The same command is entered, with the addition of `-c`:

```
# lucreate -c first_disk -m /:/dev/dsk/c0t4d0s0:ufs \
-m /usr:/dev/dsk/c0t4d0s1:ufs -n second_disk
many lines of output
lucreate: Creation of Boot Environment <second_disk> successful.
```

Following creation of a BE, you use `luupgrade(1M)` to upgrade the OS on the new BE and `luactivate(1M)` to make that BE the BE you will boot from upon the next reboot of your machine. Note that the swap partition and all shareable file systems for `first_disk` will be available to (shared with) `second_disk`.

```
# luupgrade -u -n second_disk \
-s /net/installmachine/export/solarisX/OS_image
many lines of output
luupgrade: Upgrade of Boot Environment <second_disk> successful.

# luactivate second_disk
```

See `luupgrade(1M)` and `luactivate(1M)` for descriptions of those commands.

EXAMPLE 2 Creating a BE Using a Source Other than the Current BE

The following command uses the `-s` option to specify a source BE other than the current BE.

```
# lucreate -s third_disk -m /:/dev/dsk/c0t4d0s0:ufs \
-m /usr:/dev/dsk/c0t4d0s1:ufs -n second_disk
many lines of output
lucreate: Creation of Boot Environment <second_disk> successful.
```

EXAMPLE 3 Migrating a BE from a UFS Root File System to a ZFS Root File System

The following command creates a BE of a ZFS root file system from a UFS root file system. The current BE, `c1t0d0s0`, containing a UFS root file system, is identified by the `-c` option. The new BE, `zfsBE`, is identified by the `-n` option. A ZFS storage pool must exist before the `lucreate` operation and must be created with slices rather than whole disks to be upgradeable and bootable.

```
# zpool create rpool mirror c1t0d0s0 c2t0d0s0
# lucreate -c c1t0d0s0 -n zfsBE -p rpool
```

EXAMPLE 3 Migrating a BE from a UFS Root File System to a ZFS Root File System *(Continued)*

Note that if the current BE also resides on the ZFS pool `rpool`, the `-p` option could be omitted. For example:

```
# lucreate -n zfsBE
```

EXAMPLE 4 Creating a BE from a Flash Archive

Performing this task involves use of `lucreate` with the `-s` – option and `luupgrade`.

```
# lucreate -s - -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
-n second_disk
brief messages
lucreate: Creation of Boot Environment <second_disk> successful.
```

With the `-s` option, the `lucreate` command completes it work within seconds. At this point, you can use `luupgrade` to install the flash archive:

```
# luupgrade -f -n second_disk \
-s /net/installmachine/export/solarisX/OS_image \
-J "archive_location http://example.com/myflash.flar"
```

See `luupgrade(1M)` for a description of that command.

EXAMPLE 5 Sharing and Adding Swap Partitions

In the simplest case, if you do not specify any swap partitions in an `lucreate` command, all swap partitions in the source BE are shared with the new BE. For example, assume that the current BE uses `/dev/dsk/c0t4d0s7` as its swap partition. You enter the command:

```
# lucreate -n second_disk -m /:/dev/dsk/c0t4d0s0:ufs
many lines of output
lucreate: Creation of Boot Environment <second_disk> successful.
```

Upon conclusion of the preceding command, the partition `/dev/dsk/c0t4d0s7` will be used by the BE `second_disk` when that BE is activated and booted.

If you want a new BE to use a different swap partition from that used by the source BE, enter one or more `-m` options to specify a new partition or new partitions. Assume, once again, that the current BE uses `/dev/dsk/c0t4d0s7` as its swap partition. You enter the command:

```
# lucreate -m /:/dev/dsk/c0t0d0s0:ufs -m -:/dev/dsk/c0t4d0s1:swap \
 -m -:/dev/dsk/c0t4d0s2:swap -n second_disk
many lines of output
lucreate: Creation of Boot Environment <second_disk> successful.
```

Upon activation and boot, the new BE `second_disk` will use `/dev/dsk/c0t4d0s1` and `/dev/dsk/c0t4d0s2` and will not use `/dev/dsk/c0t4d0s7`, the swap partition used by the source BE.

EXAMPLE 5 Sharing and Adding Swap Partitions *(Continued)*

Assume you want the new BE `second_disk` to share the source BE's swap partition *and* have an additional swap partition. You enter:

```
# lucreate -m /:/dev/dsk/c0t0d0s0:ufs -m -:/dev/dsk/c0t4d0s1:swap \
 -m -:shared:swap -n second_disk
many lines of output
lucreate: Creation of Boot Environment <second_disk> successful.
```

Upon activation and boot, the new BE `second_disk` will use for swapping `/dev/dsk/c0t4d0s7`, shared with the source BE, and, in addition, `/dev/dsk/c0t4d0s1`.

EXAMPLE 6 Using Swap Partitions on Multiple Disks

The command below creates a BE on a second disk and specifies swap partitions on both the first and second disks.

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m -:/dev/dsk/c0t4d0s1:swap \
 -m -:/dev/dsk/c0t0d0s1:swap -n second_disk
many lines of output
lucreate: Creation of Boot Environment <second_disk> successful.
```

Following completion of the preceding command, the BE `second_disk` will use both `/dev/dsk/c0t0d0s1` and `/dev/dsk/c0t4d0s1` as swap partitions. These swap assignments take effect only after booting from `second_disk`. If you have a long list of swap partitions, it is useful to use the `-M` option, as shown below.

EXAMPLE 7 Using a Combination of `-m` and `-M` Options

In this example, a list of swap partitions is collected in the file `/etc/lu/swapslices`. The location and name of this file is user-defined. The contents of `/etc/lu/swapslices`:

```
-:/dev/dsk/c0t3d0s2:swap
-:/dev/dsk/c0t3d0s2:swap
-:/dev/dsk/c0t4d0s2:swap
-:/dev/dsk/c0t5d0s2:swap
-:/dev/dsk/c1t3d0s2:swap
-:/dev/dsk/c1t4d0s2:swap
-:/dev/dsk/c1t5d0s2:swap
```

This file is specified in the following command:

```
# lucreate -m /:/dev/dsk/c02t4d0s0:ufs -m /usr:/dev/dsk/c02t4d0s1:ufs \
-M /etc/lu/swapslices -n second_disk
many lines of output
lucreate: Creation of Boot Environment <second_disk> successful.
```

The BE `second_disk` will swap onto the partitions specified in `/etc/lu/swapslices`.

EXAMPLE 8 Copying Versus Sharing

The following command copies the user file system `/home` (in addition to the non–shareable file systems `/` and `/usr`) from the current BE to the new BE:

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
-m /home:/dev/dsk/c0t4d0s4:ufs -n second_disk
```

The following command differs from the preceding in that the `-m` option specifying a destination for `/home` is omitted. The result of this is that `/home` will be shared between the current BE and the BE `second_disk`.

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
-n second_disk
```

EXAMPLE 9 Using Solaris Volume Manager Volumes

The command shown below does the following:

1. Creates the mirror `d10` and establishes this mirror as the receptacle for the root file system.
2. Attaches `c0t0d0s0` and `c0t1d0s0` to single-slice concatenations `d1` and `d2`, respectively. Note that the specification of these volumes is optional.
3. Attaches the concatenations associated with `c0t0d0s0` and `c0t1d0s0` to mirror `d10`.
4. Copies the current BE's root file system to mirror `d10`, overwriting any `d10` contents.

```
# lucreate -m /:/dev/md/dsk/d10:ufs,mirror \
-m /:/dev/md/dsk/d1:attach \
-m /:/dev/dsk/c0t1d0s0,d2:attach -n newBE
```

The following command differs from the preceding only in that concatenations for the physical storage devices are not specified. In this example, `lucreate` chooses concatenation names from a list of free names and attaches these volumes to the mirror specified in the first `-m` option.

```
# lucreate -m /:/dev/md/dsk/d10:ufs,mirror \
-m /:/dev/dsk/c0t0d0s0:attach \
-m /:/dev/dsk/c0t1d0s0:attach -n newBE
```

The following command differs from the preceding commands in that one of the physical disks is detached from a mirror before being attached to the mirror you create. Also, the contents of one of the physical disks is preserved. The command does the following:

1. Creates the mirror `d10` and establishes this mirror as the receptacle for the root file system.
2. Detaches `c0t0d0s0` from the mirror to which it is currently attached.
3. Attaches `c0t0d0s0` and `c0t1d0s0` to concatenations `d1` and `d2`, respectively. Note that the specification of the these concatenations is optional.
4. Preserves the contents of `c0t0d0s0`, which presumes that `c0t0d0s0` contains a valid copy of the current BE's root file system.

EXAMPLE 9 Using Solaris Volume Manager Volumes *(Continued)*

5. Attaches the concatenations associated with `c0t0d0s0` and `c0t1d0s0` (`d1` and `d2`) to mirror `d10`.

```
# lucreate -m /:/dev/md/dsk/d10:ufs,mirror \
-m /:/dev/dsk/c0t0d0s0,d1:detach,attach,preserve \
-m /:/dev/dsk/c0t1d0s0,d2:attach -n newBE
```

The following command is a follow-on to the first command in this set of examples. This command detaches a concatenation (containing `c0t0d0s0`) from one mirror (`d10`, in the first command) and attaches it to another (`d20`), preserving its contents.

```
# lucreate -m /:/dev/md/dsk/d20:ufs,mirror \
-m /:/dev/dsk/c0t0d0s0:detach,attach,preserve -n nextBE
```

The following command creates two mirrors, placing the `/` file system of the new BE on one mirror and the `/opt` file system on the other.

```
# lucreate -m /:/dev/md/dsk/d10:ufs,mirror \
-m /:/dev/dsk/c0t0d0s0,d1:attach \
-m /:/dev/dsk/c1t0d0s0,d2:attach \
-m /opt:/dev/md/dsk/d11:ufs,mirror \
-m /opt:/dev/dsk/c2t0d0s1,d3:attach \
-m /opt:/dev/dsk/c3t1d0s1,d4:attach -n anotherBE
```

EXAMPLE 10 Invoking FMLI-based Interface

This example is included for historical purposes as the `lu` interface is now obsolete.

The command below, by omitting `-m` or `-M` options, invokes `lu`, the FMLI-based interface for Live Upgrade operations.

```
# lucreate -n second_disk
```

The preceding command uses the current BE as the source for the target BE `second_disk`. In the FMLI interface, you can specify the target disk slices for `second_disk`. The following command is a variation on the preceding:

```
# lucreate -n second_disk -s third_disk
```

In the preceding command, a source for the target BE is specified. As before, the FMLI interface comes up, enabling you to specify target disk slices for the new BE.

EXAMPLE 11 Merging File Systems

The command below merges the `/usr/opt` file system into the `/usr` file system. First, here are the disk slices in the BE `first_disk`, expressed in the format used for arguments to the `-m` option:

EXAMPLE 11 Merging File Systems *(Continued)*

```
/:/dev/dsk/c0t4d0s0:ufs
/usr:/dev/dsk/c0t4d0s1:ufs
/usr/opt:/dev/dsk/c0t4d0s3:ufs
```

The following command creates a BE `second_disk` and performs the merge operation, merging `/usr/opt` with its parent, `/usr`.

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
-m /usr/opt:merged:ufs -n second_disk
```

EXAMPLE 12 Splitting a File System

Assume a source BE with `/`, `/usr`, and `/var` all mounted on the same disk slice. The following command creates a BE `second_disk` that has `/`, `/usr`, and `/var` all mounted on different disk slices.

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /usr:/dev/dsk/c0t4d0s1:ufs \
/var:/dev/dsk/c0t4d0s3:ufs -n second_disk
```

This separation of a file system's (such as root's) components onto different disk slices is referred to as splitting a file system.

EXAMPLE 13 Specifying Alternative Slices

The following command uses multiple `-m` options as alternative disk slices for the new BE `second_disk`.

```
# lucreate -m /:/dev/dsk/c0t4d0s0:ufs -m /:/dev/dsk/c0t4d0s1:ufs \
-m /:/dev/dsk/c0t4d0s5:ufs -n second_disk
many lines of output
lucreate: Creation of Boot Environment <second_disk> successful.
```

The preceding command specifies three possible disk slices, `s0`, `s1`, and `s5` for the `/` file system. `lucreate` selects the first one of these slices that is not being used by another BE. Note that the `-s` option is omitted, meaning that the current BE is the source BE for the creation of the new BE.

EXAMPLE 14 Specifying Separate File Systems for Non-Global Zones

The following command specifies a separate file system belonging to the zone, `zone1`, within the new BE, `second_disk`.

```
# lucreate -n second_disk -m /:/dev/dsk/c0d0s3:ufs \
-m /export/home:/dev/dsk/c0d0s5:ufs:zone1
```

The zone named `zone1`, inside the new BE, has a separate disk slice allocated for its `/export/home` file system.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.<num>`	Live Upgrade DTD (see `-X` option)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlu

See Also `luactivate(1M)`, `lucancel(1M)`, `lucompare(1M)`, `lucurr(1M)`, `ludelete(1M)`, `ludesc(1M)`, `lufslist(1M)`, `lumake(1M)`, `lumount(1M)`, `lurename(1M)`, `lustatus(1M)`, `luupgrade(1M)`, `zfs(1M)`, `zpool(1M)`, `zoneadm(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`, `zones(5)`

Notes As is true for any Solaris operating system upgrade (and not a feature of Live Upgrade), when splitting a directory into multiple mount points, hard links are not maintained across file systems. For example, if `/usr/test1/buglist` is hard linked to `/usr/test2/buglist`, and `/usr/test1` and `/usr/test2` are split into separate file systems, the link between the files will no longer exist. If `lucreate` encounters a hard link across file systems, the command issues a warning message and creates a symbolic link to replace the lost hard link.

`lucreate` cannot prevent you from making invalid configurations with respect to non-shareable file systems. For example, you could enter an `lucreate` command that would create separate file systems for `/` and `/kernel`—an invalid division of `/`. The resulting BE would be unbootable. When creating file systems for a boot environment, the rules are identical to the rules for creating file systems for the Solaris operating environment.

Mindful of the principle described in the preceding paragraph, consider the following:

- In a source BE, you must have valid `vfstab` entries for every file system you want to copy to or share with a new BE.
- You cannot create a new BE on a disk with overlapping partitions (that is, partitions that share the same physical disk space). The `lucreate` command that specifies such a disk might complete, but the resulting BE would be unbootable.

Note – As stated in the description of the `-m` option, if you use Solaris Volume Manager volumes for boot environments, use `lucreate` rather than Solaris Volume Manager commands to manipulate these volumes. The Solaris Volume Manager software has no knowledge of boot environments; the `lucreate` command contains checks that prevent you from inadvertently destroying a boot environment by, for example, overwriting or deleting a Solaris Volume Manager volume.

For versions of the Solaris operating system prior to Solaris 10, Live Upgrade supports the release it is distributed on and up to three marketing releases back. For example, if you obtained Live Upgrade with Solaris 9 (including a Solaris 9 upgrade), that version of Live Upgrade supports Solaris versions 2.6, Solaris 7, and Solaris 8, in addition to Solaris 9. No version of Live Upgrade supports a Solaris version prior to Solaris 2.6.

Starting with version 10 of the Solaris operating system, Live Upgrade supports the release it is distributed on and up to two marketing releases back. For example, if you obtained Live Upgrade with Solaris 10 (including a Solaris 10 upgrade), that version of Live Upgrade supports Solaris 8 and Solaris 9, in addition to Solaris 10. For instructions on adding Live Upgrade packages for the release you want to install, see *Solaris 10 5/08 Installation Guide: Solaris Live Upgrade and Upgrade Planning*.

Correct operation of Solaris Live Upgrade requires that a limited set of patch revisions be installed for a given OS version. Before installing or running Live Upgrade, you are required to install the limited set of patch revisions. Make sure you have the most recently updated patch list by consulting `http://sunsolve.sun.com`. Search for the infodoc 72099 on the SunSolve web site.

Name lucurr – display the name of the active boot environment

Synopsis

```
/usr/sbin/lucurr [-l error_log] [-m mount_point]
     [-o outfile] [-X]
```

Description The `lucurr` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `lucurr` command displays the name of the currently running boot environment (BE). If no BEs are configured on the system, `lucurr` displays the message "No Boot Environments are defined". Note that `lucurr` reports only the name of the current BE, not the BE that will be active upon the next reboot. Use `lustatus(1M)` or `luactivate(1M)` for this information.

The `lucurr` command requires root privileges.

Options The `lucurr` command has the following options:

`-l` *error_log* — Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.

`-m` *mount_point* — Returns the name of the BE that owns *mount_point*, where *mount_point* is the mount point of a BE's root file system. This can be a mount point of the current BE or the mount point of a BE other than the current BE. If the latter, the file system of the BE must have been mounted with `lumount(1M)` or `mount(1M)` before entering this option.

`-o` *outfile* — All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-X` — Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>	Live Upgrade DTD (see `-X` option)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also luactivate(1M), lucancel(1M), lucompare(1M), lucreate(1M), ludelete(1M), ludesc(1M), lufslist(1M), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), lutab(4), attributes(5), live_upgrade(5)

Name ludelete – delete a boot environment

Synopsis

```
/usr/sbin/ludelete [-l error_log] [-o outfile] BE_name [-X]
```

Description The `ludelete` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `ludelete` command deletes all records associated with a boot environment (BE) on all defined complete BEs. A complete BE is one that is not participating in an `lucreate(1M)`, `luupgrade(1M)`, or `lucompare(1M)` operation. Use `lustatus(1M)` to determine a BE's status. You can delete neither the current BE, nor the BE that will become current upon the next reboot.

`ludelete` does not alter any files on the BE being deleted.

The `ludelete` command requires root privileges.

Options The `ludelete` command has the following options:

`-l` *error_log* Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.

`-o` *outfile* All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-X` Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Operands *BE_name* Name of the BE to be deleted.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>	Live Upgrade DTD (see `-X` option)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also lu(1M), luactivate(1M), lucancel(1M), lucompare(1M), lucreate(1M), lucurr(1M), ludesc(1M), lufslist(1M), lumake(1M), lumount(1M), lurename(1M), lustatus(1M), luupgrade(1M), lutab(4), attributes(5), live_upgrade(5)

Name ludesc – display or set boot environment description

Synopsis

```
/usr/sbin/ludesc {-A BE_description} | {-f {filename | -}}
     [-l error_log] [-o outfile] [-X]

/usr/sbin/ludesc -n BE_name [-f filename | -]
     [-l error_log] [-o outfile] [-X]

/usr/sbin/ludesc -n BE_name [-l error_log] [-o outfile]
     [-X] BE_description
```

Description The `ludesc` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `ludesc` command allows you to manipulate boot environment (BE) descriptions. A BE description is an optional attribute of a BE. It can be text or binary data. For example, it might be a string such as "S9 beta test BE" or it might be a file that contains 8–bit multi-byte characters. The `ludesc` command in general and the options to manipulate binary-format descriptions in particular are suitable for use in programs.

You create a BE description using `ludesc` or `lucreate(1M)`. Only `ludesc` allows you to change a BE description or add a description following BE creation.

While a BE description is associated with a BE name, it is not interchangeable with that name. No Live Upgrade command allows you to specify a BE description instead of a BE name when performing an operation on a BE.

A shell might restrict what you enter for a BE description (in both `ludesc` and `lucreate(1M)`). In entering a description, use the following guidelines:

- Always enclose a description in single quotes ('), unless the description includes a single quote.
- If your description includes a single quote, enclose the description in double quotes ("). You then must use an escape sequence (usually a backslash [\]) to enter a character that is special to the shell. See `sh(1)` for a list of special characters and a description of the escape sequence mechanism.

Descriptions that include many special characters might be more conveniently inserted in a file (`-f` option) than entered on a command line (`-A` option).

When `ludesc` outputs a BE description, it does so exactly as the description was entered. Because of this feature, a description that is a text string does not have a concluding newline, which means the system prompt immediately follows the last character of the description.

The `ludesc` command requires root privileges.

Options The `ludesc` command has the following options:

`-A` *BE_description*	Displays the BE name associated with *BE_description*.
`-f` {*filename* | –}	Specify the BE description contained in *filename* or read from `stdin`. When used without `-n`, displays the BE name associated with the specified BE description. Used with `-n`, changes the description for the specified BE to the description specified with `-f`.
`-l` *error_log*	Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.
`-n` *BE_name*	With no other arguments, displays the BE description for the specified BE. With the `-f` option or the *BE_description* operand, changes the description for the specified BE to that specified with `-f` or *BE_description*.
`-o` *outfile*	All command output is sent to *outfile*, in addition to where it is sent in your current environment.
`-X`	Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Operands *BE_description* Used only with the `-n` option. *BE_description* replaces the current BE description for the specified BE.

Examples The following are examples of the use of `ludesc`.

EXAMPLE 1 Basic Use

The first command, below, assigns a description to a BE. The second command returns the name of the BE associated with the specified description. The last command returns the description associated with a specified BE.

```
# ludesc -n first_disk 'Test disk'
Setting description for boot environment <first_disk>.
Propagating the change of BE description to all BEs.

# ludesc -A 'Test disk'
first_disk
#

# ludesc -n first_disk
Test disk#
```

As seen above and noted in the DESCRIPTION, `ludesc` does not append a newline to the display of BE description that is a text string.

EXAMPLE 2 Using Binary Files

The following commands are analogs of the preceding examples, substituting a binary file–here, a file containing a description in Russian, using the Cyrillic alphabet—for a text string. In the third command, note the use of a file to capture output. Sending output of a binary file to the console can produce erratic results.

```
# ludesc -n first_disk -f arrayBE.ru
Setting description for boot environment <first_disk>.
Propagating the change of BE description to all BEs.

# ludesc -f arrayBE.ru
first_disk

# ludesc -n first_disk > /tmp/arrayBE.out
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.`*<num>*	Live Upgrade DTD (see `-X` option)

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also

`luactivate(1M)`, `lucancel(1M)`, `lucompare(1M)`, `lucreate(1M)`, `ludelete(1M)`, `lufslist(1M)`, `lumake(1M)`, `lumount(1M)`, `lurename(1M)`, `lustatus(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`

Name lufslist – list configuration of a boot environment

Synopsis

```
/usr/sbin/lufslist [-l error_log] [-o outfile] BE_name [-X]
```

Description The `lufslist` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `lufslist` command lists the configuration of a boot environment (BE). The output contains the disk slice (file system), file system type, and file system size for each BE mount point. The output also notes any separate file systems that belong to a non-global zone inside the BE being displayed.

The following is an example of `lufslist` output, which shows a separate filesystem for the zone, `zone1`, within the BE being displayed.

```
# lufslist BE_name
Filesystem             fstype       size(Mb) Mounted on
--------------------------------------------------------------------
/dev/dsk/c0t0d0s1      swap           512.11 -
/dev/dsk/c0t4d0s3      ufs           3738.29 /
/dev/dsk/c0t4d0s4      ufs            510.24 /opt

              zone zone1 within boot environment BE_name
/dev/dsk/c0t4d0s7      ufs            7000.48 /export
```

File system type can be `ufs`, `swap`, or `vxfs`, for a Veritas file system. Under the `Filesystem` heading can be a disk slice or a logical device, such as a disk metadevice used by volume management software.

The `lufslist` command requires root privileges or the Primary Administrator role.

Options The `lufslist` command has the following options:

`-l` *error_log* Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.

`-o` *outfile* All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-X` Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Operands *BE_name* Name of the BE for which file systems are to be reported. You cannot specify a BE that is involved in another Live Upgrade operation.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>	Live Upgrade DTD (see `-X` option)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also `luactivate(1M)`, `lucancel(1M)`, `lucompare(1M)`, `lucreate(1M)`, `lucurr(1M)`, `ludelete(1M)`, `ludesc(1M)`, `lumake(1M)`, `lumount(1M)`, `lurename(1M)`, `lustatus(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`, `zones(5)`

Name lumake – populate a boot environment

Synopsis

```
/usr/sbin/lumake [-l error_log] [-o outfile] [-s source_BE] -n BE_name
     [-X]

/usr/sbin/lumake [-l error_log] -t time [-o outfile]
     [-s source_BE] -n BE_name [-m email_address] [-X]
```

Description The `lumake` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `lumake` command populates (that is, copies files to) the file systems of a specified boot environment (BE) by copying files from the corresponding file systems of the active or a source (`-s`) BE. Any existing data on the target BE are destroyed. All file systems on the target BE are re-created.

The target BE must already exist. Use `lucreate(1M)` to create a new BE.

The `lumake` command requires root privileges.

Options The `lumake` command has the following options:

`-n` *BE_name* — Name of the BE to be populated.

`-s` *source_BE* — The optional name of a source BE. If you omit this option, `lumake` uses the current BE as the source. A BE must have the status "complete" before you can copy from it. Use `lustatus(1M)` to determine a BE's status.

`-l` *error_log* — Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.

`-o` *outfile* — All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-t` *time* — Setup a batch job to populate the specified BE at a specified time. The time is given in the format specified by the `at(1)` man page. At any time, you can have only one Live Upgrade operation scheduled. You can use `lucancel(1M)` to cancel a scheduled `lumake` operation.

`-m` *email_address* — Allows you to email `lumake` output to a specified address upon command completion. There is no checking of *email_address*. Use this option in conjunction with `-t`, to obtain notification of batch command completion. If you use `-m` with options other than `-t`, no email is sent.

`-X` — Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.<num>`	Live Upgrade DTD (see `-X` option)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also `luactivate(1M)`, `lucancel(1M)`, `lucompare(1M)`, `lucreate(1M)`, `lucurr(1M)`, `ludelete(1M)`, `ludesc(1M)`, `lufslist(1M)`, `lumount(1M)`, `lurename(1M)`, `lustatus(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`

Name lumount, luumount – mount or unmount all file systems in a boot environment

Synopsis

```
/usr/sbin/lumount [-l error_log] [-o outfile] BE_name
     [mount_point] [-X]

/usr/sbin/lumount

/usr/sbin/luumount [-f]
     { [-n] BE_name | [-m] mount_point | block_device}
     [-l error_log] [-o outfile] [-X]
```

Description The `lumount` and `luumount` commands are part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `lumount` and `luumount` commands enable you to mount or unmount all of the file systems in a boot environment (BE). This allows you to inspect or modify the files in a BE while that BE is not active. By default, `lumount` mounts the file systems on a mount point of the form `/.alt.`*BE_name*, where *BE_name* is the name of the BE whose file systems are being mounted. See NOTES.

`lumount` and `luumount` also mount or unmount all installed non-global zones within the BE. For each running, mounted, or ready non-global zone in the current BE, `lumount` mounts all file systems in the mounted BE that belong to the non-global zone, at the specified mount point in the non-global zone. This provides the non-global zone administrator access to the corresponding file systems that exist in the mounted BE.

When invoked with no arguments, `lumount` returns the name(s) of the mounted BEs on a system.

The `lumount` and `luumount` commands require root privileges or the Primary Administrator role.

Options The `lumount` and `luumount` commands have the following options:

`-f`	For `luumount` only, forcibly unmount a BE's file systems after attempting (and failing) an unforced unmount. This option is analogous to the `mount(1M)` `-f` option.
`-l` *error_log*	Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.
`-m` *mount_point*	`luumount` unmounts the file systems of the BE that owns *mount_point*. See description of *mount_point* under OPERANDS, below. The use of `-m` is optional when specifying a mount point for `luumount`.
`-n` *BE_name*	Name of the BE whose file systems will be unmounted. See description of *BE_name* under OPERANDS, below. The use of `-n` is optional when specifying a BE name for `luumount`.

`-o` *outfile* — All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-X` — Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

For `luumount`, if you supply an argument and specify neither `-m` nor `-n`, the command determines whether your argument is a BE name, a mount point, or a block device. If it is one of these three and the argument is associated with a BE that has mounted file systems, `luumount` unmounts the file systems of that BE. Otherwise, `luumount` returns an error.

Operands

BE_name — Name of the BE whose file systems will be mounted or unmounted. This is a BE on the current system other than the active BE. Note that, for successful completion of an `lumount` or `luumount` command, the status of a BE must be complete, as reported by `lustatus(1M)`. Also, none of the BE's disk slices can be mounted (through use of `mount(1M)`).

mount_point — For `lumount`, a mount point to use instead of the default `/.alt.`*BE_name*. If *mount_point* does not exist, `lumount` creates it. For `luumount`, the BE associated with *mount_point* will have its file systems unmounted. Note that default mount points are automatically deleted upon unmounting with `luumount`. Mount points that you specify are not deleted.

block_device — For `luumount` only, *block_device* is the root slice of a BE, such as `/dev/dsk/c0t4d0s0`. `luumount` unmounts the file systems of the BE associated with *block_device*.

Examples

EXAMPLE 1 Specifying a Mount Point

The following command creates the mount point `/test` and mounts the file systems of the BE `second_disk` on `/test`.

```
# lumount second_disk /test
/test
```

You can then `cd` to `/test` to view the file systems of `second_disk`. If you did not specify `/test` as a mount point, `lumount` would create a default mount point named `/.alt.second_disk`.

If you have installed non-global zones on your system, this command will also mount all non-global zones in `second_disk` inside their corresponding non-global zones in the currently running system at the mount point `/test` (or `/.alt.second_disk` if a mount point was not specified).

EXAMPLE 2 Unmounting File Systems

The following command unmounts the file systems of the BE `second_disk`. In this example, we `cd` to `/` to ensure we are not in any of the file systems in `second_disk`.

EXAMPLE 2 Unmounting File Systems *(Continued)*

```
# cd /
# luumount second_disk
#
```

If `/dev/dsk/c0t4d0s0` were the root slice for `second_disk`, you could enter the following command to match the effect of the preceding command.

```
# cd /
# luumount /dev/dsk/c0t4d0s0
#
```

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.<num>`	Live Upgrade DTD (see `-X` option)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also `luactivate(1M)`, `lucancel(1M)`, `lucompare(1M)`, `lucreate(1M)`, `lucurr(1M)`, `ludelete(1M)`, `ludesc(1M)`, `lufslist(1M)`, `lumake(1M)`, `lurename(1M)`, `lustatus(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`, `zones(5)`

Notes If a BE name contains slashes (/), `lumount` replaces those slashes with colons in a default mount point name. For example:

```
# lumount 'first/disk'
/.alt.first:disk
```

Name lurename – change the name of a boot environment

Synopsis

```
/usr/sbin/lurename -e BE_name -n new_name [-l error_log]
     [-o outfile] [-X]
```

Description The `lurename` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `lurename` command renames the boot environment (BE) *BE_name* to *new_name*.

The string *new_name* must not exceed 30 characters in length and must consist only of alphanumeric characters and other ASCII characters that are not special to the Unix shell. See the "Quoting" section of `sh(1)`. The BE name can contain only single-byte, 8–bit characters. It cannot contain whitespace characters. Also, *new_name* must be unique on the system.

A BE must have the status "complete" before you rename it. Use `lustatus(1M)` to determine a BE's status. Also, you cannot rename a BE that has file systems mounted with `lumount(1M)` or `mount(1M)`.

Renaming a BE is often useful when you upgrade the BE from one Solaris release to another. For example, following an operating system upgrade, you might rename the BE `solaris7` to `solaris8`.

The `lurename` command requires root privileges.

Options The `lurename` command has the options listed below.

`-e` *BE_name* — Name of the BE whose name you want to change.

`-l` *error_log* — Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.

`-n` *new_name* — New name of the BE. *new_name* must be unique on a given system.

`-o` *outfile* — All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-X` — Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Exit Status The following exit values are returned:

`0` — Successful completion.

`>0` — An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.<num>`	Live Upgrade DTD (see `-X` option)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also `luactivate(1M)`, `lucancel(1M)`, `lucompare(1M)`, `lucreate(1M)`, `lucurr(1M)`, `ludelete(1M)`, `ludesc(1M)`, `lufslist(1M)`, `lumake(1M)`, `lumount(1M)`, `lustatus(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`

Name lustatus – display status of boot environments

Synopsis

```
/usr/sbin/lustatus [-l error_log] [-o outfile] [BE_name]
     [-X]
```

Description The `lustatus` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `lustatus` command displays the status information of the boot environment (BE) *BE_name*. If no BE is specified, the status information for all BEs on the system is displayed.

The headings in the `lustatus` information display are described as follows:

`Boot Environment Name` Name of the BE.

`Is Complete` Indicates whether a BE is able to be booted. Any current activity or failure in an `lucreate(1M)` or `luupgrade(1M)` operation causes a BE to be incomplete. For example, if there is a copy operation proceeding on or scheduled for a BE, that BE is considered incomplete.

`Active Now` Indicates whether the BE is currently active. The "active" BE is the one currently booted.

`Active On Reboot` Indicates whether the BE becomes active upon next reboot of the system.

`Can Delete` Indicates that no copy, compare, or upgrade operations are being performed on a BE. Also, none of that BE's file systems are currently mounted. With all of these conditions in place, the BE can be deleted.

`Copy Status` Indicates whether the creation or repopulation of a BE is scheduled or active (that is, in progress). A status of `ACTIVE`, `COMPARING` (from `lucompare(1M)`), `UPGRADING`, or `SCHEDULED` prevents you performing Live Upgrade copy, rename, or upgrade operations.

The following is an example `lustatus` display:

```
Boot Environment     Is       Active Active    Can    Copy
Name                 Complete Now    On Reboot Delete Status
-------------------- -------- ------ --------- ------ ----------
disk_a_S7            yes      yes    yes       no     -
disk_b_S7db          yes      no     no        no     UPGRADING
disk_b_S8            no       no     no        no     -
S9testbed            yes      no     no        yes    -
```

Note that you could not perform copy, rename, or upgrade operations on `disk_b_S8`, because it is not complete, nor on `disk_b_S7db`, because a Live Upgrade operation is pending.

The `lustatus` command requires root privileges.

Options The `lustatus` command has the following options:

`-l` *error_log* Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.

`-o` *outfile* All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-X` Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Operands *BE_name* Name of the BE for which to obtain status. If *BE_name* is omitted, `lustatus` displays status for all BEs in the system.

Exit Status The following exit values are returned:

`0` Successful completion.

>`0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>	Live Upgrade DTD (see `-X` option)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also `luactivate(1M)`, `lucancel(1M)`, `lucompare(1M)`, `lucreate(1M)`, `lucurr(1M)`, `ludesc(1M)`, `ludelete(1M)`, `lufslist(1M)`, `lumake(1M)`, `lumount(1M)`, `lurename(1M)`, `luupgrade(1M)`, `lutab(4)`, `attributes(5)`, `live_upgrade(5)`

Name luupgrade – installs, upgrades, and performs other functions on software on a boot environment

Synopsis

```
/usr/sbin/luupgrade [-iIufpPtTcC] [options]
```

Description The `luupgrade` command is part of a suite of commands that make up the Live Upgrade feature of the Solaris operating environment. See `live_upgrade(5)` for a description of the Live Upgrade feature.

The `luupgrade` command enables you to install software on a specified boot environment (BE). Specifically, `luupgrade` performs the following functions:

- Upgrades an operating system image on a BE (`-u` option). The source for the image can be any valid Solaris installation medium, including a Solaris Flash archive.
- Extract a Solaris Flash archive onto a BE (`-f` option). (See `flar(1M)`.)
- Add a package to (`-p`) or remove a package from (`-P`) a BE.
- Add a patch to (`-t`) or remove a patch from (`-T`) a BE.
- Check (`-C`) or obtain information about (`-I`) packages.
- Check an operating system installation medium (`-c`).

Before using `luupgrade`, you must have created a BE, using the `lucreate(1M)` command. You can upgrade only BEs other than the current BE.

The functions described in the preceding list each has its own set of options, which are described separately for each function.

Note that, for successful completion of an `luupgrade` operation, the status of a BE must be complete, as reported by `lustatus(1M)`. Also, the BE must not have any mounted disk slices, mounted either with `lumount(1M)` or `mount(1M)`.

`luupgrade` allows you to install an operating system image from a different marketing release of the Solaris operating system from the release running on the machine from which you invoke `luupgrade`. This feature has the following conditions:

- You can install Live Upgrade packages (`SUNWluu`, `SUNWlur`, and `SUNWlucfg`) from a given release of the Solaris operating system on a machine running a previous release. You can install these packages on a machine running a version of Solaris that is up to three releases prior to the release of the Live Upgrade packages. Live Upgrade is not supported on Solaris releases prior to Solaris 2.6. Thus, you can, for example, install Solaris 2.9 packages on Solaris 2.8, 2.7, and 2.6 machines.
- You can upgrade to a release of the Solaris operating system that is the same as the release of the Live Upgrade packages installed on a machine. This feature allows you to upgrade to Solaris upgrade releases within a marketing release. For example, if have the Solaris 9 FCS Live Upgrade packages installed on a machine, you can use `luupgrade` to upgrade a BE to the Solaris 9 update 3 release of the Solaris operating system.

See the *Solaris Installation Guide* for instructions on installing Live Upgrade packages.

The `luupgrade` command requires root privileges.

Options that Apply to All Uses

The following options are available for all uses of `luupgrade`:

`-l` *error_log* — Error and status messages are sent to *error_log*, in addition to where they are sent in your current environment.

`-o` *outfile* — All command output is sent to *outfile*, in addition to where it is sent in your current environment.

`-N` — Dry-run mode. Enables you to determine whether your command arguments are correctly formed. Does not apply to the `-c` (check medium) function.

`-X` — Enable XML output. Characteristics of XML are defined in DTD, in `/usr/share/lib/xml/dtd/lu_cli.dtd.`<*num*>, where <*num*> is the version number of the DTD file.

Upgrading an Operating System Image

The `luupgrade` command uses `-u` to upgrade an operating system image. The syntax is as follows:

```
luupgrade -u -n BE_name -k autoreg_file [ -l error_log ]
     [ -o outfile ][-N] -s os_image_path [ -j profile_path [-D] ]
```

The first option, `-u`, indicates the function to perform—to install an OS image. The remaining options for this use of `luupgrade`, shown above, are described as follows:

`-n` *BE_name* — Name of the BE to receive an OS upgrade.

`-k` *autoreg_file* — Path to auto-registration information file. See `sysidcfg(4)` for a list of valid keywords for use in this file.

`-s` *os_image_path* — Path name of a directory containing an OS image. This can be a directory on an installation medium such as a DVD or CD, or can be an NFS or UFS directory.

`-j` *profile_path* — Path to a JumpStart profile. See the section "JumpStart Profile Keywords," below, for a list of valid keywords for use in a profile invoked by `luupgrade`. See `pfinstall(1M)` and the Solaris installation documentation for information on the JumpStart software.

`-D` — Tests the profile values provided with `-j` against the disk configuration of the specified BE. The upgrade is not performed. The effect of this option is a dry run to test your profile. `luupgrade` creates log files, specified in its output, which allow you to examine the command's results.

Before upgrading a boot environment, do the following:

- Run `analyze_patches`.

- Install Live Upgrade packages for the operating system version to which you are upgrading.

The `analyze_patches` command is available in the `/Misc` directory on the Solaris software DVD (formerly the Solaris installation CD). This command determines which patches would be removed as a result of the upgrade. Then, following the upgrade, you can reinstall the list of patches provided by `analyze_patches`.

The Live Upgrade packages, `SUNWluu`, `SUNWlur`, and `SUNWlucfg`, are available on the Solaris software DVD (or CD, depending on the Solaris version). Before running `luupgrade` with the `-u` option, ensure that you have installed the packages from the version of Solaris to which you want to upgrade.

Note that if you are upgrading from a medium with multiple components, such as from multiple DVDs, use `luupgrade` with the `-i` option, as described in the section below, to install software from the second and any following media.

Continuing an Upgrade by Running an Installer Program

The `luupgrade` command uses `-i` to run an installer program. As discussed below, its primary use is following an invocation of `luupgrade` with the `-u` option. The syntax for `-i` is as follows:

```
luupgrade -i -n BE_name [ -l error_log ] [ -o outfile ] [ -N ]
-s installation_medium [ -O "installer_options" ]
```

The first option, `-i`, indicates the function to perform—to run an installer program on the installation medium specified with `-s`. The remaining options for this use of `luupgrade`, shown above, are described as follows:

`-n` *BE_name*	Name of the BE on which software is to be installed.
`-O` "*installer_options*"	Options passed directly to the Solaris installer program. See `installer(1M)` for descriptions of the installer options.
`-s` *installation_medium*	Path name of an installation medium. This can be a DVD, CD, or an NFS or UFS directory.

With the `-i` option, `luupgrade` looks for an installation program on the specified medium and runs that program.

The primary use of the `-i` option is to upgrade an operating system image from a multiple–component medium, such as multiple DVDs. In this use, an `luupgrade` command with the `-i` option follows an invocation of `luupgrade` with `-u`. See EXAMPLES. The `-u` option is described above.

Installing from a Solaris Flash Archive

The `luupgrade` command uses `-f` to install an operating system from a Solaris Flash archive. Note that installing an archive overwrites all files on the target BE. The syntax is as follows:

```
luupgrade -f -n BE_name [ -l error_log ] [ -o outfile ] [ -N ] [-D]
-s os_image_path ( -a archive | -j profile_path | -J "profile" )
```

The first option, `-f`, indicates the function to perform—to install an OS from a Solaris Flash archive. The remaining options for this use of `luupgrade`, shown above, are described as follows:

`-n` *BE_name*	Name of the BE to receive an OS installation.
`-D`	Tests the profile values provided with `-j` or `-J` against the disk configuration of the specified BE. The upgrade is not performed. The effect of this option is a dry run to test your profile. `luupgrade` creates log files, specified in its output, which allow you to examine the command's results.
`-s` *os_image_path*	Path name of a directory containing an OS image. This can be a directory on an installation medium, such as a DVD or CD, or can be an NFS or UFS directory.
`-a` *archive*	Path to the Solaris Flash archive when the archive is available on the local file system. You must specify one of `-a`, `-j`, or `-J`.
`-j` *profile_path*	Path to a JumpStart profile that is configured for a Solaris Flash installation. See the section "JumpStart Profile Keywords," below, for a list of valid keywords for use in a profile invoked by `luupgrade`. See `pfinstall(1M)` and the Solaris installation documentation for information on the JumpStart software. You must specify one of `-a`, `-j`, or `-J`.
`-J` *"profile"*	Entry from a JumpStart profile that is configured for a Solaris Flash installation. The only valid keyword for this option is `archive_location`. See `pfinstall(1M)` and the Solaris installation documentation for information on the JumpStart software. You must specify one of `-a`, `-j`, or `-J`.

Note that the version of the OS image specified with `-s` must be identical to the version of the OS contained in the Solaris Flash archive specified with the `-a`, `-j`, or `-J` options.

Adding or Removing Packages

The `luupgrade` command uses `-p` to add a package and `-P` to remove a package. The syntax is as follows:

For adding packages:

```
luupgrade -p -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
 -s packages_path [ -O "pkgadd_options" ] [ -a admin ]
[ pkginst [ pkginst...]]
```

For removing packages:

```
luupgrade -P -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
[ -O "pkgrm_options" ] [ pkginst [ pkginst...]]
```

The first option, `-p`, to add packages, or `-P` to remove packages, indicates the function to perform. The remaining options for this use of `luupgrade`, shown above, are described as follows:

`-n` *BE_name*	Name of the BE to which packages will be added or from which packages will be removed.
`-s` *packages_path*	(For adding packages only.) Path name of a directory containing packages to add. You can substitute `-d` for `-s`. The `-d` support is for `pkgadd(1M)` compatibility.
`-d` *packages_path*	Identical to `-s`. Use of `-s` is recommended.
`-O` "*pkgadd_options*" or "*pkgrm_options*"	Options passed directly to `pkgadd` (for `-p`) or `pkgrm` (for `-P`). See `pkgadd(1M)` and `pkgrm(1M)` for descriptions of the options for those commands.
`-a` *admin*	(For adding packages only.) Path to an admin file. Identical to the `pkgadd -a` option. Use of the `-a` option here is identical to `-O` "`-a` *admin*"
pkginst [*pkginst...*]	Zero or more packages to add or remove. For adding packages, the default is to add all of the packages specified with the `-s` option, above. Separate multiple package names with spaces.

It is critically important that any packages you add be compliant with the SVR4 Advanced Packaging Guidelines. See WARNINGS, below.

Adding or Removing Patches

The `luupgrade` command uses `-t` to add a patch and `-T` to remove a patch. The syntax is as follows:

For adding patches:

```
luupgrade -t -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
-s patch_path [ -O "patchadd_options" ] [ patch_name [ patch_name...]]
```

For removing patches:

```
luupgrade -T -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
[ -O "patchrm_options" ] [ patch_name [ patch_name...]]
```

The first option, `-t`, to add patches, or `-T` to remove patches, indicates the function to perform. The remaining options for this use of `luupgrade`, shown above, are described as follows:

`-n` *BE_name*	Name of the BE to which patches will be added or from which patches will be removed.
`-s` *patch_path*	(For adding patches only.) Path name of a directory containing patches to add or path name of a patch_order file.
`-O` "*patchadd_options*" or "*patchrm_options*"	Options passed directly to `patchadd` (for `-p`) or `patchrm` (for `-P`). See `patchadd(1M)` or `patchrm(1M)` for a description of these options.
patch_name [*patch_name...*]	Zero or more patches to add or remove. For adding patches, the default is to add all of the patches specified with the `-s` option, above. Separate multiple patch names with spaces.

It is critically important that any patches you add be compliant with the SVR4 Advanced Packaging Guidelines. See WARNINGS, below.

Checking or Returning Information on Packages

Use the `-C` to perform a `pkgchk(1M)` on all or the specified packages on a BE. Use the `-I` option to perform a `pkginfo(1)`.

For performing a `pkgchk`:

```
luupgrade -C -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
[ -O "pkgchk_options" ][ pkginst [ pkginst...]]
```

For performing a `pkginfo`:

```
luupgrade -I -n BE_name [ -l error_log ][ -o outfile ] [ -N ]
[ -O "pkginfo_options" ][ pkginst [ pkginst...]]
```

The first option, `-C`, for `pkgchk`, or `-I`, for `pkginfo`, indicates the function to perform. The remaining options for this use of `luupgrade`, shown above, are described as follows:

`-n` *BE_name*	Name of the BE on which packages will be checked or on whose packages information will be returned.
`-O` "*pkgchk_options*" or "*pkginfo_options*"	Options passed directly to `pkgchk` (for `-C`) or `pkginfo` (for `-I`). See `pkgchk(1M)` or `pkginfo(1)` for a description of these options.
pkginst [*pkginst...*]	Zero or more packages to check or for which to have information returned. If you omit package names, `luupgrade` returns information on all of the packages on the BE. Separate multiple

package names with spaces.

Checking an OS Installation Medium

With the `-c` option, `luupgrade` allows you to check that a local or remote medium, such as a DVD or CD, is a valid installation medium. The `-c` option returns useful information about the specified medium. The syntax for this use of `luupgrade` is as follows:

```
luupgrade -c [ -l error_log ] [ -o outfile ] -s path_to_medium
```

The first option, `-c`, indicates the function to perform—to check on an installation medium. The `-s` option, shown above, is described as follows:

`-s` *path_to_medium* Path name to an installation medium such as a DVD or CD.

JumpStart Profile Keywords

This section specifies the Solaris JumpStart keywords that can be used in a profile with `luupgrade`, using the `-j` option in conjunction with the `-u` (upgrade) or `-f` (flash) options. For `-u`, there are no required keywords. For `-f`, you must specify a value for `install_type`: `flash_install` for a full flash archive or `flash_update` for a differential flash archive. Also for the `-f` option with the `-j` option, you must specify the `-a` (archive location) option or specify the `archive_location` keyword in your profile.

The `archive_location` keyword is the only valid argument for the `-J` option.

The following optional keywords are sometimes used in profiles used with the `-u` and `-f` options:

`cluster` Designates the software group to add to the system.

`geo` Designates the regional locale or locales that you want to install on or add to a system. See the *Solaris Installation Guide* for a list of possible values.

`isa_bits` Specifies whether 64–bit or 32–bit packages are to be installed. Valid values are `64` and `32`.

`locale` Designates the locale packages you want to install on or add to a system. See the *Solaris Installation Guide* for a list of possible values.

`package` Specifies a package to be added to or deleted from a system.

The following keywords must not be used in a profile used with `luupgrade`:

- `boot_device`
- `dontuse`
- `fdisk`
- `filesys`
- `layout_constraint`
- `noreboot`
- `partitioning`
- `root_device`
- `usedisk`

See the *Solaris Installation Guide* for descriptions of all JumpStart profile keywords and instructions for creating a JumpStart profile.

Examples

EXAMPLE 1 Removing, then Adding Packages

The following example removes from then adds a set of packages to a boot environment.

```
# luupgrade -P -n second_disk SUNWabc SUNWdef SUNWghi
```

Now, to add the same packages:

```
# luupgrade -p -n second_disk -s /net/installmachine/export/packages \
SUNWabc SUNWdef SUNWghi
```

The following command adds the `-O` option to the preceding command. This option passes arguments directly to `pkgadd`.

```
# luupgrade -p -n second_disk -s /net/installmachine/export/packages \
-O "-r /net/testmachine/export/responses" SUNWabc SUNWdef SUNWghi
```

See `pkgadd(1M)` for a description of the options for that command.

EXAMPLE 2 Upgrading to a New OS from a Combined Image

The following example upgrades the operating environment on a boot environment. The source image is stored as a combined image on a remote disk or on a DVD.

```
# luupgrade -u -n second_disk \
-s /net/installmachine/export/solarisX/OS_image
```

Following the command above you could enter the command below to activate the upgraded BE.

```
# luactivate second_disk
```

Then, upon the next reboot, `second_disk` would become the current boot environment. See `luactivate(1M)`.

EXAMPLE 3 Upgrading to a New OS from Multiple CDs

The following example is a variation on the preceding. The OS upgrade resides on two CDs. To begin the upgrade on a SPARC machine, you enter:

```
# luupgrade -u -n second_disk -s /cdrom/cdrom0/s0
```

On x86 machines, replace the `s0` in the argument to `-s` with `s2`.

When the installer is finished with the contents of the first CD, insert the next CD in the drive and enter the following:

EXAMPLE 3 Upgrading to a New OS from Multiple CDs *(Continued)*

```
# luupgrade -i -n second_disk -s /cdrom/cdrom0 \
-O "-nodisplay -noconsole"
```

Note the use of `-i` rather than `-u` in the preceding. Were there additional CDs, you would enter the same command as the one immediately above. The `-O` options, above, are passed to `installer(1M)`. If you omit these options, a graphical interface is invoked following the insertion and reading of the second CD. See `installer(1M)` for a description of the `-O` options.

Note that a multiple-CD upgrade is not complete until you have entered and completed `luupgrade` commands for all of the CDs in a set. Following installation of packages from a CD, you might receive a message such as:

`WARNING: <`*num*`> packages must be installed on boot environment <`*disk_device*`>.`

Such a message indicates the requirement that you install packages from one or more additional CDs, as in the example above. If you do not complete package installation, you will not be able to use `luactivate` to activate (designate for booting) the upgraded BE.

EXAMPLE 4 Upgrading Using a JumpStart Profile

The following example command uses the `-D` option to test the profile `/home2/profiles/test.profile`.

```
# luupgrade -u -n second_disk \
-s /net/installmachine/export/solarisX/OS_image \
-j /home2/profiles/test.profile -D
```

Assuming the results of this command were acceptable, you could omit the `-D` in the preceding command to perform the upgrade.

EXAMPLE 5 Installing a New OS from a Solaris Flash Archive

The following example installs the operating environment on a boot environment, using a Solaris Flash archive. The file pointed to by `-J` is a JumpStart profile that specifies a flash installation.

```
# luupgrade -f -n second_disk \
-s /net/installmachine/export/solarisX/OS_image \
-J "archive_location http://example.com/myflash.flar"
```

The following command differs from the preceding only in that `-j` replaces `-J`. You could append the `-D` option to either of these commands to test the profile prior to actually performing the flash installation.

```
# luupgrade -f -n second_disk \
-s /net/installmachine/export/solarisX/OS_image \
```

EXAMPLE 5 Installing a New OS from a Solaris Flash Archive *(Continued)*

```
-j /net/example/flash_archives/flash_gordon
```

Either of the preceding commands works for a full or differential flash installation. Whether a flash installation is differential or full is determined by the value of the `install_type` keyword in the profile. See "JumpStart Profile Keywords," above.

EXAMPLE 6 Obtaining Information on Packages

The following example runs a `pkgchk` on the packages `SUNWluu`, `SUNWlur`, and `SUNWlucfg`, passing to `pkgchk` the `-v` option.

```
# luupgrade -C -n second_disk -O "-v" SUNWluu SUNWlur SUNWlucfg
```

The following command runs `pkginfo` on the same set of packages:

```
# luupgrade -I -n second_disk -O "-v" SUNWluu SUNWlur SUNWlucfg
```

For both commands, if the package names were omitted, `luupgrade` returns package information on all of the packages in the specified BE. See `pkgchk(1M)` and `pkginfo(1)` for a description of the options for those commands.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/etc/lutab`	list of BEs on the system
`/usr/share/lib/xml/dtd/lu_cli.dtd.<num>`	Live Upgrade DTD (see `-X` option in "Options that Apply to All Uses," above)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluu

See Also `installer(1M)`, `luactivate(1M)`, `lucancel(1M)`, `lucompare(1M)`, `lucreate(1M)`, `lucurr(1M)`, `ludelete(1M)`, `ludesc(1M)`, `lufslist(1M)`, `lumake(1M)`, `lumount(1M)`, `lurename(1M)`, `lustatus(1M)`, `lutab(4)`, `sysidcfg(4)`, `attributes(5)`, `live_upgrade(5)`, `zones(5)`

Warnings For adding packages or patches (`-p`, `-P`, `-t`, or `-T`), `luupgrade` requires packages or patches that comply with the SVR4 Advanced Packaging Guidelines and the guidelines spelled out in Appendix C of the *Solaris 10 Installation Guide: Basic Installations*. This means that the package or patch is compliant with the `pkgadd(1M)` or `patchadd(1M)` `-R` option, described in

the man pages for those utilities. While nearly all Sun packages and patches conform to these guidelines, Sun cannot guarantee the conformance of packages and patches from third-party vendors. Some older Sun packages and patches might not be `-R` compliant. If you encounter such a package or patch, please report it to Sun. A non-conformant package can cause the package- or patch-addition software in `luupgrade` to fail or, worse, alter the current BE.

Notes For versions of the Solaris operating system prior to Solaris 10, Live Upgrade supports the release it is distributed on and up to three marketing releases back. For example, if you obtained Live Upgrade with Solaris 9 (including a Solaris 9 upgrade), that version of Live Upgrade supports Solaris versions 2.6, Solaris 7, and Solaris 8, in addition to Solaris 9. No version of Live Upgrade supports a Solaris version prior to Solaris 2.6.

Starting with version 10 of the Solaris operating system, Live Upgrade supports the release it is distributed on and up to two marketing releases back. For example, if you obtained Live Upgrade with Solaris 10 (including a Solaris 10 upgrade), that version of Live Upgrade supports Solaris 8 and Solaris 9, in addition to Solaris 10.

Correct operation of Solaris Live Upgrade requires that a limited set of patch revisions be installed for a given OS version. Before installing or running Live Upgrade, you are required to install the limited set of patch revisions. Make sure you have the most recently updated patch list by consulting `http://sunsolve.sun.com`. Search for the infodoc 72099 on the SunSolve web site.

Name luxadm – administer Sun Fire 880 storage subsystem and FC_AL devices

Synopsis

```
luxadm [options]... subcommand [options]... enclosure
     [,dev] | pathname...
```

Description The `luxadm` program is an administrative command that manages the SENA, Sun Fire 880 internal storage subsystem, and individual Fiber Channel Arbitrated Loop (FC_AL) devices. `luxadm` performs a variety of control and query tasks depending on the command line arguments and options used.

The command line must contain a subcommand. The command line may also contain options, usually at least one enclosure name or pathname, and other parameters depending on the subcommand. You need specify only as many characters as are required to uniquely identify a subcommand.

Specify the device that a subcommand interacts with by entering a pathname. For the SENA subsystem, a disk device or enclosure services controller may instead be specified by entering the World Wide Name (WWN) for the device or a port to the device. The device may also be specified by entering the name of the SENA enclosure, and an optional identifier for the particular device in the enclosure. The individual FC_AL devices may be specified by entering the WWN for the device or a port to the device.

Pathname Specify the device or controller by either a complete physical pathname or a complete logical pathname.

For SENA, a typical physical pathname for a device is:

```
/devices/sbus@1f,0/SUNW,socal@1,0/sf@0,0/ssd@w2200002037000f96,
         0:a,raw
```

For all SENA IBs (Interface Boards) and Sun Fire 880 SES device controllers on the system, a logical link to the physical paths is kept in the directory `/dev/es`. An example of a logical link is `/dev/es/ses0`.

The WWN may be used in place of the pathname to select an FC_AL device, SENA subsystem IB, or Sun Fire 880 internal storage subsystem. The WWN is a unique 16 hexadecimal digit value that specifies either the port used to access the device or the device itself. A typical WWN value is:

```
2200002037000f96
```

See NOTES for more information on the WWN formats.

For a disk in a Sun Fire 880 internal storage subsystem, a typical physical pathname is:

```
/devices/pci@8,600000/SUNW,qlc@2/fp@0,0/ssd@w2100002037a6303c,0:a
```

and a typical logical pathname is:

```
/dev/rdsk/c2t8d0s2
```

For individual FC_AL devices, a typical physical pathname is:

```
/devices/sbus@3.0/SUNW,socal@d,10000/sf@0,0/ssd@w2200002037049fc3,0:a,raw
```

and a typical logical pathname is:

```
/dev/rdsk/c1t0d0s2
```

Enclosure For SENA, a device may be identified by its enclosure name and slotname:

box_name[,f*slot_number*]
box_name[,r*slot_number*]

box_name is the name of the SENA enclosure, as specified by the `enclosure_name` subcommand. When used without the optional *slot_number* parameter, the *box_name* identifies the SENA subsystem IB.

`f` or `r` specifies the front or rear slots in the SENA enclosure.

slot_number specifies the slot number of the device in the SENA enclosure, `0-6` or `0-10`.

For a Sun Fire 880 internal storage subsystem, a device may also be identified by its enclosure name and slot name. However, there is only one set of disks:

box_name[,s*slot_number*]

box_name is the name of the Sun Fire 880 enclosure, as specified by the `enclosure_name` subcommand. When used without the optional *slot_number* parameter, *box_name* identifies the Sun Fire 880 internal storage subsystem enclosure services device. Use *s* to specify the disk slot number in the Sun Fire 880 internal storage subsystem, `0 - 11`.

See `disks(1M)` and `devlinks(1M)` for additional information on logical names for disks and subsystems.

Options The following options are supported by all subcommands:

`-e` Expert mode. This option is not recommended for the novice user.

`-v` Verbose mode.

Options that are specific to particular subcommands are described with the subcommand in the `USAGE` section.

Operands The following operands are supported:

enclosure The *box_name* of the SENA or Sun Fire 880 internal storage subsystem.

fibre_channel_HBA_port The path to the host controller port. A typical path is:

```
/devices/pci@8,600000/pci@1/SUNW,qlc@4/fp@0,0:devctl
```

pathname The logical or physical path of a SENA IB, Sun Fire 880 internal storage subsystem, or disk device. *pathname* can also be the WWN of a SENA IB, SENA disk, or individual FC_AL device.

Usage

Subcommands

`display` *enclosure*[,*dev*] . . . | *pathname* . . .
`display -p` *pathname* . . .
`display -r` *enclosure*[,*dev*] . . . | *pathname* . . .
`display -v` *enclosure*[,*dev*] . . . | *pathname* . . .

Displays enclosure or device specific data.

Subsystem data consists of enclosure environmental sense information and status for all subsystem devices, including disks.

Disk data consists of inquiry, capacity, and configuration information.

`-p` Displays performance information for the device or subsystem specified by *pathname*. This option only applies to subsystems that accumulate performance information.

`-r` Displays error information for the FC_AL device specified by the pathname, or, if the path is a SENA, for all devices on the loop. The `-r` option only applies to SENA subsystems and individual FC_AL devices.

`-v` Displays in verbose mode, including mode sense data.

`download` [`-s`] [`-f` *filename_path*] *enclosure*. . .

Download the prom image pointed to the SENA subsystem Interface Board unit or the Sun Fire 880 internal storage subsystem specified by the enclosure or pathname.

When the SENA's download is complete, the SENA will be reset and the downloaded code executed. If no

filename is specified, the default prom image will be used. The default prom image for the SENA is in the directory `usr/lib/locale/C/LC_MESSAGES` and is named `ibfirmware`

When the Sun Fire 880 internal storage subsystem's download is complete, the subsystem resets and the downloaded code begins execution. The default firmware image for the Sun Fire 880 internal storage subsystem is in: `/usr/platform/SUNW,Sun-Fire-880/lib/image`

`-s` Save. The `-s` option is used to save the downloaded firmware in the FEPROM. If `-s` is not specified, the downloaded firmware will not be saved across power cycles.

The `-s` option does not apply to the Sun Fire 880 internal storage subsystem as it always stores downloaded firmware in the flash memory.

When using the `-s` option, the `download` subcommand modifies the FEPROM on the subsystem and should be used with *caution*.

`enclosure_name` *new_name enclosure* | *pathname*

Change the enclosure name of the enclosure or enclosures specified by the enclosure or pathname. The new name (*new_name*) must be 16 or less characters. Only alphabetic or numeric characters are acceptable. This subcommand applies only to the SENA and the Sun Fire 880 internal storage subsystem.

`failover primary` | `secondary` *pathname*

Select which Sun Storage T3 storage array partner group controller accesses a given logical volume. If `primary` is specified, the logical volume is accessed through the primary controller. If

`secondary` is specified, the logical volume is accessed through the secondary controller specified by *pathname*.

`fcal_s_download` [`-f` *fcode-file*]

Download the fcode contained in the file *fcode-file* into *all* the FC100/S Sbus Cards. This command is interactive and expects user confirmation before downloading the fcode.

Use `fcal_s_download` *only* in single-user mode. Using `fcal_s_download` to update a host adapter while there is I/O activity through that adapter *will* cause the adapter to reset. Newly updated FCode will not be executed or visible until a system reboot.

`-f` *fcode-file* — When invoked without the `-f` option, the current version of the fcode in each FC100/S Sbus card is printed.

`fcode_download -p`
`fcode_download -d` *dir-name*

Locate the installed FC/S, FC100/S, FC100/P, or FC100/2P host bus adapter cards and download the FCode files in *dir-name* to the appropriate cards. The command determines the correct card for each type of file, and is interactive. User confirmation is required before downloading the FCode to each device.

Use `fcode_download` to load FCode only in single-user mode. Using `fcode_download` to update a host adapter while there is I/O activity through that adapter causes the adapter to reset. Newly updated FCode will not be executed or visible until a system reboot.

`-d` *dir-name* — Download the FCode files contained in the directory *dir-name* to the appropriate adapter cards.

`-p` — Prints the current version of FCode loaded on each card. No download is performed.

`inquiry` *enclosure*[,*dev*] . . . | *pathname* . . . — Display the inquiry information for the selected device specified by the enclosure or pathname.

`insert_device` [*enclosure*,*dev* . . .] — Assist the user in the hot insertion of a new device or a chain of new devices. Refer to `NOTES` for limitations on hotplug operations. This subcommand applies only to the SENA, Sun Fire 880 internal storage subsystem, and individual FC_AL drives. For the SENA, if more than one enclosure has been specified, concurrent hot insertions on multiple busses can be performed. With no arguments to the subcommand, entire enclosures or individual FC_AL drives can be inserted. For the SENA or the Sun Fire 880 internal storage subsystem, this subcommand guides the user interactively through the hot insertion steps of a new device or chain of devices. If a list of disks was entered it will ask the user to verify the list of devices to be inserted is correct, at which point the user can continue or quit. It then interactively asks the user to insert the disk(s) or enclosure(s) and then creates and displays the logical pathnames for the devices.

`led` *enclosure*,*dev* . . . | *pathname*. . .
Display the current state of the LED associated with the disk specified by the enclosure or pathname. This subcommand only applies to subsystems that support this functionality.

`led_blink` *enclosure,dev . . .* | *pathname . . .*
Requests the subsystem to start blinking the LED associated with the disk specified by the enclosure or pathname. This subcommand only applies to subsystems that support this functionality.

`led_off` *enclosure,dev . . .* | *pathname . . .*
Requests the subsystem to disable (turn off) the LED associated with the disk specified by the enclosure or pathname. On a SENA subsystem, this may or may not cause the LED to turn off or stop blinking depending on the state of the SENA subsystem. Refer to the SENA Array Installation and Service Manual (p/n 802-7573). This subcommand only applies to subsystems that support this functionality.

`led_on` *pathname . . .*
Requests the subsystem to enable (turn on) the LED associated with the disk specified by the pathname. This subcommand only applies to subsystems that support this functionality.

`power_off` [`-F`] *enclosure*[,*dev*] . . . | *pathname . . .*
When a SENA is addressed, this subcommand causes the SENA subsystem to go into the power-save mode. The SENA drives are not available when in the power-save mode. When a drive in a SENA is addressed the drive is set to the drive off/unmated state. In the drive off/unmated state, the drive is spun down (stopped) and in bypass mode. This command does not apply to the Sun Fire 880 internal storage subsystem.

`-F` The force option only applies to the SENA. Instructs `luxadm` to attempt to power off one or more devices even if those devices are being used by this host (and are, therefore, busy).

Warning: Powering off a device which has data that is currently being used will cause unpredictable results. Users should attempt to power off the device normally (without `-F`) first, only resorting to this option when sure of the consequences of overriding normal checks.

`power_on` *enclosure*[`,`*dev*] . .
Causes the SENA subsystem to go out of the power-save mode, when this subcommand is addressed to a SENA.. When this subcommand is addressed to a drive the drive is set to its normal start-up state. This command does not apply to the Sun Fire 880 internal storage subsystem.

`probe` [`-p`]
Finds and displays information about all attached SENA subsystems, Sun Fire 880 internal storage subsystems, and individual FC_AL devices, including the logical pathname, the WWNs, and enclosure names. This subcommand warns the user if it finds different SENAs with the same enclosure names.

`-p` Includes the physical pathname in the display.

`qlgc_s_download` [`-f` *fcode-file*]
Download the FCode contained in the file *fcode-file* into all the FC100/P, FC100/2P PCI host adapter cards. This command is interactive and expects user confirmation before downloading the FCode to each device. Only use `qlgc_s_download` in single-user mode. Using `qlgc_s_download` to update a host adapter while there is I/O activity through that adapter will cause the adapter to reset. Newly updated FCode will not be executed or visible until a system reboot.

`-f` *fcode-file* When invoked without the `-f` option, the current version of the FCode in each FC100/P, FC100/2P PCI card is printed.

`release` *pathname*
Release a reservation held on the specified disk. The pathname should be the physical or logical pathname for the disk.

This subcommand is included for historical and diagnostic purposes only.

`remove_device` [`-F`] *enclosure*[,*dev*] . . . | *pathname* . . .
Assists the user in hot removing a device or a chain of devices. This subcommand can also be used to remove entire enclosures. This subcommand applies to the SENA, Sun Fire 880 internal storage subsystem, and individual FC_AL drives. Refer to `NOTES` for limitations on hotplug operations. For the SENA, Sun Fire 880 internal storage subsystem, and individual FC_AL devices, this subcommand guides the user through the hot removal of a device or devices. During execution it will ask the user to verify the list of devices to be removed is correct, at which point the user can continue or quit. It then prepares the disk(s) or enclosure(s) for removal and interactively asks the user to remove the disk(s) or enclosure(s).

For Multi-Hosted disk, the steps taken are:

- Issue the `luxadm remove_device` command on the first host. When prompted to continue, wait.
- Issue the `luxadm remove_device` command on the secondary hosts. When prompted to continue, wait.
- Continue with the `remove_device` command on the first host. Remove the device when prompted to do so.
- Complete the `luxadm remove_device` command on the additional hosts.

`-F` Instructs `luxadm` to attempt to hot plug one or more devices even if those devices are being used by this host (and are, therefore, *busy* or *reserved*), to *force* the hotplugging operation.

Warning: Removal of a device which has data that is currently being used will cause unpredictable results. Users should attempt to hotplug normally (without `-F`) first, only resorting to this option when sure of the consequences of overriding normal hotplugging checks.

`reserve` *pathname*
: Reserve the specified disk for exclusive use by the issuing host. The pathname used should be the physical or logical pathname for the disk.

  This subcommand is included for historical and diagnostic purposes only.

`set_boot_dev` [`-y`] *pathname*
: Set the boot-device variable in the system PROM to the physical device name specified by *pathname*, which can be a block special device or the pathname of the directory on which the boot file system is mounted. The command normally runs interactively requesting confirmation for setting the default boot-device in the PROM. The `-y` option can be used to run it non-interactively, in which case no confirmation is requested or required.

`start` *pathname*
: Spin up the specified disk(s) in a SENA.

`stop` *pathname...*
: Spin down the specified disks in a SENA.

SENA, Sun Fire 880 Internal Storage Subsystem, and Individual FC_AL Drive Expert Mode Subcommands

The following subcommands are for expert use only, and are applicable only to the SENA, Sun Fire 880 internal storage subsystem, and fiber channel loops. They should only be used by users that are knowledgeable about the SENA subsystem and fiber channel loops.

If you specify a disk to an expert subcommand that operates on a bus, the subcommand operates on the bus to which the specified disk is attached.

`-e bypass [-ab]` *enclosure,dev*
`-e bypass -f` *enclosure*
: Request the enclosure services controller to set the LRC (Loop Redundancy Circuit) to the bypassed state for the port and device specified.

  This subcommand supports the following options:

  `-a`
  : Bypass port `a` of the device specified.

  `-b`
  : Bypass port `b` of the device specified.

`-e dump_map` *fibre_channel_HBA_port*
: Display WWN data for a target device or host bus adapter on the specified fibre channel port. If there are no target devices on the specified port, an error is returned.

`-e enable [-ab]` *enclosure,dev*

`-e enable -f` *enclosure*

Request the enclosure services controller to set the LRC (Loop Redundancy Circuit) to the enabled state for the port and device specified.

This subcommand supports the following options:

`-a` Enable port `a` of the device specified.

`-b` Enable port `b` of the device specified.

`-e forcelip` *enclosure*[`,`*dev*] `...` | *pathname* `...`

Force the link to reinitialize, using the Loop Initialization Primitive (LIP) sequence. The enclosure or pathname can specify any device on the loop. Use the pathname to specify a specific path for multiple loop configurations.

This is an expert only command and should be used with caution. It will reset all ports on the loop.

`-e rdls` *enclosure*[`,`*dev*] `...` | *pathname* `...`

Read and display the link error status information for all available devices on the loop that contains the device specified by the enclosure or pathname.

Other Expert Mode Subcommands

See `NOTES` for limitations of these subcommands. They should only be used by users that are knowledgeable about the systems they are managing.

These commands do not apply to the Sun Fire 880 internal storage subsystem.

`-e bus_getstate` *pathname* Get and display the state of the specified bus.

`-e bus_quiesce` *pathname* Quiesce the specified bus.

`-e bus_reset` *pathname* Reset the specified bus only.

`-e bus_resetall` *pathname* Reset the specified bus and all devices.

`-e bus_unquiesce` *pathname* Unquiesce the specified bus. the specified device.

`-e dev_getstate` *pathname* Get and display the state of the specified device.

`-e dev_reset` *pathname* Reset the specified device.

`-e offline` *pathname*	Take the specified device offline.
`-e online` *pathname*	Put the specified device online.

Examples

EXAMPLE 1 Displaying the SENAs and Individual FC_AL Devices on a System

The following example finds and displays all of the SENAs and individual FC_AL devices on a system:

```
example% luxadm probe
```

EXAMPLE 2 Displaying a SENA or Sun Fire 880 Internal Storage Subsystem

The following example displays a SENA or Sun Fire 880 internal storage subsystem:

```
example% luxadm display /dev/es/ses0
```

EXAMPLE 3 Displaying Two Subsystems

The following example displays two subsystems using the enclosure names:

```
example% luxadm display BOB system1
```

EXAMPLE 4 Displaying Information about the First Disk

The following example displays information about the first disk in the front of the enclosure named `BOB`. Use `f` to specify the front disks. Use `r` to specify the rear disks.

```
example% luxadm display BOB,f0
```

EXAMPLE 5 Displaying Information on a Sun Fire 880 Internal Storage Subsystem

The Sun Fire 880 internal storage subsystem has only one set of disks. In this case, use *s* to specify the slot:

```
example% luxadm display BOB,s0
```

EXAMPLE 6 Displaying Information about a SENA disk, an Enclosure, or an Individual FC_AL Drive

The following example displays information about a SENA disk, an enclosure, or an individual FC_AL drive with the port WWN of `2200002037001246`:

```
example% luxadm display 2200002037001246
```

EXAMPLE 7 Using Unique Characters to Issue a Subcommand

The following example uses only as many characters as are required to uniquely identify a subcommand:

```
example% luxadm disp BOB
```

EXAMPLE 8 Displaying Error Information

The following example displays error information about the loop that the enclosure BOB is on:

```
example% luxadm display -r BOB
```

EXAMPLE 9 Downloading New Firmware into the Interface Board

The following example downloads new firmware into the Interface Board in the enclosure named BOB (using the default path for the file to download):

```
example% luxadm download -s BOB
```

EXAMPLE 10 Displaying Information from the SCSI Inquiry Command

The following example displays information from the SCSI inquiry command from all individual disks on the system, using only as many characters as necessary to uniquely identify the inquiry subcommand:

```
example% luxadm inq /dev/rdsk/c?t?d?s2
```

EXAMPLE 11 Hotplugging

The following example hotplugs a new drive into the first slot in the front of the enclosure named BOB:

```
example% luxadm insert_device BOB,f0
```

The following example hotplugs a new drive into the first slot in the Sun Fire 880 internal storage subsystem named SF880-1:

```
example% luxadm insert_device SF880-1,s0
```

EXAMPLE 12 Running an Expert Subcommand

The following example runs an expert subcommand. The subcommand forces a loop initialization on the loop that the enclosure BOB is on:

```
example% luxadm -e forcelip BOB
```

EXAMPLE 13 Using the Expert Mode Hot Plugging Subcommands

An example of using the expert mode hot plugging subcommands to hot remove a disk follows. See NOTES for hot plugging limitations.

The first step reserves the SCSI device so that it can't be accessed by way of its second SCSI bus:

```
example# luxadm reserve /dev/rdsk/c1t8d0s2
```

EXAMPLE 14 Taking the Disk to be Removed Offline

The next two steps take the disk to be removed offline then quiesce the bus:

EXAMPLE 14 Taking the Disk to be Removed Offline *(Continued)*

```
example# luxadm -e offline /dev/rdsk/c1t8d0s2
example# luxadm -e bus_quiesce /dev/rdsk/c1t8d0s2
```

EXAMPLE 15 Unquiescing the Bus

The user then removes the disk and continues by unquiescing the bus, putting the disk back online, then unreserving it:

```
example# luxadm -e bus_unquiesce /dev/rdsk/c1t8d0s2
example# luxadm -e online /dev/rdsk/c1t8d0s2
example# luxadm release /dev/rdsk/c1t8d0s2
```

Environment Variables

See `environ(5)` for a description of the `LANG` environment variable that affects the execution of `luxadm`.

Exit Status

The following exit values are returned:

`0` Successful completion.

`–1` An error occurred.

Files

`usr/lib/firmware/fc_s/fc_s_fcode`

`usr/lib/locale/C/LC_MESSAGES/ibfirmware`

Attributes

See `attributes(5)` for descriptions of the following attributes:

usr/sbin

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWluxop

See Also

`devlinks(1M)`, `disks(1M)`, `attributes(5)`, `environ(5)`, `ses(7D)`

SENA Array Installation and Service Manual (p/n 802-7573).

RAID Manager 6.1 Installation and Support Guide Answerbook

RAID Manager 6.1 User's Guide Answerbook

Notes

See the *SENA Array Installation and Service Manual* for additional information on the SENA. Refer to *Tutorial for SCSI use of IEEE Company_ID*, R. Snively, for additional information regarding the IEEE extended WWN. See `SEE ALSO`. Currently, only some device drivers support hot plugging. If hot plugging is attempted on a disk or bus where it is not supported, an error message of the form:

```
luxadm: can't acquire "PATHNAME": No such file or directory
```

will be displayed.

You must be careful not to quiesce a bus that contains the root or the `/usr` filesystems or any swap data. If you do quiesce such a bus a deadlock can result, requiring a system reboot.

Name m64config, SUNWm64_config – configure the M64 Graphics Accelerator

Synopsis

```
/usr/sbin/m64config [-defaults] [-depth 8 | 24 | 32]
     [-dev device-filename] [-file machine | system]
     [-prconf] [-propt]
     [-res video-mode [now | try] [noconfirm | nocheck ]]

/usr/sbin/m64config [-prconf] [-propt]

/usr/sbin/m64config [-help] [-res ?]
```

Description `m64config` configures the M64 Graphics Accelerator and some of the X11 window system defaults for M64.

The first form of `m64config` stores the specified options in the OWconfig file. These options will be used to initialize the M64 device the next time the window system is run on that device. Updating options in the OWconfig file provides persistence of these options across window system sessions and system reboots.

The second and third forms which invoke only the `-prconf`, `-propt`, `-help`, and `-res ?` options do not update the OWconfig file. Additionally, for the third form all other options are ignored.

Options may be specified for only one M64 device at a time. Specifying options for multiple M64 devices requires multiple invocations of `m64config`.

Only M64-specific options can be specified through `m64config`. The normal window system options for specifying default depth, default visual class and so forth are still specified as device modifiers on the `openwin` command line. See the *OpenWindows Desktop Reference Manual* for details.

The user can also specify the OWconfig file that is to be updated. By default, the machine-specific file in the `/etc/openwin` directory tree is updated. The `-file` option can be used to specify an alternate file to use. For example, the system-global OWconfig file in the `/usr/openwin` directory tree can be updated instead.

Both of these standard OWconfig files can only be written by root. Consequently, the `m64config` program, which is owned by the root user, always runs with `setuid` root permission.

Options

`-defaults`
: Resets all option values to their default values.

`-depth 8 | 24 | 32`
: Sets the depth (bits per pixel) on the device. Possible values are `8`, `24`, or `32` (where `32` uses 24 bits per pixel). Log out of the current window system session and log back in for the change to take effect. `24` or `32` enables TrueColor graphics in the window system, at the expense of screen resolution.

The 32 setting enables simultaneous 8– and 24–bit color windows on m64 devices that support it. With setting 32, -propt shows depth 32 and -prconf shows depth 24. To check window depth, use the xwininfo utility. The xwininfo utility is usually shipped in the package containing frame buffer software (such as SUNWxwplt).

The maximum resolution that is available with 24 bits per pixel depends on the amount of memory installed on the PGX card. For 2-MB PGX cards, the maximum available resolution is 800x600. For 4-MB cards, it is 1152x900. For 8-MB cards, it is 1920x1080. If there is not enough memory for the specified combination of resolution and depth, m64config displays an error message and exits.

-dev *device-filename*
Specifies the M64 special file. If not specified, m64config will try /dev/fbs/m640 through /dev/fbs/m648 until one is found.

-file machine| system
Specifies which OWconfig file to update. If machine, the machine-specific OWconfig file in the /etc/openwin directory tree is used. If system, the global OWconfig file in the /usr/openwin directory tree is used. If the file does not exist, it is created. This option has no effect unless other options are specified. The default is *machine*.

-help
Prints a list of the m64config command line options, along with a brief explanation of each.

-prconf
Prints the M64 hardware configuration. The following is a typical display using the -prconf option:

```
--- Hardware Configuration for /dev/fbs/m640 ---
ASIC: version 0x41004754
DAC: version 0x0
PROM: version 0x0
Card possible resolutions: 640x480x60, 800x600x75, 1024x768x60
    1024x768x70, 1024x768x75, 1280x1024x75, 1280x1024x76
    1280x1024x60, 1152x900x66, 1152x900x76, 1280x1024x67
    960x680x112S, 960x680x108S, 640x480x60i, 768x575x50i
    1280x800x76, 1440x900x76, 1600x1000x66, 1600x1000x76
    vga, svga, 1152, 1280, stereo, ntsc, pal
Monitor possible resolutions: 720x400x70, 720x400x85, 640x480x60
    640x480x67, 640x480x72, 640x480x75, 800x600x56, 800x600x60
    800x600x72, 800x600x75, 832x624x75, 1024x768x85, 1024x768x60
    1024x768x70, 1024x768x75, 1280x1024x75, 1280x1024x76,
     1152x900x66, 1152x900x76, 1280x1024x67, 960x680x112S
    vga, svga, 1152, 1280, stereo
Possible depths: 8, 24
Current resolution setting: 1280x1024x76
Current depth: 8
```

`-propt`
Prints the current values of all M64 options in the OWconfig file specified by the `-file` option for the device specified by the `-dev` option. Prints the values of options as they will be in the OWconfig file after the call to `m64config` completes. The following is a typical display using the `-propt` option:

```
--- OpenWindows Configuration for /dev/fbs/m640 ---
OWconfig: machine
Video Mode: not set
Depth: 8
```

`-res` *video-mode* [`now` | `try` [`noconfirm` | `nocheck`]]
Specifies the video mode used to drive the monitor connected to the specified M64 device. Video modes are built-in. *video-mode* has the format of *width*x*height*x*rate*. *width* is the screen width in pixels, *height* is the screen height in pixels, and *rate* is the vertical frequency of the screen refresh. As a convenience, `-res` also accepts formats with `@` preceding the refresh rate instead of `x`. For example, `1280x1024@76`.

A list of valid video modes is obtained by issuing the following command: `m64config -res '?'`. Note that the `?` must be quoted. Not all resolutions are supported by both the video board and by the monitor. `m64config` will not permit you to set a resolution the board does not support, and will request confirmation before setting a resolution the monitor does not support.

`Symbolic names`

For convenience, some video modes have symbolic names defined for them. Instead of the form *width*x*height*x*rate*, one of these names may be supplied as the argument to `-res`. The meaning of the symbolic name `none` is that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

Name	Corresponding Video Mode
`svga`	1024x768x60
`1152`	1152x900x76
`1280`	1280x1024x76
`none`	(video mode currently programmed in device)

The `-res` option also accepts additional sub-options immediately following the video mode specification. Any or all of these may be present.

`nocheck` If present, the normal error checking based on the monitor sense code will be suspended. The video mode specified by the user will be accepted regardless of whether it is appropriate for the currently attached monitor.

This option is useful if a different monitor is to be connected to the M64 device. *Use of this option implies* `noconfirm` *as well.*

`noconfirm` Using the `-res` option, the user could potentially put the system into an unusable state, a state where there is no video output. This can happen if there is ambiguity in the monitor sense codes for the particular code read. To reduce the chance of this, the default behavior of `m64config` is to print a warning message to this effect and to prompt the user to find out if it is okay to continue. The noconfirm option instructs `m64config` to bypass this confirmation and to program the requested video mode anyway. This option is useful when `m64config` is being run from a shell script.

`now` If present, not only will the video mode be updated in the OWconfig file, but the M64 device will be immediately programmed to display this video mode. (This is useful for changing the video mode before starting the window system).

It is inadvisable to use this sub-option with `m64config` while the configured device is being used (for example, while running the window system); unpredictable results may occur. To run `m64config` with the `now` sub-option, first bring the window system down. If the `now` sub-option is used within a window system session, the video mode will be changed immediately, but the width and height of the affected screen won't change until the window system is exited and reentered again. Consequently, this usage is strongly discouraged.

`try` If present, the specified video mode will be programmed on a trial basis. The user will be asked to confirm the video mode by typing `y` within 10 seconds. Or the user may terminate the trial before 10 seconds are up by typing any character. Any character other than 'y' or carriage return is considered a no and the previous video mode will be restored and `m64config` will not change the video mode in the OWconfig file (other options specified will still take effect). If a carriage return is typed, the user is prompted for a yes or no answer on whether to keep the new video mode.

This sub-option should not be used with `m64config` while the configured device is being used (for example, while running the window system) as unpredictable results may occur. To run `m64config` with the `try` sub-option, the window system should be brought down first.

Defaults For a given invocation of `m64config` command line if an option does not appear on the command line, the corresponding OWconfig option is not updated; it retains its previous value.

When the window system is run, if an M64 option has never been specified by `m64config`, a default value is used. The option defaults are as follows:

Option	Default
`-dev`	`/dev/fbs/m640`
`-file`	machine
`-res`	none

The default for the `-res` option of none means that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

This provides compatibility for users who are used to specifying the device resolution through the PROM. On some devices (for example, GX) this is the only way of specifying the video mode. This means that the PROM ultimately determines the default M64 video mode.

Examples

EXAMPLE 1 Switching the Monitor Type

The following example switches the monitor type to the maximum resolution of `1280` x `1024` at `76` Hz:

```
example% /usr/sbin/m64config -res 1280x1024x76
```

Files

`/dev/fbs/m640`	device special file
`/etc/openwin/server/etc/OWconfig`	system config file (creates or updates the file)
`/usr/lib/fbconfig/SUNWm64_config`	symbolic link to `usr/sbin/m64config`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWm64cf

See Also

`attributes(5)`, `m64(7D)`

OpenWindows Desktop Reference Manual

Name mail.local – store mail in a mailbox

Synopsis `/usr/lib/mail.local [-f` *sender*`] [-d]` *recipient*

Description `mail.local` reads the standard input up to an end-of-file and appends it to each user's mail file (mailbox). This program is intended to be used by `sendmail(1M)` as a mail delivery agent for local mail. It is not a user interface agent.

Messages are appended to the user's mail file in the `/var/mail` directory. The user must be a valid user name.

Each delivered mail message in the mailbox is preceded by a "Unix From line" with the following format:

`From` *sender_address time_stamp*

The *sender_address* is extracted from the SMTP envelope address (the envelope address is specified with the `-f` option).

A trailing blank line is also added to the end of each message.

The mail files are locked with a `.lock` file while mail is appended.

The mail files are created with mode `660`, owner is set to *recipient*, and group is set to `mail`. If the "biff" service is returned by `getservbyname(3SOCKET)`, the biff server is notified of delivered mail. This program also computes the `Content-Length:` header which will be used by the mailbox reader to mark the message boundary.

Options The following options are supported:

`-f` *sender*	Specifies the "envelope from address" of the message. This flag is technically optional, but should be used.
`-d`	Specifies the recipient of the message. This flag is also optional and is supported here for backward compatibility. That is, `mail.local` *recipient* is the same as `mail.local -d` *recipient*.
`-l`	Turn on LMTP mode.
`-r` *from*	Specify the sender's name (for backward compatibility).
`-7`	Do not advertise 8BITMIME support in LMTP mode.
`-b`	Return a permanent error instead of a temporary error if a mailbox exceeds quota.

Operands The following operand is supported:

recipient	The recipient of the mail message.

Environment Variables

`TZ` Used to set the appropriate time zone on the timestamp.

Exit Status

The following exit values are returned:

`0` Successful operation.

`>0` An error occurred.

Files

`/tmp/local.`*XXXXXX* temporary files

`/tmp/lochd.`*XXXXXX* temporary files

`/var/mail/`*user_name* user's mail file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsndmu

See Also

`mail(1)`, `comsat(1M)`, `sendmail(1M)`, `getservbyname(3SOCKET)`, `attributes(5)`

Name makedbm – make a dbm file, or get a text file from a dbm file

Synopsis

```
makedbm [-b] [-l] [-s] [-E] [-i yp_input_file]
     [-o yp_output_name] [-d yp_domain_name]
     [-m yp_master_name] [-S delimiter]
     [-D number_of_delimiters] infile outfile

makedbm [-u dbmfilename]
```

Description The `makedbm` utility takes the *infile* and converts it to a pair of files in `ndbm` format (see `ndbm(3C)`), namely *outfile*`.pag` and *outfile*`.dir`. Each line of the input file is converted to a single `dbm` record. All characters up to the first TAB or SPACE form the key, and the rest of the line is the data. If a line ends with '\' (backslash), the data for that record is continued on to the next line. `makedbm` does not treat '#' (pound-sign) as a special character.

Because `makedbm` is mainly used in generating `dbm` files for the NIS name service, it generates a special entry with the key *yp_last_modified*, which is the date of *infile* (or the current time, if *infile* is '–'). The entries that have keys with the prefix *yp_* are interpreted by NIS server utilities.

Options The following options are supported:

`-b`	Insert the `YP_INTERDOMAIN` into the output. This key causes `ypserv(1M)` to use DNS for host name and address lookups for hosts not found in the maps.
`-d` *yp_domain_name*	Create a special entry with the key *yp_domain_name*.
`-D` *number_of delimiters*	Specify *number_of_delimiters* to skip before forming the key.
`-E`	Delimiters are escaped.
`-i` *yp_input_file*	Create a special entry with the key *yp_input_file*.
`-l`	Lower case. Convert the keys of the given map to lower case, so that, for example, host name matches succeed independent of upper or lower case distinctions.
`-m` *yp_master_name*	Create a special entry with the key *yp_master_name*. If no master host name is specified, *yp_master_name* is set to the local host name.
`-o` *yp_output_name*	Create a special entry with the key *yp_output_name*.
`-s`	Secure map. Accept connections from secure NIS networks only.
`-S` *delimiter*	Specify the *delimiter* to use instead of the default delimiter for forming the key.
`-u` *dbmfilename*	Undo a `dbm` file. Prints out the file in text format, one entry per line, with a single space separating keys from values.

Operands The following operands are supported:

infile Input file for `makedbm`. If *infile* is '–' (dash), the standard input is read.

outfile One of two output files in `ndbm` format: *outfile*`.pag` and *outfile*`.dir`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `ypserv(1M)`, `ndbm(3C)`, `attributes(5)`

Name makemap – create database maps for sendmail

Synopsis

```
makemap [-N] [-d] [-f] [-o] [-r] [-s] [-v] [-C file]
     [-c cachesize] [-D commentchar] [-e] [-l] [-t delim]
     [-u] mantype mapname
```

Description `makemap` creates the database maps used by the keyed map lookups in `sendmail(1M)`. `makemap` reads from the standard input and outputs to the specified *mapname*.

In all cases, `makemap` reads lines from the standard input consisting of two words separated by whitespace. The first is the database key, the second is the value. The value may contain `%`*n* strings to indicated parameter substitution. Literal percents should be doubled (`%%`). Blank lines and lines beginning with `#` are ignored.

`makemap` handles three different database formats. Database format is selected using the *maptype* parameter. See `OPERANDS`.

Options The following options are supported:

`-c` *cachesize*	Use the specified hash and B-Tree cache size (*cachesize*).
`-C` *file*	Use the specified sendmail configuration file (*file*) for looking up the TrustedUser option.
`-d`	Allow duplicate keys in the map. This is only allowed on B-Tree format maps. If two identical keys are read, both be inserted into the map.
`-D` *commentchar*	Use the specified character to indicate a comment (which is ignored) instead of the default of '`#`'.
`-e`	Allow empty value (right hand side).
`-f`	Normally, all upper case letters in the key are folded to lower case. This flag disables that behavior. This is intended to mesh with the `-f` flag in the `K` line in `sendmail.cf`. The value is never case folded.
`-l`	List supported map types.
`-N`	Include the null byte that terminates strings in the map. This must match the `-N` flag in the `K` line in `sendmail.cf`
`-o`	Append to an old file. This allows you to augment an existing file.
`-r`	Allow replacement of existing keys. Normally `makemap` complains if you repeat a key, and does not do the insert.
`-s`	Ignore safety checks on maps being created. This includes checking for hard or symbolic links in world writable directories.
`-t` *delim*	Use the specified delimiter (*delim*) instead of whitespace.

`-u` Dump (unmap) the content of the database to standard output. Note that, if the `-t` option is also provided, the specified delimiter is used when the content is dumped instead of whitespace.

`-v` Verbosely print what it is doing.

Operands The following operands are supported:

mapname File name of the database map being created.

maptype Specifies the database format. The following *maptype* parameters are available:

`dbm` Specifies DBM format maps.

`btree` Specifies B-Tree format maps.

`hash` Specifies hash format maps.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsndmu

See Also `editmap(1M)`, `sendmail(1M)`, `attributes(5)`

Name makeuuid – generate Universal Unique Identifiers

Synopsis

```
makeuuid [-e ether] [-n count] [-R root]
```

Description The `makeuuid` command generates UUIDs (Universal Unique Identifiers) conforming to the OSF DCE specification for UUIDs. The specification states:

"A UUID is an identifier that is unique across both space and time, with respect to the space of all UUIDs. A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network.

"The generation of UUIDs does not require a registration authority for each single identifier. Instead, it requires a unique value over space for each UUID generator. This spatially unique value is [normally] specified as an IEEE 802 address, which is usually already applied to network-connected systems."

The `makeuuid` command generates one or more UUIDs on the standard output.

Options The `makeuuid` command supports the following options:

`-e` *ether* Supplies an alternate address to be used in the generation of the UUIDs. Normally, the system's Ethernet address is acquired and used during the generation of a UUID. However, this requires root privileges to open and read the network devices. If this is not possible, you must supply an alternate Ethernet address.

`-n` *count* Generate multiple UUIDs. This option generates the specified number of UUIDs, one per line. Using this form is more efficient than, and functionally equivalent to, calling the `makeuuid` command multiple times. This can be used, for example, when a large number of UUIDs need to be generated for a given application.

`-R` *root* Use *root* as the root filesystem path when updating the shared state file (see FILES). The shared state file must be writable by the user running `makeuuid`, otherwise no UUIDs will be generated and the command will return in failure.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

Usage Normally, you run the `makeuuid` command with root privileges, as the Ethernet address and state files can be easily accessed and updated. If this is not possible, you must use the `-R` and `-e` options to specify an alternate root and Ethernet address to use when calculating the UUIDs.

Examples **EXAMPLE 1** Generating Multiple UUIDs

The following command generates 3000 UUIDs:

```
example# makeuuid -n 3000
```

EXAMPLE 2 Invoking Without Root Privileges

If you cannot obtain root privileges, you must specify an alternate Ethernet address and state file location:

```
example% makeuuid -e 11:22:33:44:55:66 -R /export/root/example2
```

See the caveat on the use of the `-R` option in the description of that option, above.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Out of memory.

`–1` Invalid Ethernet address given or access denied.

Files

`/var/sadm/system/uuid_state` UUID state file. Use of time values is one way that UUID generators, such as `makeuuid`, guarantee uniqueness. A state file is a mechanism that allows `makeuuid` to "remember" the last time value it used so it can increment that value for use in a new UUID. See the Internet Draft "UUIDs and GUIDs," dated February 4, 1998, for details on the state file mechanism.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwsr2

See Also `prodreg(1M)`, `Intro(3)`, `libwsreg(3LIB)`, `attributes(5)`

Notes The formal UUID specification is in the OSF DCE specification, available at `www.opengroup.org`. As of the date of publication of this man page, a copy of the specification is available at:

```
http://www.opengroup.org/onlinepubs/9629399/apdxa.htm
```

Sun has no control over the availability of documents on the `www.opengroup.org` web site.

Name masfcnv – SNMP configuration migration script

Synopsis

```
/usr/sfw/lib/sma_snmp/masfcnv [-cimnrs] [-l agentmaster]
     [-p enabledisableerror] [-t noneadd]
     [-u agentmastererror] [-y agentmastererror]

masfcnv [-V]

masfcnv [-?]
```

Description The `masfcnv` script is used to assist the system administrator in migrating an existing set of configuration files for the Sun SNMP Management Agent for Sun Fire and Netra Systems (MASF) to the Systems Management Agent (SMA).

The script accepts as input the currently installed set of MASF and SMA configuration files and outputs a new set of SMA configuration files. Existing SMA configuration files are backed up by appending `.bak` to the filename. The administrator can choose to output the new configuration to standard output, instead of replacing the current configuration, by specifying the `-n` option.

The migration script must be run as the superuser. Failure to do so causes the script to exit with an error message. Before running the script you should ensure that both the SMA and MASF agents are not running. If the agents are running they will be shut down by the script.

The migration script installs a new startup script for the MASF agent in `/etc/init.d`, as well as a backup of the old script. During migration, MASF will be configured as an AgentX subagent of SMA. All migration settings will be migrated to the SMA configuration file.

The migration script aborts if any unrecognized directives are found in either the MASF configuration files or the SMA configuration files. This can be overridden with the `-i` option. If this option is selected, the behavior is to retain unrecognized directives that were present in the SMA configuration, but remove those present in the MASF configuration.

The migration script then proceeds to migrate access control and trap configuration. As a side effect of running the migration script, the following directives might be expanded by the script into multiple directives with an equivalent interpretation:

- `rwcommunity`
- `rocommunity`
- `rwuser`
- `rouser`
- `trapcommunity`
- `trapsink`
- `trap2sink`
- `informsink`

Access Control Migration

Access control directives are expanded into the equivalent com2sec, group, access and view directives. Existing group names are renamed by prepending a prefix to avoid conflict with any which may already be defined in SMA.

When migrating SNMPv1 or v2c access control, a conflict can occur if both MASF and SMA configuration files have defined access permissions for the same community and source address. The default behavior is to abort with a message, unless a use of the `-y` option specifies otherwise. If `-y agent` is specified then the MASF configuration takes precedence. If `-y master` is specified then the SMA configuration is retained.

When migrating USM configuration (SNMPv3), a conflict can occur if both SMA and MASF configurations define a user with the same `securityName`. If this occurs, the behavior of the script is determined by the `-u` option. If `-u agent` is specified, the configuration of the user defined in the MASF configuration files is the one that is retained. Otherwise, if the `-u master` option is specified, the use defined in the SMA configuration files is retained.

By default, the migration script attempts to migrate USM users from MASF to SMA. The script determines whether there are any SNMPv3 users present in the SMA configuration and whether the default `engineID` has been overridden in the SMA configuration files. If neither of these conditions obtain, then the any `usmUser` statements containing localized authentication keys can be migrated to SMA, along with the MASF `engineID`. This results in the `engineID` of the SMA master agent changing.

If the script determines that there are existing SNMPv3 users or a manually configured `engineID` present in the SMA configuration, only those users defined in `createUser` statements are transferred. Those users that were defined in `usmUser` statements are transferred but will have their passwords reset to a random value. You should notify your users of their new password or reset the password yourself by editing the newly-generated configuration file.

Trap/Inform Migration

The migration script performs a check to determine whether a trap destination defined for MASF is already specified in an existing SMA `trapsink`, `trap2sink` or `informsink` directive. If this is the case, then the directive in the MASF configuration will be discarded to avoid duplicate traps/informs being received.

`trapsink`, `trap2sink` and `informsink` directives specified in the existing SMA configuration are considered valid destinations for MASF traps/informs and will receive them from the MASF subagent after migration.

If the `-t none` option was specified on the command line, the migration script carries over any remaining MASF trap/inform directives without modification.

If the `-t add` option was specified (the default), the migration script expands any `trapsink`, `trap2sink`, or `informsink` directives to use the `TARGET-MIB` and `NOTIFICATION-MIB`. The `TARGET-MIB` specifies targets using IP addresses, so it might be desirable to use the `-t none` option if, for example, the network allocates IP addresses to hostnames dynamically by means of DHCP.

The expanded directives defines filters specific to the MASF agent so that traps from other subagents will not be received by migrated trap destinations. Existing filters present in the SMA configuration are, by default, not modified and might or might not receive MASF traps, depending upon the filters that were originally defined for them.

If the `-l` option is specified, any filters already defined in the `TARGET-MIB` and the `NOTIFICATION-MIB` for SMA are extended to include traps from MASF. In the event that a trap destination is already configured in the `TARGET-MIB` with the same target address and community as an existing MASF trap/inform sink, a conflict will arise.

If `-l agent` was specified and a conflict arises, the migration script uses the target SNMP parameters (that is, the SNMP version and choice of trap/inform) defined by the MASF `trap/informsink` directive to send traps to this destination. Otherwise, if the `-l master` option was specified, the conflict will be resolved using the target SNMP parameters specified in the SMA configuration.

Miscellaneous

If the migration script encounters in the MASF configuration file any of the directives listed below and the directives are either not present or differ from the SMA configuration, the script will log a warning message.

- `syslocation`
- `syscontact`
- `sysname`
- `sysservices`
- `agentgroup`
- `agentuser`
- `authtrapenable`

Options

The following options are supported:

`-?`
`--help`

Displays usage information.

`-c`
`--no-community`

Do not transfer v1/v2c communities.

`-i`
`--ignore-unrecognized-directives`

Continue processing if unrecognized directives are present.

`-l agent | master`
`--master-trap-target=agent | master`

If `agent` is specified, the existing SMA trap targets will be configured to receive traps that were previously sent to destinations for the Sun Fire SNMP agent. If `master` is specified, the targets will be configured to receive Sun Fire SNMP traps, but existing SNMP target parameters will be used.

`-m`
`--no-usmuser`
: Do not transfer usm (v3) users.

`-n`
`--dry-run`
: Run the migration without modifying any files. If an error arises, continue processing. This can be used to determine the likely migration issues.

`-p enable | disable | error`
`--use-agent-port=enable | disable | error`
: Indicates whether the port originally used by the Sun Fire SNMP agent should be used by the SMA agent after migration (if the two agents are using different ports). If `enable` is specified, then the port used by the Sun Fire SNMP agent will also be used by the SMA agent after migration. If `disable` is specified, the ports used by SMA will not be updated by the migration tool. If the `error` option is specified and the SMA agent is not already using the same ports as those used by the original Sun Fire SNMP agent, an error is reported and the migration process is terminated. If no option is specified the default behavior is equivalent to the `error` flag.

`-r`
`--no-trap`
: Do not transfer trap destinations.

`-s`
`--skip-user`
: If a user is found in the MASF configuration file that cannot be created in the new configuration because of a change in the engine ID, then output a message indicating that the user could not be migrated (needs to be manually recreated) and continue processing. If this option is not present, the migration tool will consider such a situation as an error and abort.

`-t none | add`
`--trap-filter=none | add`
: If `none` is specified then the script will copy trap directives directly. The administrator might need to manually update the

configuration file to ensure traps are only delivered to their intended destinations. If `add` is specifed, trap filters will be constructed so that traps originating from the original Sun Fire SNMP agent are delivered only to the destinations that originally received them. The default behavior is `add`.

`-u agent | master | error`
`--select-user=agent | master | error`

Specifies that if a user with the same name is found in both configuration files that the conflict is to be resolved using the specified configuration file as input. Selecting a user from a particular will also cause the group declaration for that user to be taken from the same file. If `agent` is specified then the user will be taken from the configuration file for the Sun Fire SNMP Agent. If `master` is specified, the user will be taken from the SMA configuration. Otherwise, if `error` is given, the script will terminate. If this option is not present, the default behavior is equivalent to the `error` flag.

`-V`
`--version`

Display the version of this script.

`-y agent | master | error`
`--select-community=agent | master | error`

Examples

EXAMPLE 1 Simplest Case

The command shown below is appropriate for a simple migration. The migration fails if there are any potential conflicts.

```
# masfcnv
```

EXAMPLE 2 Migrating Such That MASF Settings Override

To migrate the MASF configuration such that it will always succeed, that MASF settings will override in the event of a conflict with SMA, and that access will still be provided on the original MASF port, enter:

```
# masfcnv -is -l agent -p enable -u agent -y agent
```

EXAMPLE 3 Dry Run, Retaining SMA Settings

To attempt a dry run and migrate the configuration such that any conflicts will be resolved by retaining existing SMA settings, enter:

```
masfcnv -l master -u master -y master
```

Exit Status

`0`	Success.
`non-zero`	A problem occurred during migration.

Files

`/etc/sma/snmp/snmpd.conf` `/var/sma_snmp/snmpd.conf`	SMA configuration files
`/etc/opt/SUNWmasf/conf/snmpd.conf` `/var/opt/SUNWmasf/snmpd.dat`	MASF configuration files
`/tmp/sma_migration.log`	`masfcnv` log file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	Stable

See Also

`attributes(5)`

Name mdlogd – Solaris Volume Manager daemon

Synopsis

```
mdlogd
```

Description `mdlogd` implements a simple daemon that watches the system console looking for messages written by the Solaris Volume Manger. When a Solaris Volume Manager message is detected, `mdlogd` sends a generic SNMP trap.

To enable traps, you must configure `mdlogd` into the SNMP framework. See *Solaris Volume Manager Administration Guide*.

Usage `mdlogd` implements the following SNMP MIB:

```
SOLARIS-VOLUME-MGR-MIB DEFINITIONS ::= BEGIN
        IMPORTS
                enterprises FROM RFC1155-SMI
                DisplayString FROM SNMPv2-TC;

        -- Sun Private MIB for Solaris Volume Manager


        sun       OBJECT IDENTIFIER ::= { enterprises 42 }
        sunSVM       OBJECT IDENTIFIER ::= { sun 104 }

        -- this is actually just the string from /dev/log that
        -- matches the md: regular expressions.
        -- This is an interim SNMP trap generator to provide
        -- information until a more complete version is available.

        -- this definition is a formalization of the old
        -- Solaris DiskSuite mdlogd trap mib.

        svmOldTrapString OBJECT-TYPE
                        SYNTAX DisplayString (SIZE (0..255))
                        ACCESS read-only
                        STATUS mandatory
                        DESCRIPTION
                        "This is the matched string that
                         was obtained from /dev/log."
        ::= { sunSVM 1 }

        -- SVM Compatibility ( error trap )

        svmNotice        TrapTRAP-TYPE
                        ENTERPRISE sunSVM
                        VARIABLES { svmOldTrapString }
                        DESCRIPTION
                                "SVM error log trap for NOTICE.
                                 This matches 'NOTICE: md:'"
```

```
        ::= 1

        svmWarningTrap  TRAP-TYPE
                        ENTERPRISE sunSVM
                        VARIABLES { svmOldTrapString }
                        DESCRIPTION
                                "SVM error log trap for WARNING..
                                 This matches 'WARNING: md:'"
        ::= 2

        svmPanicTrap    TRAP-TYPE
                        ENTERPRISE sunSVM
                        VARIABLES { svmOldTrapString }
                        DESCRIPTION
                                "SVM error log traps for PANIC..
                                This matches 'PANIC: md:'"
        ::= 3
END
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlvma, SUNWlvmr
Interface Stability	Obsolete

See Also snmpdx(1M), attributes(5)

Solaris Volume Manager Administration Guide

Name mdmonitord – daemon to monitor metadevices

Synopsis

```
/usr/sbin/mdmonitord [-t time_interval]
```

Description The `mdmonitord` utility is part of Solaris Volume Manager. It monitors and checks RAID1 (mirrors), RAID5 and hot spares.

There are two methods for checking:

- At fixed time intervals.
- When a RAID-1 (mirror), RAID-5, or hot spare fails. A failure generates an error event which triggers a check of these metadevices.

Options The following options are supported:

`-t` Time interval in seconds. The default value is 0, which causes probes to occur only upon an error. If you want to run `mdmonitord` at a regular interval, a value of 1800 (seconds, every half hour) is recommended as a starting point.

Exit Status The following exit values are returned:

`0` Successful completion.

>0 An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also `svcs(1)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `svcadm(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `smf(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Notes Since frequent probes can affect performance, it is recommended that the intervals between probes be limited.

The `mdmonitord` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/mdmonitor
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name medstat – check the status of mediator hosts for a given diskset

Synopsis

```
/usr/sbin/medstat [-q] -s setname
```

Description If a specified diskset has been configured for mediators, `medstat` attempts to contact these hosts to see if they are accessible and returns the results of the communication.

Options

`-q` This optional argument disables the printing of informative text. When used with `-q`, `medstat` still prints error messages and returns a result code.

`-s` *setname* Specifies the name of a diskset on which `medstat` will work.

Examples **EXAMPLE 1** Checking diskset

This example checks the mediator hosts for the selected diskset.

```
# medstat -s relo-red
```

The name of the diskset is `relo-red`. The `medstat` command prints the status for each mediator host. Additionally, if the mediator quorum is met, either through a "golden" mediator host or because half+1 of the mediator hosts respond, the exit code is 0. If the quorum is not met, then the exit code is 1. If no mediator hosts have been configured for the named diskset, the exit code is 2. The status field will contain one of the following values: Unreachable, Bad, Fatal, or Ok, where Unreachable indicates an RPC/communication problem, Bad indicates an error in the mediator data, Fatal indicates any other error condition, and Ok indicates no error conditions.

Files

`/etc/lvm/meddb` Contains the mediator data for a host that has been selected as a mediator host.

Exit Status The following exit values are returned:

`0` Successful completion.

>0 An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu
Interface Stability	Evolving

See Also `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.cf(4)`, `md.tab(4)`, `mddb.cf(4)`, `meddb(4)`, `mediator(7D)`

Sun Cluster documentation, *Solaris Volume Manager Administration Guide*

Notes This command is designed for use in the high availability product.

Name metaclear – delete active metadevices and hot spare pools

Synopsis

```
/usr/sbin/metaclear -h

/usr/sbin/metaclear [-s setname] -a [-f]

/usr/sbin/metaclear component

/usr/sbin/metaclear [-s setname] [-f] metadevice...
 hot_spare_pool...

/usr/sbin/metaclear [-s setname] -r [-f] metadevice...
 hot_spare_pool...

/usr/sbin/metaclear [-s setname] -p component

/usr/sbin/metaclear [-s setname] -p metadevice
```

Description The `metaclear` command deletes the specified metadevice or *hot_spare_pool.*, or purges all soft partitions from the designated component. Once a metadevice or hot spare pool is deleted, it must be re-created using `metainit` before it can be used again.

Any metadevice currently in use (open) cannot be deleted.

Options Root privileges are required for all of the following options except `-h`.

`-a`	Deletes all metadevices and configured hot spare pools in the set named by `-s`, or the local set by default.
`-f`	Deletes (forcibly) a metadevice that contains a subcomponent in an error state.
`-h`	Displays usage message.
`-p`	Deletes (purges) all soft partitions from the specified metadevice or component.
`-r`	Recursively deletes specified metadevices and hot spare pools, but does not delete metadevices on which others depend.
`-s` *setname*	Specifies the name of the diskset on which `metaclear` will work. Using the `-s` option causes the command to perform its administrative function within the specified diskset. Without this option, the command performs its function on local metadevices and/or hot spare pools.

Operands

metadevice ...	Specifies the name(s) of the metadevice(s) to be deleted.
component	Specifies the `c*d*t*s*` name(s) of the components containing soft partitions to be deleted.

hot_spare_pool ... Specifies the name(s) of the hot spare pools to be deleted in the form `hsp`*nnn*, where *nnn* is a number in the range 000-999.

Examples

EXAMPLE 1 Deleting Various Devices

The following example deletes a metadevice named `d10`.

```
# metaclear /dev/md/dsk/d10
```

The following example deletes all local metadevices and hot spare pools on the system.

```
# metaclear -a
```

The following example deletes a mirror, `d20`, with an submirror in an error state.

```
# metaclear -f d20
```

The following example deletes a hot spare pool, `hsp001`.

```
# metaclear hsp001
```

The following example deletes a soft partition, `d23`.

```
# metaclear d23
```

The following example purges all soft partitions on the slice `c2t3d5s2` if those partitions are not being used by other metadevices or are not open.

```
# metaclear -p c2t3d5s2
```

The following example purges soft partitions from a metadevice.

```
# metaclear -p d2
d3: Soft Partition is cleared
d4: Soft Partition is cleared
d5: Soft Partition is cleared
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also mdmonitord(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

Solaris Volume Manager Administration Guide

Name metadb – create and delete replicas of the metadevice state database

Synopsis

```
/sbin/metadb -h

/sbin/metadb [-s setname]

/sbin/metadb [-s setname] -a [-f] [-k system-file] mddbnn

/sbin/metadb [-s setname] -a [-f] [-k system-file]
     [-c number] [-l length] slice...

/sbin/metadb [-s setname] -d [-f] [-k system-file] mddbnn

/sbin/metadb [-s setname] -d [-f] [-k system-file] slice...

/sbin/metadb [-s setname] -i

/sbin/metadb [-s setname] -p [-k system-file]
     [mddb.cf-file]
```

Description The `metadb` command creates and deletes replicas of the metadevice state database. State database replicas can be created on dedicated slices, or on slices that will later become part of a simple metadevice (concatenation or stripe) or RAID5 metadevice. Do not place state database replicas on fabric-attached storage, SANs, or other storage that is not directly attached to the system and available at the same point in the boot process as traditional SCSI or IDE drives. See `NOTES`.

The metadevice state database contains the configuration of all metadevices and hot spare pools in the system. Additionally, the metadevice state database keeps track of the current state of metadevices and hot spare pools, and their components. Solaris Volume Manager automatically updates the metadevice state database when a configuration or state change occurs. A submirror failure is an example of a state change. Creating a new metadevice is an example of a configuration change.

The metadevice state database is actually a collection of multiple, replicated database copies. Each copy, referred to as a replica, is subject to strict consistency checking to ensure correctness.

Replicated databases have an inherent problem in determining which database has valid and correct data. To solve this problem, Volume Manager uses a *majority consensus algorithm*. This algorithm requires that a majority of the database replicas be available before any of them are declared valid. This algorithm strongly encourages the presence of at least three initial replicas, which you create. A consensus can then be reached as long as at least two of the three replicas are available. If there is only one replica and the system crashes, it is possible that all metadevice configuration data can be lost.

The majority consensus algorithm is conservative in the sense that it will fail if a majority consensus cannot be reached, even if one replica actually does contain the most up-to-date data. This approach guarantees that stale data will not be accidentally used, regardless of the failure scenario. The majority consensus algorithm accounts for the following: the system will

stay running with exactly half or more replicas; the system will panic when less than half the replicas are available; the system will not reboot without one more than half the total replicas.

When used with no options, the `metadb` command gives a short form of the status of the metadevice state database. Use `metadb -i` for an explanation of the flags field in the output.

The initial state database is created using the `metadb` command with both the `-a` and `-f` options, followed by the slice where the replica is to reside. The `-a` option specifies that a replica (in this case, the initial) state database should be created. The `-f` option forces the creation to occur, even though a state database does not exist. (The `-a` and `-f` options should be used together only when no state databases exist.)

Additional replicas beyond those initially created can be added to the system. They contain the same information as the existing replicas, and help to prevent the loss of the configuration information. Loss of the configuration makes operation of the metadevices impossible. To create additional replicas, use the `metadb -a` command, followed by the name of the new slice(s) where the replicas will reside. All replicas that are located on the same slice must be created at the same time.

To delete all replicas that are located on the same slice, the `metadb -d` command is used, followed by the slice name.

When used with the `-i` option, `metadb` displays the status of the metadevice state databases. The status can change if a hardware failure occurs or when state databases have been added or deleted.

To fix a replica in an error state, delete the replica and add it back again.

The metadevice state database (`mddb`) also contains a list of the replica locations for this set (local or shared diskset).

The local set `mddb` can also contain host and drive information for each of the shared disksets of which this node is a member. Other than the diskset host and drive information stored in the local set `mddb`, the local and shared diskset `mddb`s are functionality identical.

The `mddb`s are written to during the resync of a mirror or during a component failure or configuration change. A configuration change or failure can also occur on a single replica (removal of a `mddb` or a failed disk) and this causes the other replicas to be updated with this failure information.

Options Root privileges are required for all of the following options except `-h` and `-i`.

The following options can be used with the `metadb` command. Not all the options are compatible on the same command line. Refer to the `SYNOPSIS` to see the supported use of the options.

`-a`
Attach a new database device. The `/kernel/drv/md.conf` file is automatically updated with the new information and the `/etc/lvm/mddb.cf` file is updated as well. An alternate way to create replicas is by defining them in the `/etc/lvm/md.tab` file and specifying the assigned name at the command line in the form, `mddb`*nn*, where *nn* is a two-digit number given to the replica definitions. Refer to the `md.tab(4)` man page for instructions on setting up replicas in that file.

`-c` *number*
Specifies the number of replicas to be placed on each device. The default number of replicas is 1.

`-d`
Deletes all replicas that are located on the specified *slice*. The `/kernel/drv/md.conf` file is automatically updated with the new information and the `/etc/lvm/mddb.cf` file is updated as well.

`-f`
The `-f` option is used to create the initial state database. It is also used to force the deletion of replicas below the minimum of one. (The `-a` and `-f` options should be used together only when no state databases exist.)

`-h`
Displays a usage message.

`-i`
Inquire about the status of the replicas. The output of the `-i` option includes characters in front of the device name that represent the status of the state database. Explanations of the characters are displayed following the replica status and are as follows:

`d`
replica does not have an associated device ID.

`o`
replica active prior to last `mddb` configuration change

`u`
replica is up to date

`l`
locator for this replica was read successfully

`c`
replica's location was in `/etc/lvm/mddb.cf`

`p`
replica's location was patched in kernel

`m`
replica is master, this is replica selected as input

`r`
replica does not have device relocation information

`t`
tagged data is associated with the replica

`W`
replica has device write errors

`a`
replica is active, commits are occurring to this

`M`
replica had problem with master blocks

`D`
replica had problem with data blocks

`F`
replica had format problems

`S`
replica is too small to hold current database

`R`
replica had device read errors

`B`
tagged data associated with the replica is not valid

`-k` *system-file*
Specifies the name of the kernel file where the replica information should be written. The default *system-file* is `/kernel/drv/md.conf`. This option is for use with the local diskset only.

`-l` *length*
Specifies the size of each replica. The default *length* is 8192 blocks, which should be appropriate for most configurations. "Replica" sizes of less than 128 blocks are not recommended.

`-p`
Specifies updating the system file (`md.conf`) in the current working directory with entries from the `/etc/lvm/mddb.cf` file. This option is normally used to update a newly built system before it is booted for the first time. If the system has been built on a system other than the one where it will run, the location of the `mddb.cf` on the local machine can be passed as an argument. The system file to be updated can be changed using the `-k` option. This option is for use with the local diskset only.

`-s` *setname*
Specifies the name of the diskset on which the `metadb` command will work. Using the `-s` option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local database replicas.

slice
Specifies the logical name of the physical slice (partition), such as `/dev/dsk/c0t0d0s3`.

Examples

EXAMPLE 1 Creating Initial State Database Replicas

The following example creates the initial state database replicas on a new system.

```
# metadb -a -f c0t0d0s7 c0t1d0s3 c1t0d0s7 c1t1d0s3
```

The `-a` and `-f` options force the creation of the initial database and replicas. You could then create metadevices with these same slices, making efficient use of the system.

EXAMPLE 2 Adding Two Replicas on Two New Disks

This example shows how to add two replicas on two new disks that have been connected to a system currently running Volume Manager.

```
# metadb -a c0t2d0s3 c1t1d0s3
```

EXAMPLE 3 Deleting Two Replicas

This example shows how to delete two replicas from the system. Assume that replicas have been set up on `/dev/dsk/c0t2d0s3` and `/dev/dsk/c1t1d0s3`.

```
# metadb -d c0t2d0s3 c1t1d0s3
```

Although you can delete all replicas, you should never do so while metadevices still exist. Removing all replicas causes existing metadevices to become inoperable.

Files

`/etc/lvm/mddb.cf`
Contains the location of each copy of the metadevice state database.

`/etc/lvm/md.tab`
Workspace file for metadevice database configuration.

`/kernel/drv/md.conf`
Contains database replica information for all metadevices on a system. Also contains Solaris Volume Manager configuration information.

Exit Status

The following exit values are returned:

`0`
successful completion

`>0`
an error occurred

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	storage/svm
Interface Stability	Committed

See Also mdmonitord(1M), metaclear(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metaparam(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

Solaris Volume Manager Administration Guide

Notes Replicas cannot be stored on fabric-attached storage, SANs, or other storage that is not directly attached to the system. Replicas must be on storage that is available at the same point in the boot process as traditional SCSI or IDE drives. A replica can be stored on a:

- Dedicated local disk partition
- Local partition that will be part of a volume
- Local partition that will be part of a UFS logging device

Name metadevadm – update metadevice information

Synopsis

```
/usr/sbin/metadevadm  [-h] [-n] [ [-l]-r] [-s setname]
 [-u disk_specifier] [-v]
```

Description The `metadevadm` command facilitates the administration of device ID entries in Solaris Volume Manager. Use this command when the pathname stored in the metadevice state database no longer correctly addresses the device or when a disk drive has had its device ID changed.

This command requires root privileges.

Options The following options are supported.

`-h` Provide a help display.

`-l` Specify that `metadevadm` log to `syslog(3C)`. `metadevadm` logs to the the `DAEMON` facility at the `ERR` level by default. See `syslog.conf(4)` for additional information on changing logging levels.

Use this option anytime. It is most useful in startup scripts and less useful interactively.

`-n` Emulate the effect of a command, without making any changes to the system.

`-r` Recompute the pathname and disk specifier (including slice) associated with all devices in the metadevice state database if the device supports device IDs. If a device does not support device IDs or the device is not available, then no action is taken for that device.

Use this option when the disk has been moved or readdressed.

This option is run automatically at boot time to detect device ID changes and update the state database.

`-s` *setname* Specify the name of the disk set on which `metadevadm` works. This option causes the command to perform its administrative function within the specified disk set. Without this option, the command performs its function on devices in the local disk set.

`-u` *disk_specifier* Obtain the device ID associated with the *disk_specifier* (for example, `c1t2d0`) of a device and update the metadevice state database. If the device ID has not changed this option does nothing. Use this option when a disk drive has had its

device ID changed during a firmware upgrade or due to changing the controller of a storage subsystem.

`-v` Execute in verbose mode. This option has no effect when used with `-u`. Verbose is the default.

Examples

EXAMPLE 1 Updating Device ID of Disk

The following example updates the device `c2t3d0`:

```
# metadevadm -u c2t3d0
Updating SLVM device relocation information for c2t3d0.
Old device reloc information: id1928019239129312301201201001201209139 8
New device reloc information: id1938019239129312301201201001201209139 8
```

The following example is a variation of the preceding, using the full pathname.

```
# metadevadm -u /dev/dsk/c2t3d0
```

The following example uses the `-n` option, which means that the command is emulated, but does not take effect. Note that when the `-v` option is used with `-u`, `-v` has no effect (verbose is the default).

```
# metadevadm -u -v -n c2t3d0
Updating SLVM device relocation information for c2t3d0.
Old device reloc information: id1928019239129312301201201001201209139 8
New device reloc information: id1938019239129312301201201001201209139 8
```

EXAMPLE 2 Recomputing Pathnames

In the following example, all device names are valid.

```
# metadevadm -r
Disk movement detected.
Updating device names in SLVM.
```

In the following example, once again device names are valid.

```
# metadevadm -r -v
Disk movement detected.
Updating device names in SLVM.
c0t0d0s0 changed to c0t0d1s0 from device relocation information
id12098123lkmklsdjaasdkfjadfjakds
```

In the following example, `metadevadm` detects an invalid device name.

```
# metadevadm -r
Invalid device relocation information detected in SLVM.
Please check status of following disk(s):
c3t0d0
```

Return Values The following exit values are returned:

`0` Command was successful.

`1` `metadevadm` encountered an error condition.

`2` An invalid device ID was detected when using the `-r` option. This is for use in the `rc2.d` script. See `init.d(4)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also `mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Name metahs – manage hot spares and hot spare pools

Synopsis

```
/usr/sbin/metahs [-s setname] -a all component

/usr/sbin/metahs [-s setname] -a hot_spare_pool [component]

/usr/sbin/metahs [-s setname] -d hot_spare_pool [component]

/usr/sbin/metahs [-s setname] -d all component

/usr/sbin/metahs [-s setname] -e component

/usr/sbin/metahs [-s setname] -r hot_spare_pool component-old

/usr/sbin/metahs [-s setname] -r all component-old
 component-new

/usr/sbin/metahs [-s setname] -i [hot_spare_pool]...
```

Description The `metahs` command manages existing hot spares and hot spare pools. It is used to add, delete, enable, and replace components (slices) in hot spare pools. Like the `metainit` command, the `metahs` command can also create an initial hot spare pool. The `metahs` command does not replace a component of a metadevice. This function is performed by the `metareplace` command.

Hot spares are always in one of three states: available, in-use, or broken. Available hot spares are running and ready to accept data, but are not currently being written to or read from. In-use hot spares are currently being written to and read from. Broken hot spares are out of service and should be repaired. The status of hot spares is displayed when `metahs` is invoked with the `-i` option.

Solaris Volume Manager supports storage devices and logical volumes, including hot spares, greater than 1 terabyte (TB) when Solaris 10 is running a 64-bit kernel.

If a system with large volumes or hot spares is rebooted under a 32-bit Solaris 10 kernel, the large volumes are visible through `metastat` output, but they cannot be accessed, modified or deleted, and no new large volumes can be created. Any volumes or file systems on a large volume in this situation are also unavailable. If a system with large volumes is rebooted under a version of Solaris prior to Solaris 10, Solaris Volume Manager will not start. All large volumes must be removed before Solaris Volume Manager runs under another version of the Solaris Operating Environment.

Options Root privileges are required for any of the following options except `-i`.

The following options are supported:

`-a all` *component*
: Add *component* to all hot spare pools. `all` is not case sensitive.

`-a` *hot_spare_pool* [*component*]
: Add the *component* to the specified *hot_spare_pool*. *hot_spare_pool* is created if it does not already exist.

`-d all` *component*
Delete *component* from all the hot spare pools. The *component* cannot be deleted if it is in the in-use state.

`-d` *hot_spare_pool* [*component*]
Delete *hot_spare_pool*, if the *hot_spare_pool* is both empty and not referenced by a metadevice. If *component* is specified, it is deleted from the *hot_spare_pool*. Hot spares in the in-use state cannot be deleted.

`-e` *component*
Enable *component* to be available for use as a hot spare. The *component* can be enabled if it is in the broken state and has been repaired.

`-i` [*hot_spare_pool* `. . .`]
Display the status of the specified *hot_spare_pool* or for all hot spare pools if one is not specified.

`-r all` *component-old component-new*
Replace *component-old* with *component-new* in all hot spare pools which have the component associated. Components cannot be replaced from any hot spare pool if the old hot spare is in the in-use state.

`-r` *hot_spare_pool component-old component-new*
Replace *component-old* with *component-new* in the specified *hot_spare_pool*. Components cannot be replaced from a hot spare pool if the old hot spare is in the in-use state.

`-s` *setname*
Specify the name of the diskset on which `metahs` works. Using the `-s` option causes the command to perform its administrative function within the specified diskset. Without this option, the command performs its function on local hot spare pools.

Operands The following operands are supported:

component	The logical name for the physical slice (partition) on a disk drive, such as `/dev/dsk/c0t0d0s2`.
hot_spare_pool	Hot spare pools must be of the form `hsp`*nnn*, where *nnn* is a number in the range 000-999.

Examples **EXAMPLE 1** Adding a Hot Spare to a Hot Spare Pool

The following example adds a hot spare `/dev/dsk/c0t0d0s7` to a hot spare pool `hsp003`:

```
# metahs -a hsp003 c0t0d0s7
```

When the hot spare is added to the pool, the existing order of the hot spares already in the pool is preserved. The new hot spare is added at the end of the list of hot spares in the hot spare pool specified.

EXAMPLE 2 Adding a Hot Spare to All Currently Defined Pools

This example adds a hot spare to the hot spare pools that are currently defined:

```
# metahs -a all c0t0d0s7
```

The keyword `all` in this example specifies adding the hot spare, `/dev/dsk/c0t0d0s7`, to all the hot spare pools.

EXAMPLE 3 Deleting a Hot Spare

This example deletes a hot spare, `/dev/dsk/c0t0d0s7`, from a hot spare pool, `hsp003`:

```
# metahs -d hsp003 c0t0d0s7
```

When you delete a hot spare, the position of the remaining hot spares in the pool changes to reflect the new order. For instance, if in this example `/dev/dsk/c0t0d0s7` were the second of three hot spares, after deletion the third hot spare would move to the second position.

EXAMPLE 4 Replacing a Hot Spare

This example replaces a hot spare that was previously defined:

```
# metahs -r hsp001 c0t1d0s0 c0t3d0s0
```

In this example, the hot spare `/dev/dsk/c0t1d0s0` is replaced by `/dev/dsk/c0t3d0s0`. The order of the hot spares does not change.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Warnings Do not create large (>1 TB) volumes if you expect to run the Solaris Operating Environment with a 32-bit kernel or if you expect to use a version of the Solaris Operating Environment prior to Solaris 10.

REFERENCE

System Administration Commands - Part 2

Name metaimport – imports disk sets into existing Solaris Volume Manager configurations

Synopsis

```
metaimport  -s setname [-n] [-v] [-f] [disks]...

metaimport  -r [disks]...

metaimport  -V

metaimport  -?
```

Description The `metaimport` command allows the importing of disk sets, including replicated disk sets, into an existing Solaris Volume Manager configuration. Replicated disk sets are disk sets created using remote replication software.

The default Solaris Volume Manager configuration specifies a maximum number of disk sets that can be configured. The `metaimport` command fails if importing the disk set would result in exceeding the number of disk sets configured on the system. To increase the number of disk sets allowed on a system, see the *Solaris Volume Manager Administration Guide*.

Use `metaset(1M)` or `metastat(1M)` to view the configuration of the imported set.

You must run `metaimport` as root.

`metaimport` requires a functional Solaris Volume Manager configuration before it runs.

Options The following options are supported:

`-f` Force the import, even if a quorum of replicas from the imported disk set is not available. This option could result in corrupt configurations and should only be used when `metaimport` fails with the "Insufficient quorum detected; exiting" error. If only a partial disk set is available, this option might be necessary to successfully import. Some or all data could be corrupted or unavailable when importing a partial set or a set lacking a replica quorum.

`-n` Does not actually perform the operation, but shows the output or errors that would have resulted from the operation, had it been run.

`-r` Report on the non-configured disk sets found on the system. If no disk device or LUN is specified, `metaimport` reports on all non-configured disk sets attached to the system. When the name of one disk is specified, `metaimport` reports on the disk set (or virtual LUN) containing the specified disk. If two or more disks are specified, metaimport reports on the set (or sets, if they belong to different disk sets) containing the specified disks. If two or more disks are specified, `metaimport` reports on the set (or sets, if they belong to different disk sets) containing the specified disks.

`-s` *setname*	Specify the disk set name to use when importing. The imported disk set will be called *setname*, without regard to the name it may have had on a different system.
`-v`	Verbose. Provides detailed information about the `metadb` replica location and status.
`-V`	Version information.
`-?`	Display a help message.

Examples

EXAMPLE 1 Importing a Disk Set

The following example creates a disk set called `blue` and identifies `c1t5d0` as a disk containing a state database replica from the disk set being imported.

```
# metaimport -s blue c1t5d0
```

EXAMPLE 2 Reporting Disk Sets to Import

The following example scans all disks and LUNs attached to the system and configured as part of the system. It scans for disks that could be part of a disk set to be imported. Components that are already part of the Solaris Volume Manager configuration are ignored.

This use of `metaimport` provides suggested forms of the `metaimport` command to use to actually import the disk sets that have been found. You can specify a component on the command line to reduce the scope of the scan and generate results more quickly.

```
 # metaimport -r
```

Exit Status

`0`	Successful completion.
`>0`	An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu
Interface Stability	Stable

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `attributes(5)`

Solaris Volume Manager Administration Guide

Name metainit – configure metadevices

Synopsis

```
/sbin/metainit  -h

/sbin/metainit  [generic options] concat/stripe  numstripes
 width  component... [-i  interlace]

/sbin/metainit  [width  component... [-i  interlace]]
 [-h  hot_spare_pool]

/sbin/metainit  [generic options] mirror  -m  submirror
 [read_options] [write_options] [pass_num]

/sbin/metainit  [generic options] RAID  -r  component...
 [-i  interlace] [-h  hot_spare_pool] [-k] [-o  original_column_count]

/sbin/metainit  [generic options] hot_spare_pool
 [hotspare...]

/sbin/metainit  [generic options] metadevice-name

/sbin/metainit  [generic options] -a

/sbin/metainit  [generic options] softpart  -p [-e]
 component [-A  alignment] size

/sbin/metainit  -r
```

Description The `metainit` command configures metadevices and hot spares according to the information specified on the command line. Alternatively, you can run `metainit` so that it uses configuration entries you specify in the `/etc/lvm/md.tab` file (see `md.tab(4)`). All metadevices must be set up by the `metainit` command before they can be used.

Solaris Volume Manager supports storage devices and logical volumes greater than 1 terabyte (TB) when a system runs a 64-bit Solaris kernel. Support for large volumes is automatic. If a device greater than 1 TB is created, Solaris Volume Manager configures it appropriately and without user intervention.

If a system with large volumes is rebooted under a 32–bit Solaris kernel, the large volumes are visible through `metastat` output. Large volumes cannot be accessed, modified or deleted, and no new large volumes can be created. Any volumes or file systems on a large volume in this situation are unavailable. If a system with large volumes is rebooted under a version of Solaris prior to the Solaris 9 4/03 release, Solaris Volume Manager does not start. You must remove all large volumes before Solaris Volume Manager runs under an earlier version of the Solaris Operating System.

If you edit the `/etc/lvm/md.tab` file to configure metadevices, specify one complete configuration entry per line. You then run the `metainit` command with either the `-a` option, to activate all metadevices you entered in the `/etc/lvm/md.tab` file, or with the metadevice name corresponding to a specific configuration entry.

`metainit` does not maintain the state of the volumes that would have been created when `metainit` is run with both the `-a` and `-n` flags. Any volumes in `md.tab` that have dependencies on other volumes in `md.tab` are reported as errors when `metainit -a -n` is run, although the operations might succeed when `metainit -a` is run. See `md.tab(4)`.

Solaris Volume Manager never updates the `/etc/lvm/md.tab` file. Complete configuration information is stored in the metadevice state database, not `md.tab`. The only way information appears in `md.tab` is through editing it by hand.

When setting up a disk mirror, the *first* step is to use `metainit` create a one-on-one concatenation for the root slice. See `EXAMPLES`.

Options The following options are supported:

Generic Options Root privileges are required for all of the following options except `-h`.

The following generic options are supported:

`-f` Forces the `metainit` command to continue even if one of the slices contains a mounted file system or is being used as `swap`, or if the stripe being created is smaller in size than the underlying soft partition. This option is required when configuring mirrors on root (`/`), `swap`, and `/usr`.

`-h` Displays usage message.

`-n` Checks the syntax of your command line or `md.tab` entry without actually setting up the metadevice. If used with `-a`, all devices are checked but not initialized.

`-r` Only used in a shell script at boot time. Sets up all metadevices that were configured before the system crashed or was shut down. The information about previously configured metadevices is stored in the metadevice state database (see `metadb(1M)`).

`-s` *setname* Specifies the name of the diskset on which `metainit` works. Without the `-s` option, the `metainit` command operates on your local metadevices and/or hotspares.

Concat/Stripe Options The following concat/stripe options are supported:

concat/stripe Specifies the metadevice name of the concatenation, stripe, or concatenation of stripes being defined.

numstripes Specifies the number of individual stripes in the metadevice. For a simple stripe, *numstripes* is always 1. For a concatenation, *numstripes* is equal to the number of slices. For a concatenation of stripes, *numstripes* varies according to the number of stripes.

width Specifies the number of slices that make up a stripe. When *width* is greater than 1, the slices are striped.

component The logical name for the physical slice (partition) on a disk drive, such as `/dev/dsk/c0t0d0s0`. For RAID level 5 metadevices, a minimum of three slices is necessary to enable striping of the parity information across slices.

`-i` *interlace* Specifies the interlace size. This value tells Solaris Volume Manager how much data to place on a slice of a striped or RAID level 5 metadevice before moving on to the next slice. *interlace* is a specified value, followed by either 'k' for kilobytes, 'm' for megabytes, or 'b' for blocks. The characters can be either uppercase or lowercase. The *interlace* specified cannot be less than 16 blocks, or greater than 100 megabytes. If *interlace* is not specified, it defaults to 16 kilobytes.

`-h` *hot_spare_pool* Specifies the *hot_spare_pool* to be associated with the metadevice. If you use the command line, the hot spare pool must have been previously created by the `metainit` command before it can be associated with a metadevice. The *hot_spare_pool* must be of the form `hsp`*nnn*, where *nnn* is a number in the range 000-999. Use `/-h hsp`*nnn* when the concat/stripe being created is to be used as a submirror.

Mirror Options The following mirror options are supported:

mirror `-m` *submirror*

Specifies the metadevice name of the mirror. The `-m` indicates that the configuration is a mirror. *submirror* is a metadevice (stripe or concatentation) that makes up the initial one-way mirror. Solaris Volume Manager supports a maximum of four-way mirroring. When defining mirrors, first create the mirror with the `metainit` command as a one-way mirror. Then attach subsequent submirrors using the `metattach` command. This method ensures that Solaris Volume Manager properly syncs the mirrors. (The second and any subsequent submirrors are first created using the `metainit` command.)

read_options

The following read options for mirrors are supported:

`-g` Enables the geometric read option, which results in faster performance on sequential reads.

`-r` Directs all reads to the first submirror. This should only be used when the devices comprising the first submirror are substantially faster than those of the second mirror. This flag cannot be used with the `-g` flag.

If neither the `-g` nor `-r` flags are specified, reads are made in a round-robin order from all submirrors in the mirror. This enables load balancing across the submirrors.

write_options

The following write options for mirrors are supported:

`-S` Performs serial writes to mirrors. The first submirror write completes before the second is started. This can be useful if hardware is susceptible to partial sector failures. If `-S` is not specified, writes are replicated and dispatched to all mirrors simultaneously.

pass_num
A number in the range 0-9 at the end of an entry defining a mirror that determines the order in which that mirror is resynced during a reboot. The default is 1. Smaller pass numbers are resynced first. Equal pass numbers are run concurrently. If 0 is used, the resync is skipped. 0 should be used only for mirrors mounted as read-only, or as `swap`.

RAID Level 5 Options

The following RAID level 5 options are available:

RAID `-r`
Specifies the name of the RAID level 5 metadevice. The `-r` specifies that the configuration is RAID level 5.

`-k`
For RAID level 5 metadevices, informs the driver that it is not to initialize (zero the disk blocks) due to existing data. Only use this option to recreate a previously created RAID level 5 device.

Use the `-k` option with extreme caution. This option sets the disk blocks to the `OK` state. If any errors exist on disk blocks within the metadevice, Solaris Volume Manager might begin fabricating data. Instead of using the `-k` option, you might want to initialize the device and restore data from tape.

`-o` *original_column_count*
For RAID level 5 metadevices, used with the `-k` option to define the number of original slices in the event the originally defined metadevice was grown. This is necessary since the parity segments are not striped across concatenated devices.

Use the `-o` option with extreme caution. This option sets the disk blocks to the `OK` state. If any errors exist on disk blocks within the metadevice, Solaris Volume Manager might begin fabricating data. Instead of using the `-o` option, you might want to initialize the device and restore data from tape.

Soft Partition Options

The following soft partition options are supported:

softpart `-p` [`-e`] *component* [`-A` *alignment*] *size*
The *softpart* argument specifies the name of the soft partition. The `-p` specifies that the configuration is a soft partition.

The `-e` specifies that the entire disk specified by *component* as `c*t*d*` should be repartitioned and reserved for soft partitions. The specified component is repartitioned such that slice 7 reserves space for system (state database replica) usage and slice 0 contains all remaining space on the disk. Slice 7 is a minimum of 4MB, but can be larger, depending on the disk geometry. The newly created soft partition is placed on slice 0 of the device.

The *component* argument specifies the disk (`c*t*d*`), slice (`c*t*d*s*`), or meta device (`d*`) from which to create the soft partition. The *size* argument determines the space to use for the soft partition and can be specified in `K` or `k` for kilobytes, `M` or `m` for megabytes, `G` or `g` for gigabytes, `T` or `t` for terabyte (one terabyte is the maximum size), and `B` or `b` for blocks (sectors). All values represent powers of 2, and upper and lower case options are equivalent. Only integer values are permitted.

The `-A` alignment option sets the value of the soft partition extent alignment. This option used when it is important specify a starting offset for the soft partition. It preserves the data alignment between the metadevice address space and the address space of the underlying physical device. For example, a hardware device that does checksumming should not have its I/O requests divided by Solaris Volume Manager. In this case, use a value from the hardware configuration as the value for the alignment. When you use this option in conjunction with a software I/O load, the alignment value corresponds to the I/O load of the application. This prevents I/O from being divided unnecessarily and affecting performance.

The literal `all`, used instead of specifying size, specifies that the soft partition should occupy all available space on the device.

Hot Spare Pool Options The following hot spare pool options are supported:

hot_spare_pool [*hotspare...*]
When used as arguments to the `metainit` command, *hot_spare_pool* defines the name for a hot spare pool, and *hotspare...* is the logical name for the physical slice(s) for availability in that pool. *hot_spare_pool* is a number of the form `hsp`*nnn*, where *nnn* is a number in the range 000-999.

`md.tab` File Options The following `md.tab` file options are supported:

metadevice-name
When the `metainit` command is run with a *metadevice-name* as its only argument, it searches the `/etc/lvm/md.tab` file to find that name and its corresponding entry. The order in which entries appear in the `md.tab` file is unimportant. For example, consider the following `md.tab` entry:

```
d0 2 1 c1t0d0s0 1 c2t1d0s0
```

When you run the command `metainit d0`, it configures metadevice `d0` based on the configuration information found in the `md.tab` file.

`-a`
Activates all metadevices defined in the `md.tab` file.

`metainit` does not maintain the state of the volumes that would have been created when `metainit` is run with both the `-a` and `-n` flags. If a device `d0` is created in the first line of the `md.tab` file, and a later line in `md.tab` assumes the existence of `d0`, the later line fails when `metainit -an` runs (even if it would succeed with `metainit -a`).

Examples

EXAMPLE 1 Creating a One-on-One Concatenation

The following command creates a one-on-one concatenation for the root slice. Such a command is the first step you take when setting up a mirror for the root slice (and any other slice that cannot be unmounted). The `-f` option is required it create a volume with an existing file system, such as root(`/`).

```
# metainit -f d1 1 1 c0t0d0s0
```

The preceding command makes `d1` a one-on-one concatenation, using the root slice. You can then enter:

```
# metainit d0 -m d1
```

...to make a one-way mirror of the root slice.

EXAMPLE 2 Concatenation

All drives in the following examples have the same size of 525 Mbytes.

This example shows a metadevice, `/dev/md/dsk/d7`, consisting of a concatenation of four slices.

```
# metainit d7 4 1 c0t1d0s0 1 c0t2d0s0 1 c0t3d0s0 1 /dev/dsk/c0t4d0s0
```

The number 4 indicates there are four individual stripes in the concatenation. Each stripe is made of one slice, hence the number 1 appears in front of each slice. The first disk sector in all of these devices contains a disk label. To preserve the labels on devices `/dev/dsk/c0t2d0s0`, `/dev/dsk/c0t3d0s0`, and `/dev/dsk/c0t4d0s0`, the metadisk driver must skip at least the first sector of those disks when mapping accesses across the concatenation boundaries. Because skipping only the first sector would create an irregular disk geometry, the entire first cylinder of these disks is skipped. This allows higher level file system software to optimize block allocations correctly.

EXAMPLE 3 Stripe

This example shows a metadevice, `/dev/md/dsk/d15`, consisting of two slices.

```
# metainit d15 1 2 c0t1d0s0 c0t2d0s0 -i 32k
```

The number 1 indicates that one stripe is being created. Because the stripe is made of two slices, the number 2 follows next. The optional `-i` followed by `32k` specifies the interlace size as 32 Kbytes. If the interlace size were not specified, the stripe would use the default value of 16 Kbytes.

EXAMPLE 4 Concatentation of Stripes

This example shows a metadevice, `/dev/md/dsk/d75`, consisting of a concatenation of two stripes of three disks.

EXAMPLE 4 Concatentation of Stripes *(Continued)*

```
# metainit d75 2 3 c0t1d0s0 c0t2d0s0 \
      c0t3d0s0 -i 16k \
      3 c1t1d0s0 c1t2d0s0 c1t3d0s0 -i 32k
```

On the first line, the `-i` followed by 16k specifies that the stripe interlace size is 16 Kbytes. The second set specifies the stripe interlace size as 32 Kbytes. If the second set did not specify 32 Kbytes, the set would use the default interlace value of 16 Kbytes. The blocks of each set of three disks are interlaced across three disks.

EXAMPLE 5 Mirroring

This example shows a two-way mirror, `/dev/md/dsk/d50`, consisting of two submirrors. This mirror does not contain any existing data.

```
# metainit d51 1 1 c0t1d0s0
# metainit d52 1 1 c0t2d0s0
# metainit d50 -m d51
# metattach d50 d52
```

In this example, two submirrors, `d51` and `d52`, are created with the `metainit` command. These two submirrors are simple concatenations. Next, a one-way mirror, `d50`, is created using the `-m` option with `d51`. The second submirror is attached later using the `metattach` command. When creating a mirror, any combination of stripes and concatenations can be used. The default read and write options in this example are a round-robin read algorithm and parallel writes to all submirrors.

EXAMPLE 6 Creating a metadevice in a diskset

This example shows a metadevice, `/dev/md/dsk/d75`, consisting of a concatenation of two stripes within a diskset called `set1`.

```
# metainit -s set1 d75 2 3 c2t1d0s0 c2t2d0s0 \
      c2t3d0s0 -i 32k
# metainit -s set1 d51 1 1 c2t1d0s0
# metainit -s set1 d52 1 1 c3t1d0s0
# metainit -s set1 d50 -m d51
# metattach -s set1 d50 d52
```

In this example, a diskset is created using the `metaset` command. Metadevices are then created within the diskset using the `metainit` command. The two submirrors, `d51` and `d52`, are simple concatenations. Next, a one-way mirror, `d50`, is created using the `-m` option with `d51`. The second submirror is attached later using the `metattach` command. When creating a mirror, any combination of stripes and concatenations can be used. The default read and write options in this example are a round-robin read algorithm and parallel writes to all submirrors.

EXAMPLE 7 RAID Level 5

This example shows a RAID level 5 device, `d80`, consisting of three slices:

```
# metainit d80 -r c1t0d0s0 c1t1d0s0 c1t3d0s0 -i 20k
```

In this example, a RAID level 5 metadevice is defined using the `-r` option with an interlace size of 20 Kbytes. The data and parity segments are striped across the slices, `c1t0d0s0`, `c1t2d0s0`, and `c1t3d0s0`.

EXAMPLE 8 Soft Partition

The following example shows a soft partition device, `d1`, built on metadevice `d100` and 100 Mbytes (indicated by `100M`) in size:

```
# metainit d1 -p d100 100M
```

The preceding command creates a 100 Mbyte soft partition on the `d100` metadevice. This metadevice could be a RAID level 5, stripe, concatenation, or mirror.

EXAMPLE 9 Soft Partition on Full Disk

The following example shows a soft partition device, `d1`, built on disk `c3t4d0`:

```
# metainit d1 -p -e c3t4d0 9G
```

In this example, the disk is repartitioned and a soft partition is defined to occupy all 9 Gbytes of disk `c3t4d0s0`.

EXAMPLE 10 Soft Partition Taking All Available Space

The following example shows a soft partition device, `d1`, built on disk `c3t4d0`:

```
# metainit d1 -p -e c3t4d0 all
```

In this example, the disk is repartitioned and a soft partition is defined to occupy all available disk space on slice `c3t4d0s0`.

EXAMPLE 11 Hot Spare

This example shows a two-way mirror, `/dev/md/dsk/d10`, and a hot spare pool with three hot spare components. The mirror does not contain any existing data.

```
# metainit hsp001 c2t2d0s0 c3t2d0s0 c1t2d0s0
# metainit d41 1 1 c1t0d0s0 -h hsp001
# metainit d42 1 1 c3t0d0s0 -h hsp001
# metainit d40 -m d41
# metattach d40 d42
```

In this example, a hot spare pool, `hsp001`, is created with three slices from three different disks used as hot spares. Next, two submirrors are created, `d41` and `d42`. These are simple

EXAMPLE 11 Hot Spare *(Continued)*

concatenations. The `metainit` command uses the `-h` option to associate the hot spare pool `hsp001` with each submirror. A one-way mirror is then defined using the `-m` option. The second submirror is attached using the `metattach` command.

EXAMPLE 12 Setting the Value of the Soft Partition Extent Alignment

This example shows how to set the alignment of the soft partition to 1 megabyte.

```
# metainit -s red d13 -p c1t3d0s4 -A 1m 4m
```

In this example the soft partition, `d13`, is created with an extent alignment of 1 megabyte. The `metainit` command uses the `-A` option with an alignment of `1m` to define the soft partition extent alignment.

Files

`/etc/lvm/md.tab`
: Contains list of metadevice and hot spare configurations for batch-like creation.

Warnings

This section contains information on different types of warnings.

Devices and Volumes Greater Than 1 TB

Do not create large (>1 TB) volumes if you expect to run the Solaris Operating Environment with a 32–bit kernel or if you expect to use a version of the Solaris Operating Environment prior to Solaris 10.

Multi-Way Mirror

Do not use the `metainit` command to create a multi-way mirror. Rather, create a one-way mirror with `metainit` then attach additional submirrors with `metattach`. When the `metattach` command is not used, no resync operations occur and data could become corrupted.

If you use `metainit` to create a mirror with multiple submirrors, the following message is displayed:

```
WARNING: This form of metainit is not recommended.
The submirrors may not have the same data.
Please see ERRORS in metainit(1M) for additional information.
```

Truncation of Soft Partitions

When creating stripes on top of soft partitions it is possible for the size of the new stripe to be less than the size of the underlying soft partition. If this occurs, `metainit` fails with an error indicating the actions required to overcome the failure.

If you use the `-f` option to override this behavior, the following message is displayed:

```
WARNING: This form of metainit is not recommended.
The stripe is truncating the size of the underlying device.
Please see ERRORS in metainit(1M) for additional information.
```

Write-On-Write Problem

When mirroring data in Solaris Volume Manager, transfers from memory to the disks do not all occur at exactly the same time for all sides of the mirror. If the contents of buffers are changed while the data is in-flight to the disk (called write-on-write), then different data can end up being stored on each side of a mirror.

This problem can be addressed by making a private copy of the data for mirror writes, however, doing this copy is expensive. Another approach is to detect when memory has been modified across a write by looking at the dirty-bit associated with the memory page. Solaris Volume Manager uses this dirty-bit technique when it can. Unfortunately, this technique does not work for raw I/O or direct I/O. By default, Solaris Volume Manager is tuned for performance with the liability that mirrored data might be out of sync if an application does a "write-on-write" to buffers associated with raw I/O or direct I/O. Without mirroring, you were not guaranteed what data would actually end up on media, but multiple reads would return the same data. With mirroring, multiple reads can return different data. The following line can be added to `/etc/system` to cause a stable copy of the buffers to be used for all raw I/O and direct I/O write operations.

```
set md_mirror:md_mirror_wow_flg=0x20
```

Setting this flag degrades performance.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdr

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Limitations

Recursive mirroring is not allowed; that is, a mirror cannot appear in the definition of another mirror.

Recursive logging is not allowed; that is, a trans metadevice cannot appear in the definition of another metadevice.

Stripes, concatenations, and RAID level 5 metadevices must consist of slices only.

Mirroring of RAID level 5 metadevices is not allowed.

Soft partitions can be built on raw devices, or on stripes, RAID level 5, or mirrors.

RAID level 5 or stripe metadevices can be built directly on soft partitions.

Notes Trans metadevices have been replaced by UFS logging. Existing trans devices are *not* logging--they pass data directly through to the underlying device. See `mount_ufs(1M)` for more information about UFS logging.

Name metaoffline, metaonline – place submirrors offline and online

Synopsis

```
/usr/sbin/metaoffline  -h

/usr/sbin/metaoffline  [-s setname] [-f] mirror submirror

/usr/sbin/metaonline  -h

/usr/sbin/metaonline  [-s setname] mirror submirror
```

Description The `metaoffline` command prevents Solaris Volume Manager from reading and writing to the submirror that has been taken offline. While the submirror is offline, all writes to the mirror will be kept track of (by region) and will be written when the submirror is brought back online. The `metaoffline` command can also be used to perform online backups: one submirror is taken offline and backed up while the mirror remains accessible. (However, if this is a two-way mirror, data redundancy is lost while one submirror is offline.) The `metaoffline` command differs from the `metadetach` command because it does not sever the logical association between the submirror and the mirror. To completely remove a submirror from a mirror, use the `metadetach` command.

A submirror that has been taken offline will only remain offline until the `metaonline` command is invoked or the system is rebooted.

When the `metaonline` command is used, reading from and writing to the submirror resumes. A resync is automatically invoked to resync the regions written while the submirror was offline. Writes are directed to the submirror during resync. Reads, however, will come from a different submirror. Once the resync operation completes, reads and writes are performed on that submirror. The `metaonline` command is only effective on a submirror of a mirror that has been taken offline.

The `metaoffline` and `metaonline` commands can not be used on RAID 1 volumes in application-based recovery (ABR) mode.

A submirror that has been taken offline with the `metaoffline` command can only be mounted as read-only.

Options Root privileges are required for all of the following options except `-h`.

`-f`	Forces offlining of submirrors that have slices requiring maintenance.
`-h`	Displays usage message.
`-s` *setname*	Specifies the name of the diskset on which `metaoffline` and `metaonline` will work. Using the `-s` option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.
mirror	Specifies the metadevice name of the mirror from which the submirror will be either taken offline or put online.

submirror Specifies the metadevice name of the submirror to be either taken offline or put online.

Examples

EXAMPLE 1 Taking a Submirror Offline

This example takes one submirror, `d9`, offline from mirror `d10`.

```
# metaoffline d10 d9
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Notes

The `metaonline` and `metaoffline` commands are not applicable to mirrors in application-based recovery (ABR) mode.

Name metaparam – modify parameters of metadevices

Synopsis

```
/usr/sbin/metaparam  -h

/usr/sbin/metaparam  [-s setname] [concat/stripe or
 RAID5 options] concat/stripe RAID

/usr/sbin/metaparam  [-s setname] [mirror options]
 mirror
```

Description The `metaparam` command is used to display or modify current parameters of metadevices.

If just the metadevice is specified as an argument to the `metaparam` command, the current settings are displayed.

The `metaparam` command enables most metadevice (volume) parameters to be changed. Only the interlace value cannot be changed by `metaparam`, because it is established when the metadevice is created and cannot be changed thereafter.

Options Root privileges are required for all of the options.

The following options are supported:

`-h` Displays usage message.

`-s` *setname* Specify the name of the diskset on which `metaparam` works. Using the `-s` option causes the command to perform its administrative function within the specified diskset. Without this option, the command performs its function on local metadevices.

Concat/STRIPE Or Raid5 Options

–`h` *hot_spare_pool* | `none` Specifies the hot spare pool to be used by a metadevice. If `none` is specified, the metadevice is disassociated with the hot spare pool assigned to it. If the metadevice is currently using a hot spare, then `metaparam` cannot replace the hot spare pool.

concat/stripe | *RAID* Specifies the metadevice name of the concatenation, stripe, or concatenation of stripes, or of the RAID5 metadevice.

Mirror Options

–`r roundrobin | geometric | first` Modifies the read option for a mirror. The `-r` option must be followed by either `roundrobin`, `geometric`, or `first`. `roundrobin`, which is the default action under the `metainit` command, specifies reading the disks in a round-robin (load balancing) method. `geometric` allows for faster

	performance on sequential reads. `first` specifies reading only from the first submirror.
`–w parallel | serial`	Modifies the write option for a mirror. The `-w` option must be followed by either `parallel` or `serial`. `parallel`, the default action under the `metainit` command, specifies that all writes are parallel. `serial` specifies that all writes are serial.
`-p` *pass_number*	A number from 0-to-9 that specifies the order in which a mirror is resynced during reboot. The default is 1. Smaller pass numbers are resynced first. Equal pass numbers are run concurrently. If 0 is used, the mirror resync is skipped. 0 should only be used for mirrors mounted as read-only, or as `swap`.
mirror	Specifies the metadevice name of the mirror.

Examples

EXAMPLE 1 Associating Hot Spare Pool with RAID5 Metadevice

This example associates a hot spare pool, `hsp005`, with a RAID5 metadevice, `d80`.

```
# metaparam -h hsp005 d80
```

EXAMPLE 2 Changing Read Option to Geometric

This example changes the read option on a mirror `d50` from the default of roundrobin to geometric.

```
# metaparam -r geometric d50
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also mdmonitord(1M), metaclear(1M), metadb(1M), metadetach(1M), metahs(1M), metainit(1M), metaoffline(1M), metaonline(1M), metarecover(1M), metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

Solaris Volume Manager Administration Guide

Name metarecover – recover soft partition information

Synopsis

```
/sbin/metarecover [-n] [-v] [-s setname] component -p

/sbin/metarecover [-n] [-v] [-s setname] component -p
 {-d}

/sbin/metarecover [-n] [-v] [-s setname] component -p
 {-m}
```

Description The `metarecover` command scans a specified component to look for soft partition configuration information and to regenerate the configuration.

Options The following options are supported:

`-d` Recover soft partitions in the metadevice state database from the extent headers on the device. Options `-d` and `-m` are mutually exclusive.

`-m` Regenerate the extent headers and reapplies them to the underlying device based on the soft partitions listed in the metadevice state database. Options `-d` and `-m` are mutually exclusive.

`-n` Do not actually perform the operation. Show the output or errors that would have resulted from the operation, had it been run.

`-p` Regenerate soft partitions based on the metadevice state database or extent headers on the underlying device. If neither `-d` nor `-m` are specified, this option compares the soft partition information in the metadevice state database to the extent headers.

`-s` *setname* Specify the name of the diskset on which `metarecover` works. Using the `s` option causes the command to perform its function within the specified diskset. Without the `-s` option, the `metarecover` command operates on the metadevices and/or hot spare pools in the local diskset.

This option is required to recover former `sps` from a diskset component or raw-device. *setname* must be identical to the former *setname* in which the `sps` were created. The set numbers, however, seem irrelevant.

`-v` Verbose mode, displaying the changes being made.

Operands The following operand is supported:

component Specifies the `c*t*d*s*` number of the disk or slice containing the partitions, or the device name (for example, `d10`) of the metadevice containing the partitions.

component can be a slice name, component name, `/dev/dsk path`, or `/dev/rdsk path`.

Examples

EXAMPLE 1 Updating Metadevice State Database Based on Disk Extent Headers

A disk containing soft partitions is moved from one system to another. The system administrator would like to use the existing soft partitions. `metarecover` updates the metadevice state database based on the extent headers on the disk.

```
# metarecover -v c0t3d0s2 -p -d
```

EXAMPLE 2 Updating Metadevice State Database Based on Incomplete Soft Partition Creation

A system crashes in the middle of creating a new soft partition. The soft partition is in the creating state and the driver does not let that device be opened. `metarecover` rewrites the extent headers for the partially created soft partition and mark it as `Okay`.

```
# metarecover -v c0t3d0s2 -p -m
```

EXAMPLE 3 Updating Extent Headers Based on Metadevice State Database

Someone accidentally overwrote a portion of a disk leaving extent headers destroyed. `metarecover` rewrites the extent headers to ensure a valid soft partition configuration, though user data is not recovered.

```
# metarecover -v d5 -m
```

EXAMPLE 4 Validating Soft Partition Configuration

To validate the existing soft partition configuration, use `metarecover` with only the `-p` flag.

```
# metarecover c0t3d0s2 -p
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdr

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Name metarename – rename metadevice or switch layered metadevice names

Synopsis

```
/usr/sbin/metarename [-s setname] metadevice1 metadevice2

/usr/sbin/metarename [-s setname] [-f] -x metadevice1
 metadevice2

/usr/sbin/metarename -h
```

Description There are two ways to use `metarename`, one with and one without the `-x` option. The first method (without `-x`) renames an existing metadevice to a new name. This makes managing the metadevice namespace easier. The metadevice being renamed cannot be mounted or open, nor can the new name already exist. For example, to rename a metadevice that contains a mounted file system, you would first need to unmount the file system.

With the second way to use `metarename`, using the `-x` option, `metarename` switches (exchanges) the names of an existing layered metadevice and one of its subdevices. In Solaris Volume Manager terms, a layered metadevice can be either a mirror or a trans metadevice. The `-x` option enables you to switch the metadevice names of a mirror and one of its submirrors, or a trans metadevice and its master device.

`metarename -x` makes it easier to mirror or unmirror an existing stripe or concatenation, and to remove a trans device.

When used to mirror an existing stripe or concatenatation, you must stop access to the device. For example, if the device contains a mounted file system, you must first unmount the file system before doing the rename.

You can also use the `metarename -x` command to untrans a trans metadevice from an existing device. This applies only to the master device. You cannot remove a logging device with `metarename`. Before you can rename a trans device, you must detach the logging device. Then you must stop access to the trans metadevice itself.

You cannot rename or switch metadevices that are in an error state or that have subcomponents in an error state, or metadevices actively using a hot spare replacement.

You can only switch metadevices that have a direct child/parent relationship. You could not, for example, directly exchange a stripe in a mirror that is a master device with the trans metadevice.

You must use the `-f` flag when switching members of a trans metadevice.

Only metadevices can be switched, not slices.

Options The following options are supported:

`-f`
: Force the switching of trans metadevice members.

`-h`
: Display a help message.

`-s` *setname*
Specifies the name of the diskset on which `metarename` will work. Using the `-s` option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on the local metadevices.

`-x`
Exchange the metadevice names *metadevice1* and *metadevice2*.

metadevice1
Specifies the metadevice to be renamed or switched.

metadevice2
Specifies the target metadevice name for the rename or switch operation.

Examples

EXAMPLE 1 Renaming a Metadevice

This example renames a metadevice named `d10` to `d100`. Note that `d100` must not exist for the rename to succeed.

```
# metarename d10 d100
```

EXAMPLE 2 Creating a Two-Way Mirror

This example creates a two-way mirror from an existing stripe named `d1` with a mounted file system, `/home2`.

```
# metainit d2 1 1 c13d0s1
# metainit -f d20 -m d1
# umount /home2
# metarename -x d20 d1
# metattach d1 d2
# mount /home2
```

First, a second concatenation `d2`, is created. (`d1` already exists.) The `metainit` command creates a one-way mirror, `d20`, from `d1`. Next, you umount the file system and switch `d1` for `d20`, making `d1` the top-level device (mirror). You attach the second submirror, `d2`, to create a two-way mirror. Lastly, you remount the file system.

EXAMPLE 3 Mounting a Mirrored File System on Stripe

This example takes an existing mirror named `d1` with a mounted file system, and ends up with the file system mounted on a stripe `d1`.

```
# umount /fs2
# metarename -x d1 d20
# metadetach d20 d1
# metaclear -r d20
# mount /fs2
```

First, you unmount the file system, then switch the mirror `d1` and its submirror `d20`. This makes the mirror into `d20`. Next, you detach `d1` from `d20`, then delete the mirror `d20` and its

EXAMPLE 3 Mounting a Mirrored File System on Stripe *(Continued)*

other submirror. You then remount the file system.

EXAMPLE 4 Deleting a Trans Metadevice

This example deletes a trans metadevice named `d10` while its mount point is `/myhome`. The master device, which is a stripe, is named `d2`. The logging device, also a stripe, is named `d5`.

```
# umount /myhome
# metadetach d10
# metarename -f -x d10 d2
# metaclear d2
# metaclear d5
# fsck /dev/md/dsk/d10
# mount /myhome
```

You umount the file system first, then detach the trans metadevice's logging device. The trans metadevice is switched with the master device, making the trans metadevice `d2` and the underlying stripe `d10`. You clear the trans metadevice `d2` and the logging device `d5`. `d10` must be `fsck`'d, and then the file system is remounted.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Limitations

Renaming and exchanging metadevice names can only be used for metadevices. A physical slice cannot be renamed to a metadevice, nor can a metadevice be exchanged with a physical slice name.

Metadevice names are strings of the pattern *d<xyz>* where *xyz* is a value between `0` and `8192`. You cannot use logical names for metadevices.

Notes Trans metadevices have been replaced by UFS logging. Existing trans devices are *not* logging--they pass data directly through to the underlying device. See `mount_ufs(1M)` for more information about UFS logging.

Name metareplace – enable or replace components of submirrors or RAID5 metadevices

Synopsis

```
/usr/sbin/metareplace -h

/usr/sbin/metareplace [-s setname] -e mirror component

/usr/sbin/metareplace [-s setname] mirror component-old
 component-new

/usr/sbin/metareplace [-s setname] -e RAID component

/usr/sbin/metareplace [-s setname] [-f] RAID component-old
 component-new
```

Description The `metareplace` command is used to enable or replace components (slices) within a submirror or a RAID5 metadevice.

When you replace a component, the `metareplace` command automatically starts resyncing the new component with the rest of the metadevice. When the resync completes, the replaced component becomes readable and writable. If the failed component has been hot spare replaced, the hot spare is placed in the available state and made available for other hot spare replacements.

Note that the new component must be large enough to replace the old component.

A component may be in one of several states. The `Last Erred` and the `Maintenance` states require action. Always replace components in the `Maintenance` state first, followed by a resync and validation of data. After components requiring maintenance are fixed, validated, and resynced, components in the `Last Erred` state should be replaced. To avoid data loss, it is always best to back up all data before replacing `Last Erred` devices.

Options Root privileges are required for all of the following options except `-h`.

`-e` Transitions the state of *component* to the available state and resyncs the failed component. If the failed component has been hot spare replaced, the hot spare is placed in the available state and made available for other hot spare replacements. This command is useful when a component fails due to human error (for example, accidentally turning off a disk), or because the component was physically replaced. In this case, the replacement component must be partitioned to match the disk being replaced before running the `metareplace` command.

`-f` Forces the replacement of an errored component of a metadevice in which multiple components are in error. The component determined by the `metastat` display to be in the "Maintenance" state must be replaced first. This option may cause data to be fabricated since multiple components are in error.

`-h` Display help message.

`-s` *setname*	Specifies the name of the diskset on which `metareplace` will work. Using the `-s` option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.
mirror	The metadevice name of the mirror.
component	The logical name for the physical slice (partition) on a disk drive, such as `/dev/dsk/c0t0d0s2`.
component-old	The physical slice that is being replaced.
component-new	The physical slice that is replacing *component-old*.
RAID	The metadevice name of the RAID5 device.

Examples

EXAMPLE 1 Recovering from Error Condition in RAID5 Metadevice

This example shows how to recover when a single component in a RAID5 metadevice is errored.

```
# metareplace d10 c3t0d0s2 c5t0d0s2
```

In this example, a RAID5 metadevice `d10` has an errored component, `c3t0d0s2`, replaced by a new component, `c5t0d0s2`.

EXAMPLE 2 Use of `-e` After Physical Disk Replacement

This example shows the use of the `-e` option after a physical disk in a submirror (a submirror of mirror `d11`, in this case) has been replaced.

```
# metareplace -e d11 c1t4d0s2
```

Note: The replacement disk must be partitioned to match the disk it is replacing before running the `metareplace` command.

Exit Status

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`,

metarename(1M), metaroot(1M), metaset(1M), metassist(1M), metastat(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

Solaris Volume Manager Administration Guide

Name metaroot – setup system files for root (/) metadevice

Synopsis

```
/usr/sbin/metaroot  -h

/usr/sbin/metaroot  [-n] [-k system-name] [-v vfstab-name]
 [-c mddb.cf-name] [-m md.conf-name] [-R root-path] device
```

Description The `metaroot` command edits the `/etc/vfstab` and `/etc/system` files so that the system may be booted with the root file system (`/`) on an appropriate metadevice. The only metadevices that support the root file system are a stripe with only a single slice or a mirror on a single-slice stripe.

If necessary, the `metaroot` command can reset a system that has been configured to boot the root file system (`/`) on a metadevice so that it uses a physical slice.

Options Root privileges are required for all of the following options except `-h`.

The following options are supported:

`-c` *mddb.cf-name* — Use *mddb.cf-name* instead of the default `/etc/lvm/mddb.cf` file as a source of metadevice database locations.

`-h` — Display a usage message.

`-k` *system-name* — Edit a user-supplied *system-name* instead of the default `/etc/system` system configuration information file.

`-m` *md.conf-name* — Edit the configuration file specified by *md.conf-name* rather than the default, `/kernel/drv/md.conf`.

`-n` — Print what would be done without actually doing it.

`-R` *root-path* — When metaroot modifies system files, it accesses them in their relative location under *root-path*.

The `-R` option cannot be used in combination with the `-c`, `-k`, `-m`, or `-v` options.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-v` *vfstab-name* — Edit *vfstab-name* instead of the default `/etc/vfstab` table of file system defaults.

Operands The following operands are supported:

device — Specifies either the metadevice or the conventional disk device (slice) used for the root file system (`/`).

Examples

EXAMPLE 1 Specifying Root File System on Metadevice

The following command edits `/etc/system` and `/etc/vfstab` to specify that the root file system is now on metadevice `d0`.

```
# metaroot d0
```

EXAMPLE 2 Specifying Root File System on SCSI Disk

The following command edits `/etc/system` and `/etc/vfstab` to specify that the root file system is now on the SCSI disk device `/dev/dsk/c0t3d0s0`.

```
# metaroot /dev/dsk/c0t3d0s0
```

Files

`/etc/system`	System configuration information file. See `system(4)`.
`/etc/vfstab`	File system defaults.
`/etc/lvm/mddb.cf`	Metadevice state database locations.
`/kernel/drv/md.conf`	Configuration file for the metadevice driver, `md`.

Exit Status

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Name metaset – configure disk sets

Synopsis

```
/usr/sbin/metaset  -s setname [-M-a -h hostname]

/usr/sbin/metaset  -s setname -A{enable | disable}

/usr/sbin/metaset  -s setname [-A{enable | disable}]
 -a -h hostname...

/usr/sbin/metaset  -s setname -a [-l length] [-L] drivename...

/usr/sbin/metaset  -s setname -C {take | release |
 purge}

/usr/sbin/metaset  -s setname -d [-f] -h hostname...

/usr/sbin/metaset  -s setname -d [-f] drivename...

/usr/sbin/metaset  -s setname -j

/usr/sbin/metaset  -s setname -r

/usr/sbin/metaset  -s setname -w

/usr/sbin/metaset  -s setname -t [-f] [-u tagnumber]
 [y]

/usr/sbin/metaset  -s setname -b

/usr/sbin/metaset  -s setname -P

/usr/sbin/metaset  -s setname -q

/usr/sbin/metaset  -s setname -o [-h hostname]

/usr/sbin/metaset  [-s setname]

/usr/sbin/metaset  [-s setname] -a | -d [ [m] mediator_host_list]
```

Description The `metaset` command administers sets of disks in named disk sets. Named disk sets include any disk set that is not in the local set. While disk sets enable a high-availability configuration, Solaris Volume Manager itself does not actually provide a high-availability environment.

A single-owner disk set configuration manages storage on a SAN or fabric-attached storage, or provides namespace control and state database replica management for a specified set of disks.

In a shared disk set configuration, multiple hosts are physically connected to the same set of disks. When one host fails, another host has exclusive access to the disks. Each host can control a shared disk set, but only one host can control it at a time.

When you add a new disk to any disk set, Solaris Volume Manager checks the disk format. If necessary, it repartitions the disk to ensure that the disk has an appropriately configured reserved slice 7 (or slice 6 on an EFI labelled device) with adequate space for a state database replica. The precise size of slice 7 (or slice 6 on an EFI labelled device) depends on the disk geometry. For tradtional disk sets, the slice is no less than 4 Mbytes, and probably closer to 6

Mbytes, depending on where the cylinder boundaries lie. For multi-owner disk sets, the slice is a minimum of 256 Mbytes. The minimal size for slice 7 might change in the future. This change is based on a variety of factors, including the size of the state database replica and information to be stored in the state database replica.

For use in disk sets, disks must have a dedicated slice (six or seven) that meets specific criteria:

- The slice must start at sector 0
- The slice must include enough space for disk label
- The state database replicas cannot be mounted
- The slice does not overlap with any other slices, including slice 2

If the existing partition table does not meet these criteria, or if the -L flag is specified, Solaris Volume Manager repartitions the disk. A small portion of each drive is reserved in slice 7 (or slice 6 on an EFI labelled device) for use by Solaris Volume Manager. The remainder of the space on each drive is placed into slice 0. Any existing data on the disks is lost by repartitioning.

After you add a drive to a disk set, it can be repartitioned as necessary, with the exception that slice 7 (or slice 6 on an EFI labelled device) is not altered in any way.

After a disk set is created and metadevices are set up within the set, the metadevice name is in the following form:

`/dev/md/`*setname*`/{dsk,rdsk}/d`*number*

where *setname* is the name of the disk set, and *number* is the number of the metadevice (`0-127`).

If you have disk sets that you upgraded from Solstice DiskSuite software, the default state database replica size on those sets is 1034 blocks, not the 8192 block size from Solaris Volume Manager. Also, slice 7 on the disks that were added under Solstice DiskSuite are correspondingly smaller than slice 7 on disks that were added under Solaris Volume Manager.

If disks you add to a disk set have acceptable slice 7s (that start at cylinder 0 and that have sufficient space for the state database replica), they are not reformatted.

Hot spare pools within local disk sets use standard Solaris Volume Manager naming conventions. Hot spare pools with shared disk sets use the following convention:

setname`/hsp`*number*

where *setname* is the name of the disk set, and *number* is the number of the hot spare pool (0-999).

Multi–node Environment

To create and work with a disk set in a multi—node environment, `root` must be a member of Group 14 on all hosts, or the `/.rhosts` file must contain an entry for all other host names. This is not required in a SunCluster 3.x enviroment.

Tagged data

Tagged data occurs when there are different versions of a disk set's replicas. This tagged data consists of the set owner's nodename, the hardware serial number of the owner and the time it was written out to the available replicas. The system administer can use this information to determine which replica contains the correct data.

When a disk set is configured with an even number of storage enclosures and has replicas balanced across them evenly, it is possible that up to half of the replicas can be lost (for example, through a power failure of half of the storage enclosures). After the enclosure that went down is rebooted, half of the replicas are not recognized by SVM. When the set is retaken, the `metaset` command returns an error of "stale databases", and all of the metadevices are in a read-only state.

Some of the replicas that are not recognized need to be deleted. The action of deleting the replicas also causes updates to the replicas that are not being deleted. In a dual hosted disk set environment, the second node can access the deleted replicas instead of the existing replicas when it takes the set. This leads to the possibility of getting the wrong replica record on a disk set take. An error message is displayed, and user intervention is required.

Use the `-q` to query the disk set and the `-t`, `-u`, and `-y`, options to select the tag and take the disk set. See `OPTIONS`.

Mediator Configuration

SVM provides support for a low-end HA solution consisting of two hosts that share only two strings of drives. The hosts in this type of configuration, referred to as *mediators* or mediator hosts, run a special daemon, `rpc.metamedd(1M)`. The mediator hosts take on additional responsibilities to ensure that data is available in the case of host or drive failures.

A mediator configuration can survive the failure of a single host or a single string of drives, without administrative intervention. If both a host and a string of drives fail (multiple failures), the integrity of the data cannot be guaranteed. At this point, administrative intervention is required to make the data accessible. See `mediator(7D)` for further details.

Use the `-m` option to add or delete a mediator host. See `OPTIONS`.

Options

The following options are supported:

`-a`

Add drives or hosts to the named set. For a drive to be accepted into a set, the drive must not be in use within another metadevice or disk set, mounted on, or swapped on. When the drive is accepted into the set, it is repartitioned and the metadevice state database replica (for the set) can be placed on it. However, if a slice 7 (or slice 6 on an EFI labelled device), starts at cylinder 0, and is large enough to hold a state database replica, then the disk is not repartioned. Also, a drive is not accepted if it cannot be found on all hosts specified as part of the set. This means that if a host within the specified set is unreachable due to network problems, or is administratively down, the add fails.

`-a` | `-d` | `-m` *mediator_host_list*
Add (`-a`) or delete (`-d`) mediator hosts to the specified disk set. A *mediator_host_list* is the `nodename(4)` of the mediator host to be added and (for adding) up to two other aliases for the mediator host. The nodename and aliases for each mediator host are separated only by commas. Up to three mediator hosts can be specified for the named disk set. Specify only the nodename of that host as the argument to `-m` to delete a mediator host.

In a single `metaset` command you can add or delete three mediator hosts. See `EXAMPLES`.

`-A {enable | disable}`
Specify auto-take status for a disk set. If auto-take is enabled for a set, the disk set is automatically taken at boot, and file systems on volumes within the disk set can be mounted through `/etc/vfstab` entries. Only a single host can be associated with an auto-take set, so attempts to add a second host to an auto-take set or attempts to configure a disk set with multiple hosts as auto-take fails with an error message. Disabling auto-take status for a specific disk set causes the disk set to revert to normal behavior. That is, the disk set is potentially shared (non-concurrently) among hosts, and unavailable for mounting through `/etc/vfstab`.

`-b`
Insure that the replicas are distributed according to the replica layout algorithm. This can be invoked at any time, and does nothing if the replicas are correctly distributed. In cases where the user has used the `metadb` command to manually remove or add replicas, this command can be used to insure that the distribution of replicas matches the replica layout algorithm.

`-C {take | release | purge}`
Do not interact with the Cluster Framework when used in a Sun Cluster 3 environment. In effect, this means do not modify the Cluster Configuration Repository. These options should only be used to fix a broken disk set configuration. This option is not for use with a multi-owner disk set.

take
Take ownership of the disk set but do not inform the Cluster Framework that the disk set is available

release
Release ownership of the disk set without informing the Cluster Framework. This option should only be used if the disk set ownership was taken with the corresponding `-C take` option.

purge
Remove the disk set without informing the Cluster Framework that the disk set has been purged

`-d`

Delete drives or hosts from the named disk set. For a drive to be deleted, it must not be in use within the set. The last host cannot be deleted unless all of the drives within the set are deleted. Deleting the last host in a disk set destroys the disk set.

This option fails on a multi-owner disk set if attempting to withdraw the master node while other nodes are in the set.

`-f`

Force one of three actions to occur: takes ownership of a disk set when used with `-t`; deletes the last disk drive from the disk set; or deletes the last host from the disk set. Deleting the last drive or host from a disk set requires the `-d` option.

When used to forcibly take ownership of the disk set, this causes the disk set to be grabbed whether or not another host owns the set. All of the disks within the set are taken over (reserved) and fail fast is enabled, causing the other host to panic if it had disk set ownership. The metadevice state database is read in by the host performing the take, and the shared metadevices contained in the set are accessible.

You can use this option to delete the last drive in the disk set, because this drive would implicitly contain the last state database replica.

You can use `-f` option to delete hosts from a set. When specified with a partial list of hosts, it can be used for one-host administration. One-host administration could be useful when a host is known to be non-functional, thus avoiding timeouts and failed commands. When specified with a complete list of hosts, the set is completely deleted. It is generally specified with a complete list of hosts to clean up after one-host administration has been performed.

`-h` *hostname...*

Specify one or more host names to be added to or deleted from a disk set. Adding the first host creates the set. The last host cannot be deleted unless all of the drives within the set have been deleted. The host name is not accepted if all of the drives within the set cannot be found on the specified host. The host name is the same name found in `/etc/nodename`.

`-j`

Join a host to the owner list for a multi-owner disk set. The concepts of take and release, used with traditional disk sets, do not apply to multi-owner sets, because multiple owners are allowed.

As a host boots and is brought online, it must go through three configuration levels to be able to use a multi-owner disk set:

1. It must be included in the cluster nodelist, which happens automatically in a cluster or single-node sitatuion.
2. It must be added to the multi-owner disk set with the `-a -h` options documented elsewhere in this man page
3. It must join the set. When the host is first added to the set, it is automatically joined.

On manual restarts, the administrator must manually issue

`metaset -s` *multinodesetname* `-j`

to join the host to the owner list. After the cluster reconfiguration, when the host reenters the cluster, the node is automatically joined to the set. The `metaset -j` command joins the host to all multi-owner sets that the host has been added to. In a single node situation, joining the node to the disk set starts any necessary resynchronizations.

`-L`

When adding a disk to a disk set, force the disk to be repartitioned using the standard Solaris Volume Manager algorithm. See `DESCRIPTION`.

`-l` *length*

Set the size (in blocks) for the metadevice state database replica. The length can only be set when adding a new drive; it cannot be changed on an existing drive. The default (and maximum) size is 8192 blocks, which should be appropriate for most configurations. Replica sizes of less than 128 blocks are not recommended.

`-M`

Specify that the disk set to be created or modified is a multi-owner disk set that supports multiple concurrent owners.

This option is required when creating a multi-owner disk set. Its use is optional on all other operations on a multi-owner disk set and has no effect. Existing disk sets cannot be converted to multi-owner sets.

`-o`

Return an exit status of `0` if the local host or the host specified with the `-h` option is the owner of the disk set.

`-P`

Purge the named disk set from the node on which the `metaset` command is run. The disk set must not be owned by the node that runs this command. If the node does own the disk set, the command fails.

If you need to delete a disk set but cannot take ownership of the set, use the `-P` option.

This option is not for use with a multi-owner disk set.

`-q`

Displays an enumerated list of tags pertaining to ‘‘tagged data'' that can be encountered during a take of the ownership of a disk set.

This option is not for use with a multi-owner disk set.

`-r`

Release ownership of a disk set. All of the disks within the set are released. The metadevices set up within the set are no longer accessible.

This option is not for use with a multi-owner disk set.

`-s` *setname*
Specify the name of a disk set on which `metaset` works. If no *setname* is specified, all disk sets are returned.

`-t`
Take ownership of a disk set safely. If `metaset` finds that another host owns the set, this host is not be allowed to take ownership of the set. If the set is not owned by any other host, all the disks within the set are owned by the host on which `metaset` was executed. The metadevice state database is read in, and the shared metadevices contained in the set become accessible. The `-t` option takes a disk set that has stale databases. When the databases are stale, `metaset` exits with code 66, and prints a message. At that point, the only operations permitted are the addition and deletion of replicas. Once the addition or deletion of the replicas has been completed, the disk set should be released and retaken to gain full access to the data.

This option is not for use with a multi-owner disk set.

`-u` *tagnumber*
Once a tag has been selected, a subsequent take with `-u` *tagnumber* can be executed to select the data associated with the given *tagnumber*.

w
Withdraws a host from the owner list for a multi-owner disk set. The concepts of take and release, used with traditional disk sets, do not apply to multi-owner sets, because multiple owners are allowed.

Instead of releasing a set, a host can issue

```
metaset -s multinodesetname -w
```

to withdraw from the owner list. A host automatically withdraws on a reboot, but can be manually withdrawn if it should not be able to use the set, but should be able to rejoin at a later time. A host that withdrew due to a reboot can still appear joined from other hosts in the set until a reconfiguration cycle occurs.

`metaset -w` withdraws from ownership of all multi-owner sets of which the host is a member. This option fails if you attempt to withdraw the master node while other nodes are in the disk set owner list. This option cancels all resyncs running on the node. A cluster reconfiguration process that is removing a node from the cluster membership list effectively withdraws the host from the ownership list.

`-y`
Execute a subsequent take. If the take operation encounters “tagged data," the take operation exits with code 2. You can then run the `metaset` command with the `-q` option to see an enumerated list of tags.

Examples

EXAMPLE 1 Defining a Disk Set

This example defines a disk set.

```
# metaset -s relo-red -a -h red blue
```

The name of the disk set is `relo-red`. The names of the first and second hosts added to the set are `red` and `blue`, respectively. (The hostname is found in `/etc/nodename`.) Adding the first host creates the disk set. A disk set can be created with just one host, with the second added later. The last host cannot be deleted until all of the drives within the set have been deleted.

EXAMPLE 2 Adding Drives to a Disk Set

This example adds drives to a disk set.

```
# metaset -s relo-red -a c2t0d0 c2t1d0 c2t2d0 c2t3d0 c2t4d0 c2t5d0
```

The name of the previously created disk set is `relo-red`. The names of the drives are `c2t0d0`, `c2t1d0`, `c2t2d0`, `c2t3d0`, `c2t4d0`, and `c2t5d0`. There is no slice identifier ("s*x*") at the end of the drive names.

EXAMPLE 3 Adding Multiple Mediator Hosts

The following command adds three mediator hosts to the specified disk set.

```
# metaset -s mydiskset -a -m myhost1,alias1 myhost2,alias2 myhost3,alias3
```

EXAMPLE 4 Purging a Disk Set from the Node

The following command purges the disk set `relo-red` from the node:

```
# metaset -s relo-red -P
```

EXAMPLE 5 Querying a Disk Set for Tagged Data

The following command queries the disk set `relo-red` for a list of the tagged data:

```
# metaset -s relo-red -q
```

This command produces the following results:

```
The following tag(s) were found:
 1 - vha-1000c - Fri Sep 20 17:20:08 2002
 2 - vha-1000c - Mon Sep 23 11:01:27 2002
```

EXAMPLE 6 Selecting a tag and taking a Disk set

The following command selects a tag and takes the disk set `relo-red`:

```
# metaset -s relo-red -t -u 2
```

EXAMPLE 7 Defining a Multi-Owner Disk Set

The following command defines a multi-owner disk set:

```
# metaset -s blue -M -a -h hahost1 hahost2
```

The name of the disk set is `blue`. The names of the first and second hosts added to the set are `hahost1` and `hahost2`, respectively. The hostname is found in `/etc/nodename`. Adding the first host creates the multi-owner disk set. A disk set can be created with just one host, with additional hosts added later. The last host cannot be deleted until all of the drives within the set have been deleted.

Files

`/etc/lvm/md.tab` Contains list of metadevice configurations.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also

`mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Notes

Disk set administration, including the addition and deletion of hosts and drives, requires all hosts in the set to be accessible from the network.

Name metassist – automated volume creation utility to support Solaris Volume Manager

Synopsis

```
metassist -V

metassist -?

metassist create [-v n] [-c] -F config_file

metassist create [-v n] [-c | -d] -F request_file

metassist create [-v n] [-c | -d] [-f] [-n name] [-p datapaths]
 [-r redundancy] [-a available [,available,...]]
 [-u unavailable [,unavailable,...]] -s setname
 -S size

metassist create -?
```

Description The `metassist` command provides assistance, through automation, with common Solaris Volume Manager tasks.

SUBCOMMANDS The following subcommands are supported:

`create` The `create` subcommand creates one or more Solaris Volume Manager volumes. You can specify this request on the command line or in a file specified on the command line.

If you create a volume using the command line, you can specify the characteristics of the volume in terms of the desired quality of service it will provide - its size, the number of redundant copies of the data it contains, the number of data paths by which it is accessible, and whether faulty components are replaced automatically. The diskset in which the volume will reside and the volume's size must be specified on the command line in this form of the command.

If you create a volume using a request in a file, you can specify the characteristics of the volume in terms of the quality of service they provide, as on the command line. Alternatively, the file can specify the types and component parts of the volume, (for example, mirrors, stripes, concatenations, and their component slices). The file may also specify volumes partly in terms of their types and partly in terms of their component parts, and may specify the characteristics of more than one volume. All volumes specified in a file must reside in the same diskset, whose name must be specified in the file.

If you specify the `-c` or `-d` option on the command line, the command runs without creating an actual volume or volumes. Instead, it outputs either a a Bourne shell command script (`-c` option) or a volume configuration (`-d` option). The command script, when run, creates the

specified volume or volumes. The volume configuration specifies the volume or volumes in complete detail, naming all their components.

The input file given on the command line can take one of the following forms:

- a volume request, which specifies a request for a volume with explicit attributes and components, or matching a given quality of service
- a volume configuration, produced by a previous execution of the command

Options The following option is mandatory if you specify a volume request or volume configuration in a file:

`-F` *config_file* | *request_file*
Specify the volume request or volume configuration file to process. If *config_file* or *request_file* is `-`, it is read from standard input.

The `-d` option cannot be specified when *inputfile* is a volume configuration file.

The following options are mandatory if you specify a volume request on the command line:

`-s` *set*
Specify the disk set to use when creating volumes. All the volumes and hot spare pools are created in this disk set. If necessary, disks are moved into the diskset for use in the volumes and hot spare pools. If the diskset doesn't exist the command creates it. This option is required. `metassist` works entirely within a named disk set. Use of the local, or unnamed disk set, is not allowed.

`-S` *size*
Specify the size of the volume to be created. The size argument consists of a numeric value (a decimal can be specified) followed by KB, MB, GB, or TB, indicating kilobytes, megabytes, gigabytes, or terabytes, respectively. Case is ignored when interpreting this option. This option is required.

The following options are optional command line parameters:

`-a` *device1*`,`*device2*`, . . .`
Explicitly specify the devices that can be used in the creation of this volume. Named devices may be controllers or disks. Only used when specifying a volume on the command line.

`-c`
Output the command script that would implement the specified or generated volume configuration. The command script is not run, and processing stops at this stage.

`-d`
Output the volume configuration that satisfies the specified or generated volume request. No command script is generated or executed, and processing stops at this stage.

`-f`
Specify whether the volume should support automatic component replacement after a fault. If this option is specified, a mirror is created and its submirrors are associated with a hot spare.

`-n` *name*
Specify the name of the new volume. See `metainit(1M)` for naming guidelines.

`-p` *n*
Specify the number of required paths to the storage volume. The value of *n* cannot be greater than the number of different physical paths and logical paths to attached storage. Only used when specifying a volume on the command line.

`-r` *n*
Specify the redundancy level (`0`-`4`) of the data. The default is `0`. Only used when specifying a volume on the command line. If redundancy is `0`, a stripe is created. If redundancy is `1` or greater, a mirror with this number of submirrors is created. In this case, the volume can suffer a disk failure on *n*`-1` copies without data loss. With the use of hot spares (see the `-f` option), a volume can suffer a disk failure on *n*`+hsps-1` volumes without data loss, assuming non-concurrent failures.

`-u` *device1*`,`*device2*`,...`
Explicitly specify devices to exclude in the creation of this volume. Named devices can be controllers or disks. You can use this option alone, or to exclude some of the devices listed as available with the `-a` option, Only used when specifying a volume on the command line.

`-v` *value*
Specify the level of verbosity. Values from `0` to `2` are available, with higher numbers specifying more verbose output when the command is run. `-v 0` indicates silent output, except for errors or other critical messages. The default level is `1`.

`-V`
Display program version information.

`-?`
Display help information. This option can follow a subcommand for subcommand-specific help.

Examples

EXAMPLE 1 Creating a Mirror

The following example creates a two-way, 36Gb mirror on available devices from controller 1 and controller 2. It places the volume in diskset `mirrorset`.

```
# metassist create -r 2 -a c1,c2 -s mirrorset -S 36G
```

EXAMPLE 2 Creating a Mirror with Additional Fault Tolerance

The following example creates a two-way, 36Gb mirror on available devices from controller 1 and controller 2. It provides additional fault tolerance in the form of a hot spare. It places the volume in diskset `mirrorset`.

EXAMPLE 2 Creating a Mirror with Additional Fault Tolerance *(Continued)*

```
# metassist create -f -r 2 -a c1,c2 -s mirrorset -S 36GB
```

EXAMPLE 3 Creating a Three-way Mirror and Excluding Devices

The following example creates a three-way, 180Gb mirror from storage devices on controller 1 or controller 2. It excludes the disks `c1t2d0` and `c2t2d1` from the volume. It places the volume in diskset `mirrorset`.

```
metassist create -r 3 -a c1,c2 -u c1t2d0, c2t2d1 \
    -s mirrorset -S 180GB
```

EXAMPLE 4 Determining and Implementing a Configuration

The following example determines and implements a configuration satisfying the request specified in a request file:

```
# metassist create -F request.xml
```

EXAMPLE 5 Determining a Configuration and Saving It in a volume-config File

The following example determines a configuration which satisfies the given request. It saves the configuration in a volume-config file without implementing it:

```
# metassist create -d -F request.xml > volume-config
```

EXAMPLE 6 Determining a Configuration and Saving It in a Shell Script

The following example determines a configuration which satisfies the given request. It saves the configuration in a shell script without implementing it:

```
# metassist create -c -F request.xml > setupvols.sh
```

EXAMPLE 7 Implementing the Given volume-config

The following example implements the given volume-config:

```
# metassist create -F config.xml
```

EXAMPLE 8 Converting the Given volume-config to a Shell Script

The following example converts the given volume-config to a shell script that you can run later:

```
# metassist create -c -F config.xml > setupvols.sh
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/usr/share/lib/xml/dtd/volume-request.dtd`

`/usr/share/lib/xml/dtd/volume-defaults.dtd`

`/usr/share/lib/xml/dtd/volume-config.dtd`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdr

See Also `mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `volume-config(4)`, `volume-request(4)`, `attributes(5)`, `md(7D)`

Notes The quality of service arguments are mutually exclusive with the `-F` *inputfile* argument.

When specifying a request file or quality of service arguments on the command line, the `/etc/default/metassist.xml` file is read for global and per-disk set defaults.

Characteristics of this file are specified in the DTD, in `/usr/share/lib/xml/dtd/volume-defaults.dtd`.

Characteristics of the XML request file are specified in the DTD, in `/usr/share/lib/xml/dtd/volume-request.dtd`.

Characteristics of the XML configuration file are specified in the DTD, in `/usr/share/lib/xml/dtd/volume-config.dtd`.

This command must be run as root.

This command requires a functional Solaris Volume Manager configuration before it runs.

Name metastat – display status for metadevice or hot spare pool

Synopsis

```
/usr/sbin/metastat  -h

/usr/sbin/metastat  [-a] [-B] [-c] [-i] [-p] [-q] [-s setname]
 [-t] [metadevice...] [hot_spare_pool...]

/usr/sbin/metastat  [-a] [-B] [-c] [-i] [-p] [-q] [-s setname]
 component...
```

Description The `metastat` command displays the current status for each metadevice (including stripes, concatenations, concatenations of stripes, mirrors, RAID5, soft partitions, and trans devices) or hot spare pool, or of specified metadevices, components, or hot spare pools.

It is helpful to run the `metastat` command after using the `metattach` command to view the status of the metadevice.

`metastat` displays the state of each Solaris Volume Manager RAID-1 volume on the system. The possible states include:

`Okay`	The device reports no errors.
`Needs maintenance`	A problem has been detected. This requires that the system administrator replace the failed physical device. Volumes displaying `Needs maintenance` have incurred no data loss, although additional failures could risk data loss. Take action as quickly as possible.
`Last erred`	A problem has been detected. Data loss is a possibility. This might occur if a component of a submirror fails and is not replaced by a hot spare, therefore going into `Needs maintenance` state. If the corresponding component also fails, it would go into `Last erred` state and, as there is no remaining valid data source, data loss could be a possibility.
`Unavailable`	A device cannot be accessed, but has not incurred errors. This might occur if a physical device has been removed with Solaris Dynamic Reconfiguration (DR) features, thus leaving the Solaris Volume Manager volume unavailable. It could also occur if an array or disk is powered off at system initialization, or if a >1TB volume is present when the system is booted in 32-bit mode. After the storage has been made available, run the `metastat` command with the `-i` option to update the status of the metadevices. This clears the unavailable state for accessible devices.

See the *Solaris Volume Manager Administration Guide* for instructions on replacing disks and handling volumes in Needs maintenance or Last erred states.

Options The following options are supported:

`-a` Display all disk sets. Only metadevices in disk sets that are owned by the current host are displayed.

`-B` Display the current status of all of the 64-bit metadevices and hot spares.

`-c` Display concise output.

There is one line of output for each metadevice. The output shows the basic structure and the error status, if any, for each metadevice.

The `-c` output format is distinct from the `-p` output format. The `-p` option does not display metadevice status and is not intended as human-readable output.

`-h` Display usage message.

`-i` Check the status of RAID-1 (mirror) volumes, RAID-5 volumes, and hot spares. The inquiry checks each metadevice for accessibility, starting at the top level metadevice. When problems are discovered, the metadevice state databases are updated as if an error had occurred.

`-p` Display the list of active metadevices and hot spare pools in the same format as `md.tab`. See `md.tab(4)`.

The `-p` output is designed for snapshotting the configuration for later recovery or setup.

`-q` Display the status for metadevices without the device relocation information.

`-s` *setname* Specify the name of the disk set on which `metastat` works. Using the `-s` option causes the command to perform its administrative function within the specified disk set. Without this option, the command performs its function on metadevices and hot spare pools in the local disk set.

`-t` Display the current status and timestamp for the specified metadevices and hot spare pools. The timestamp provides the date and time of the last state change.

Operands The following operands are supported:

component — Display the status of the component hosting a soft partition, including extents, starting blocks, and block count.

hot_spare_pool — Display the status of the specified hot spare pool(s).

metadevice — Display the status of the specified metadevice(s). If a trans metadevice is specified, the status of the master and log devices is also displayed. Trans metadevices have been replaced by UFS logging. See `NOTES`.

Examples **EXAMPLE 1** Output Showing Mirror with Two Submirrors

The following example shows the partial output of the `metastat` command after creating a mirror, `d0`, consisting of two submirrors, `d70` and `d80`.

```
# metastat d0
d0: Mirror
    Submirror 0: d80
      State: Okay
    Submirror 1: d70
      State: Resyncing
    Resync in progress: 15 % done
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 2006130 blocks
    .
    .
    .
```

EXAMPLE 2 Soft Partition on Mirror with Submirror

The following example shows the partial output of the `metastat` command after creating a soft partition, `d3`, on concat `d2`, which is built on a soft partition.

```
# metastat
d2: Concat/Stripe
    Size: 204800 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Hot Spare
        d0                         0     No    Okay

d0: Soft Partition
    Component: c0t3d0s0
    Status: Okay
    Size: 204800 blocks
        Extent              Start Block  Block count
             0                     129       204800
```

EXAMPLE 2 Soft Partition on Mirror with Submirror *(Continued)*

```
d3: Soft Partition
    Component: d2
    Status: Okay
    Size: 202752 blocks
         Extent              Start Block  Block count
              0                     129       202752
```

EXAMPLE 3 Trans Metadevice

The following example shows the output of the `metastat` command after creating a trans metadevice.

```
# metastat
d2: Concat/Stripe
    Size: 204800 blocks
    Stripe 0:
        Device              Start Block  Dbase State        Hot Spare
        d0                         0     No    Okay

d0: Soft Partition
    Component: c0t3d0s0
    Status: Okay
    Size: 204800 blocks
        Extent              Start Block  Block count
             0                     129       204800

d3: Soft Partition
    Component: d2
    Status: Okay
    Size: 202752 blocks
         Extent              Start Block  Block count
              0                     129       202752
```

EXAMPLE 4 Multi-owner disk set

The following example shows the output of the `metastat` command with a multi-owner disk set and application-based mirror resynchronization option. Application-based resynchronization is set automatically if needed.

```
# metastat -s oban
oban/d100: Mirror
    Submirror 0: oban/d10
      State: Okay
    Submirror 1: oban/d11
      State: Okay
    Pass: 1
```

EXAMPLE 4 Multi-owner disk set *(Continued)*

```
    Read option: roundrobin (default)
    Write option: parallel (default)
    Resync option: application based
    Owner: None
    Size: 1027216 blocks (501 MB)

oban/d10: Submirror of oban/d100
    State: Okay
    Size: 1027216 blocks (501 MB)
    Stripe 0:
        Device     Start Block  Dbase     State Reloc Hot Spare
        c1t3d0s0          0     No         Okay

oban/d11: Submirror of oban/d100
    State: Okay
    Size: 1027216 blocks (501 MB)
    Stripe 0:
        Device     Start Block  Dbase     State Reloc Hot Spare
        c1t4d0s0          0     No         Okay
```

Warnings `metastat` displays states as of the time the command is entered. It is unwise to use the output of the `metastat -p` command to create a `md.tab(4)` file for a number of reasons:

- The output of `metastat -p` might show hot spares being used.
- It might show mirrors with multiple submirrors. See `metainit(1M)` for instructions for creating multi-way mirrors using `metainit` and `metattach`.
- A slice may go into an error state after `metastat -p` is issued.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdr
Stability	Evolving

See Also `mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`,

metarename(1M), metareplace(1M), metaroot(1M), metaset(1M), metassist(1M), metasync(1M), metattach(1M), md.tab(4), md.cf(4), mddb.cf(4), md.tab(4), attributes(5), md(7D)

Solaris Volume Manager Administration Guide

Notes Trans metadevices have been replaced by UFS logging. Existing trans devices are *not* logging--they pass data directly through to the underlying device. See mount_ufs(1M) for more information about UFS logging.

Name metasync – handle metadevice resync during reboot

Synopsis

```
/usr/sbin/metasync  -h

/usr/sbin/metasync  [-s setname] [buffer_size] metadevice

/usr/sbin/metasync  [-s setname] -r [buffer_size]

/usr/sbin/metasync  -p metadevice
```

Description The `metasync` command starts a resync operation on the specified *metadevice*. All components that need to be resynced are resynced. If the system crashes during a RAID5 initialization, or during a RAID5 resync, either an initialization or resync restarts when the system reboots.

Applications are free to access a metadevice at the same time that it is being resynced by `metasync`. Also, `metasync` performs the copy operations from inside the kernel, which makes the utility more efficient.

Use the `-r` option in boot scripts to resync all possible submirrors.

Options The following options are supported:

`-h` Displays usage message.

`-p` *metadevice* Regenerates parity information for RAID5 metadevices.

`-s` *setname* Specifies the name of the diskset on which `metasync` will work. Using the `-s` option will cause the command to perform its administrative function within the specified diskset. Without this option, the command will perform its function on local metadevices.

`-r` Specifies that the `metasync` command handle special resync requirements during a system reboot. `metasync -r` should only be invoked from the `svc:/system/mdmonitor` service. The `metasync` command only resyncs those metadevices that need to be resynced. `metasync` schedules all the mirror resyncs according to their pass numbers.

To override the default buffer_size value used by the `svc:/system/mdmonitor` service, you can edit `/etc/system` to specify:

```
set md_mirror:md_resync_bufsz = 2048
```

so that resyncs occur as quickly as possible.

Operands *buffer_size* Specifies the size (number of 512-byte disk blocks) of the internal copy buffer for the mirror resync. The size defaults to 128 512-byte disk blocks (64 Kbytes). It can be no more than 2048 blocks. For best performance (quickest completion of the resync), 2048 blocks is the recommended

size.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also `mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metadetach(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metattach(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Notes The `metasync` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/mdmonitor
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name metattach, metadetach – attach or detach a metadevice

Synopsis

```
/usr/sbin/metattach [-h]

/usr/sbin/metattach [-s setname] mirror [metadevice]

/usr/sbin/metattach [-s setname] [-i interlace] concat/stripe
 component...

/usr/sbin/metattach [-s setname] RAID component...

/usr/sbin/metattach [-s setname] [-A alignment] softpart
 size | all

/usr/sbin/metadetach [-s setname] [-f] mirror submirror

/usr/sbin/metadetach [-s setname] [-f] trans
```

Description `metattach` adds submirrors to a mirror, grows metadevices, or grows soft partitions. Growing metadevices can be done without interrupting service. To grow the size of a mirror or trans, the slices must be added to the submirrors or to the master devices.

Solaris Volume Manager supports storage devices and logical volumes greater than 1 terabyte (TB) when a system runs a 64-bit Solaris kernel. Support for large volumes is automatic. If a device greater than 1 TB is created, Solaris Volume Manager configures it appropriately and without user intervention.

If a system with large volumes is rebooted under a 32–bit Solaris kernel, the large volumes are visible through `metastat` output. Large volumes cannot be accessed, modified or deleted, and no new large volumes can be created. Any volumes or file systems on a large volume in this situation are also unavailable. If a system with large volumes is rebooted under a version of Solaris prior to the Solaris 9 4/03 release, Solaris Volume Manager does not start. You must remove all large volumes before Solaris Volume Manager runs under an earlier version of the Solaris Operating System.

Solaris Volume Manager supports one-to-four-way mirrors. You can only attach a metadevice to a mirror if there are three or fewer submirrors beneath the mirror. Once a new metadevice is attached to a mirror, `metattach` automatically starts a resync operation to the new submirror.

`metadetach` detaches submirrors from mirrors and logging devices from trans metadevices.

When a submirror is detached from a mirror, it is no longer part of the mirror, thus reads and writes to and from that metadevice by way of the mirror are no longer performed through the mirror. Detaching the only existing submirror is not allowed. Detaching a submirror that has slices reported as needing maintenance (by `metastat`) is not allowed unless the `-f` (force) flag is used.

`metadetach` also detaches the logging device from a trans. This step is necessary before you can clear the trans volume. Trans metadevices have been replaced by UFS logging. Existing

trans devices are not logging. They pass data directly through to the underlying device. See `mount_ufs(1M)` for more information about UFS logging.

Detaching the logging device from a busy trans device is not allowed unless the `-f` (force) flag is used. Even so, the logging device is not actually detached until the trans is idle. The trans is in the *Detaching* state (`metastat`) until the logging device is detached.

Options Root privileges are required for all of the following options except `-h`.

The following options are supported:

`-A` *alignment*
: Set the value of the soft partition extent alignment. Use this option when it is important specify a starting offset for the soft partition. It preserves the data alignment between the metadevice address space and the address space of the underlying physical device.

  For example, a hardware device that does checksumming should not have its I/O requests divided by Solaris Volume Manager. In this case, use a value from the hardware configuration as the value for the alignment. When using this option in conjunction with a software I/O load, the alignment value corresponds to the I/O load of the application. This prevents I/O from being divided unnecessarily and affecting performance.

`-f`
: Force the detaching of metadevices that have components that need maintenance or are busy. You can use this option only when a mirror is in a maintenance state that can be fixed with `metareplace(1M)`. If the mirror is in a maintenance state that can only be fixed with `metasync(1M)` (as shown by the output of `metastat(1M)`), `metadetach -f` has no effect, because the mirrors must be resynchronized before one of them can be detached.

`-h`
: Display a usage message.

`-i` *interlace*
: Specify the interlace value for stripes, where `size` is a specified value followed by either `k` for kilobytes, `m` for megabytes, or `b` for blocks. The units can be either uppercase or lowercase. If *size* is not specified, the size defaults to the interlace size of the last stripe of the metadevice. When an interlace size change is made on a stripe, it is carried forward on all stripes that follow.

`-s` *setname*
: Specify the name of the diskset on which the `metattach` command or the `metadetach` command works.. Using the `-s` option causes the command to perform its administrative function within the specified diskset. Without this option, the command performs its function on local metadevices.

Operands The following operands are supported:

component
The logical name for the physical slice (partition) on a disk drive, such as `/dev/dsk/c0t0d0s2`, being added to the concatenation, stripe, concatenation of stripes, or RAID5 metadevice.

concat/stripe
The metadevice name of the concatenation, stripe, or concatenation of stripes.

log
The metadevice name of the logging device to be attached to the trans metadevice.

metadevice
The metadevice name to be attached to the mirror as a submirror. This metadevice must have been previously created by the `metainit` command.

mirror
The name of the mirror.

RAID
The metadevice name of the RAID5 metadevice.

size | *all*
The amount of space to add to the soft partition in `K` or `k` for kilobytes, `M` or `m` for megabytes, `G` or `g` for gigabytes, `T` or `t` for terabytes, and `B` or `b` for blocks (sectors). All values represent powers of 2, and upper and lower case options are equivalent. Only integer values are permitted. The literal `all` specifies that the soft partition should grow to occupy all available space on the underlying volume.

softpart
The metadevice name of the existing soft partition.

submirror
The metadevice name of the submirror to be detached from the mirror.

trans
The metadevice name of the trans metadevice (not the master or logging device).

Examples

EXAMPLE 1 Concatenating a New Slice to a Metadevice

This example concatenates a single new slice to an existing metadevice, `d8`. Afterwards, you would use the `growfs(1M)` command to expand the file system.

```
# metattach d8 /dev/dsk/c0t1d0s2
```

EXAMPLE 2 Detaching Logging Device from Trans Metadevice

This example detaches the logging device from a trans metadevice `d9`. Notice that you do not have to specify the logging device itself, as there can only be one.

```
# metadetach d9
```

EXAMPLE 3 Expanding a RAID5 Metadevice

This example expands a RAID5 metadevice, `d45`, by attaching another slice.

```
# metattach d45 /dev/dsk/c3t0d0s2
```

When you add additional slices to a RAID5 metadevice, the additional space is devoted to data. No new parity blocks are allocated. The data on the added slices is, however, included in the overall parity calculations, so it is protected against single-device failure.

EXAMPLE 4 Expanding a Soft Partition

The following example expands a soft partition, `d42`, attaching all space available on the underlying device.

```
# metattach d42 all
```

When you add additional space to a soft partition, the additional space is taken from any available space on the slice and might not be contiguous with the existing soft partition.

EXAMPLE 5 Adding Space to Two-Way Mirror

This example adds space to a two-way mirror by adding a slice to each submirror. Afterwards, you would use the `growfs(1M)` command to expand the file system.

```
# metattach d9 /dev/dsk/c0t2d0s5
# metattach d10 /dev/dsk/c0t3d0s5
```

This example tells the mirror to grow to the size of the underlying devices

```
# metattach d11
```

This example increases the size of the UFS on the device so the space can be used.

```
# growfs -M /export /dev/md/dsk/d11
```

EXAMPLE 6 Detaching a Submirror from a Mirror

This example detaches a submirror, `d2`, from a mirror, `d4`.

```
# metadetach d4 d2
```

EXAMPLE 7 Adding Four Slices to Metadevice

This example adds four slices to an existing metadevice, `d9`. Afterwards, you would use the `growfs(1M)` command to expand the file system.

```
# metattach d9 /dev/dsk/c0t1d0s2 /dev/dsk/c0t2d0s2 \\
     /dev/dsk/c0t3d0s2 /dev/dsk/c0t4d0s2
```

EXAMPLE 8 Setting the Value of the Soft Partition Extent Alignment

This example shows how to set the alignment of the soft partition to 1mb when the soft partition is expanded.

EXAMPLE 8 Setting the Value of the Soft Partition Extent Alignment *(Continued)*

```
# metattach -s red -A 2m d13 1m
```

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also `mdmonitord(1M)`, `metaclear(1M)`, `metadb(1M)`, `metahs(1M)`, `metainit(1M)`, `metaoffline(1M)`, `metaonline(1M)`, `metaparam(1M)`, `metarecover(1M)`, `metarename(1M)`, `metareplace(1M)`, `metaroot(1M)`, `metaset(1M)`, `metassist(1M)`, `metastat(1M)`, `metasync(1M)`, `md.tab(4)`, `md.cf(4)`, `mddb.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

Solaris Volume Manager Administration Guide

Warnings This section provides information regarding warnings for devices greater than 1 TB and for multi-way mirrors.

Devices and Volumes Greater Than 1 TB

Do not create large (>1 TB) volumes if you expect to run the Solaris Operating System with a 32-bit kernel or if you expect to use a version of the Solaris Operating System prior to Solaris 9 4/03.

Multi-Way Mirrors

When a submirror is detached from its mirror, the data on the metadevice might not be the same as the data that existed on the mirror prior to running `metadetach`. In particular, if the `-f` option was needed, the metadevice and mirror probably do not contain the same data.

Notes Trans metadevices have been replaced by UFS logging. Existing trans devices are *not* logging. They pass data directly through to the underlying device. See `mount_ufs(1M)` for more information about UFS logging.

Name mib2c – produces template code from MIB definitions

Synopsis

```
/usr/sfw/bin/mib2c [-h] -c configfile [-f outname] [-i]
     [-q] [-S var=val] mibnode [mibnode]...
```

Description The `mib2c` tool is designed to take a portion of the MIB tree (as defined by a MIB file) and generate the template C code necessary to implement the corresponding MIB module.

To implement a new MIB module, three files are necessary:

- MIB definition file
- C header file
- C implementation file

The `mib2c` tool uses the MIB definition file to produce the two C code files. Thus, `mib2c` generates a template that you can edit to add logic necessary to obtain information from the operating system or application to complete the module.

The operand *mibnode* is the top level MIB node for which you want to generate code. You must give `mib2c` a MIB node (for example, `ifTable`), not a MIB file, on the command line. This distinction is a common source of user error.

The `mib2c` tool accepts both SMIv1 and SMIv2 MIBs.

`mib2c` needs to be able to find and load a MIB file in order to generate C code for the MIB. To enable `mib2c` to find the MIB file, set the `MIBS` environment variable to include the MIB file you are using. An example of setting this environment variable is:

```
MIBS=+NET-SNMP-TUTORIAL-MIB
```

or

```
MIBS=ALL
```

The first example ensures that `mib2c` finds the `NET-SNMP-TUTORIAL-MIB` MIB, in addition to the default MIB modules. The default list of MIB modules is set when the suite is first configured and built. The list corresponds to the list of modules that the agent supports. The second example ensures that `mib2c` finds all MIBs in the search location for MIB files. The default search location for MIB files is `DATADIR/snmp/mibs`. This search location can be modified by the `MIBDIRS` environment variable.

Both the MIB files to be loaded and the MIB file search location can also be configured in the `snmp.conf` file. Please see `snmp.conf(4)` for more information.

The generated `.c` and `.h` files are created in the current working directory.

Options The following options are supported:

`-h` Display a help message.

`-c` *configfile* Use *configfile* when generating code. These files are searched for first in the current directory and then in the `DATADIR` directory, which is where the default `mib2c` configuration files are located. Running `mib2c` without the `-c` *configfile* option displays a description of the valid values for *configfile*, that is, the available configuration files, including new ones that you might have created.

For example:

```
% mib2c ifTable
```

...displays the contents of the `mib2.conf` file, which displays hints on choosing the best *configfile* option for the *mibnode*.

The following values are supported for *configfile*:

```
mib2c.scalar.conf
mib2c.int_watch.conf
mib2c.iterate.conf
mib2c.create-dataset.conf
mib2c.array-user.conf
mib2c.column_defines.conf
mib2c.column_enums.conf
```

See EXAMPLES for commands you can use to generate code for scalar objects, tables, header files, and for SunOS 4.x code.

`-f` *outname* Places the output code into *outname*.c and *outname*.h. In most cases, `mib2c` places the output code into files with names that correspond to the group names for which it is generating code.

`-i` Do not run indent in the resulting code. Omitting this option results in `indent` error messages. These can safely be ignored. For example:

```
% /usr/sfw/bin/mib2c -c mib2c.scalar.conf ifTable
writing to ifTable.h
writing to ifTable.c
running indent on ifTable.h
indent: Command line: unknown parameter "-orig"
running indent on ifTable.c
indent: Command line: unknown parameter "-orig"
% ls
ifTable.c   ifTable.h
% rm i*
% /usr/sfw/bin/mib2c -c mib2c.scalar.conf -i ifTable
writing to ifTable.h
writing to ifTable.c
```

In the first invocation of `mib2c`, above, the `indent` errors are of no consequence.

`-q`	Run in "quiet" mode, which minimizes the status messages `mib2c` generates.
`-S` *var=val*	Preset a variable *var* in the `mib2c.*.conf` file to the value *val*. None of the existing `mib2c` configuration files (`mib2c.*.conf`) currently makes use of this feature. Consider this option available only for future use.

Examples

EXAMPLE 1 Generating Code for Scalar Objects

If you are writing code for some scalars, run:

```
% mib2c -c mib2c.scalar.conf mibnode
```

If you want to magically "tie" integer variables to integer scalars, use:

```
% mib2c -c mib2c.int_watch.conf mibnode
```

EXAMPLE 2 Generating Code for Tables

Consider the case where:

- You need to "iterate" over your table data to find the correct data for the SNMP row being accessed.
- Your table data is not kept within the agent (for example, it is in the kernel and not in the memory of the agent itself).

Under such conditions, use a command such as:

```
% mib2c -c mib2c.iterate.conf mibnode
```

You can find a similar example in `agent/mibgroup/mibII/vacm_context.c`.

If your table data is kept in the agent (that is, it is not located in an external source) and is purely data-driven (that is, you do not need to perform any work when a set occurs), you can use a command such as the following:

```
% mib2c -c mib2c.create-dataset.conf mibnode
```

See `agent/mibgroup/examples/data_set.c` for a similar example.

If your table data is kept in the agent (that is, it is not located in an external source) and you can keep your data sorted by the table index, but you do need to perform work when a set occurs, use a command such as the following:

```
% mib2c -c mib2c.array-user.conf mibnode
```

EXAMPLE 3 Generating Header File Definitions

To generate just a header with a define for each column number in your table, enter a command such as:

EXAMPLE 3 Generating Header File Definitions *(Continued)*

```
% mib2c -c mib2c.column_defines.conf mibnode
```

To generate only a header with a define for each enum for any column containing enums, enter:

```
% mib2c -c mib2c.column_enums.conf mibnode
```

EXAMPLE 4 Generating Code for the SunOS 4.X Line of Code

The following command generates code for SunOS 4.x:

```
% mib2c -c mib2c.old-api.conf mibnode
```

EXAMPLE 5 Generating Code for `ucdDemoPublic`

The command below generates C template code for the header and implementation files to implement `UCD-DEMO-MIB::ucdDemoPublic`.

```
% mib2c -c mib2c.scalar.conf ucdDemoPublic

writing to ucdDemoPublic.h
writing to ucdDemoPublic.c
running indent on ucdDemoPublic.h
running indent on ucdDemoPublic.c
```

The resulting `ucdDemoPublic.c` and `ucdDemoPublic.h` files are generated in the current working directory.

EXAMPLE 6 Generating Code for `tcpConnTable`

The command below generates C template code for the header and implementation files for the module to implement `TCP-MIB::tcpConnTable`.

```
% mib2c -c mib2c.iterate.conf tcpConnTable

writing to tcpConnTable.h
writing to tcpConnTable.c
running indent on tcpConnTable.h
running indent on tcpConnTable.c
```

The resulting `tcpConnTable.c` and `tcpConnTable.h` files are generated in the current working directory.

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also snmpcmd(1M), snmp.conf(4), attributes(5)

Name mib2mof – generate MOF file(s) from input SNMP MIB file(s)

Synopsis

```
/usr/sadm/bin/mib2mof [-n] [-d directory] [-q] [-c] [-a]
     [-h] files
```

Description The `mib2mof` utility reads input Management Information Base (MIB) files and produces one or more Managed Object Format (MOF) files. MOF files contain a Common Information Model (CIM) class declaration that represents the MIB for the Solaris Simple Network Management Protocol (SNMP) provider. The SNMP provider allows Web-Based Enterprise Management (WBEM) applications to access SNMP device information.

SNMP scalar variables map to properties in the CIM class. Qualifiers on each property convey the following MIB information for each scalar variable:

- syntax
- read/write access
- OID (Object IDentifier)
- description (optional)
- index (if the variable is within a group [sequence] that defines a row)

The syntax of an SNMP scalar variable is represented in a CIM class by the property's CIM datatype. All properties are marked with write access (true or false).

The following table shows how a Solaris SNMP datatype in a MIB maps to a Web-Based Enterprise Management (WBEM) CIM datatype and then to an SNMP datatype used by the WBEM SNMP API:

SNMP SMI Datatype	SNMP Ver.	CIM Datatype	SNMP API Object type
INTEGER	v1	sint32	SnmpInt
OCTET STRING	v1	string	SnmpString
OBJECT IDENTIFIER	v1	string	SnmpOid
IpAddress	v1	string	SnmpIpAddress
Counter	v1	uint32	SnmpCounter
Gauge	v1	uint32	SnmpGauge
TimeTicks	v1	uint32	SnmpTimeticks
Opaque	v1	sint8[]	SnmpOpaque
DisplayString - see OCTET STRING	v1		
NetworkAddress - see IpAddress	v1		
Counter32 - see Counter	v2		
Counter64	v2	uint64	SnmpCounter64
Integer32	v2	sint32	SnmpInt
Gauge32 - see Gauge	v2		
Unsigned32	v2	uint32	SnmpGauge
TruthValue	v2	sint32	SnmpInt
BITS - see OCTET STRING	v2		

The `mib2mof` utility includes its required `Solaris_SNMPmib_core.txt` file (containing core `MIB` definitions), installed in `/usr/sadm/mof`. The `mib2mof` utility looks first for `mib` core file in local directory. If this file is not found in the local directory, `mib2mof` looks in `/usr/sadm/mof`.

A MOF file is generated for each SNMP group and table row sequence (that is, the columns in one row) found in the supplied MIBs. (This does not include the core MIB definitions contained in the `Solaris_SNMPmib_core.txt file`.)

There is no MOF file or property for an SNMP table - all table access is through the rows and columns of the table, and the SNMP variable for the table is marked as inaccessible in the MIB.

The MOF file created contains a CIM class that represents an SNMP group or row and a CIM class to represent a CIM association. The output file name (and CIM class) is of the format <SNMP_><MIB name><Group name>.mof.

Options The following options are supported:

`-a` Generate MOF files for all of the input MIB files. If `-a` is not given, a MOF file is generated only for the last file of the input list.

`-c` Do not use the default `Solaris_SNMPmib_core.txt` definitions file shipped with the Solaris SNMP Provider for WBEM. If this option is specified, you must specify another `MIB_CORE` definitions file as one of the input files.

`-d` *directory* Generate output MOF files in the specified directory.

`-h` Show how to invoke `mib2mof` and list its arguments.

`-n` Parse the input MIB files without generating any output.

`-q` Include the DESCRIPTION clause of `SNMP OBJECT-TYPE` as a qualifier in the generated MOF file.

Operands The following operands are supported:

files List of SNMP MIB files to be converted.

Exit Status The `mib2mof` utility terminates with exit status `0`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbcou

See Also `init.wbem(1M)`, `mofcomp(1M)`, `wbemadmin(1M)`, `attributes(5)`

Name mibiisa – Sun SNMP Agent

Synopsis

```
mibiisa [-ar] [-c config-dir] [-d debug-level] [-p port]
     [-t cache-timer]
```

Description The `mibiisa` utility is an RFC 1157-compliant SNMP agent. It supports MIB-II as defined in *RFC 1213*, with Sun extensions under Sun's enterprise number. The MIB (Management Information Base) is both readable and writable. The `mibiisa` utility supports all SNMP protocol operations including GET-REQUEST, GETNEXT-REQUEST, SET-REQUEST, GET-REPLY, and TRAP.

The SMA (Systems Management Agent) is the default SNMP agent in Solaris. MIB-II subagent `mibiisa` does not run by default. To enable `mibiisa`, rename the configuration file from `/etc/snmp/conf/mibiisa.rsrc-` to `/etc/snmp/conf/mibiisa.rsrc`. SMA has the capability to handle any MIB-II requests. See `netsnmp(5)`.

The `mibiisa` utility supports the coldStart, linkUp, linkDown, and authentication traps. The authentication trap may be disabled by a command-line switch, which itself may be overridden by a management station writing to a MIB variable in the standard SNMP MIB group.

The `mibiisa` utility supports four distinct views of the MIB. The view used for any request is determined by the community string contained in that request.

To enhance security, `mibiisa` supports an option to block all writes to the MIB. You can also limit the set of management stations from which the agent will accept requests in the configuration file used when starting the `mibiisa`. See the Security section for more information.

Unless overridden, `mibiisa` uses UDP port 161, the standard SNMP port. The `mibiisa` utility issues traps through the same port on which it receives SNMP requests.

The `mibiisa` utility must run with super-user privileges and is typically started at system startup via `/etc/rc3.d`. `mibiisa` may not be started using `inetd(1M)`. When started, `mibiisa` detaches itself from the keyboard, disables all signals except `SIGKILL`, `SIGILL`, `SIGUSR1`, and `SIGUSR2`, and places itself in the background.

Options The following options are supported by `mibiisa`:

`-a` Disable the generation of authentication traps. However, an SNMP manager may write a value into `snmpEnableAuthenTraps` to enable or disable authentication traps.

`-c` *config-dir* Specify a directory where it expects `snmpd.conf` file, on startup. The default directory is `/etc/snmp/conf`.

`-d` *debug-level* Debug. A value of 0 disables all debug and is the default. Levels 1 through 3 represent increasing levels of debug output. When `mibiisa` receives the

signal `SIGUSR1`, it resets the debug-level to 0. When `mibiisa` receives the signal `SIGUSR2`, it increments the debug-level by one.

Debug output is sent to the standard output in effect at the time `mibiisa` is started. No matter what debug level is in effect, certain significant events are logged in the system log.

`-p` *port* — Define an alternative UDP port on which `mibiisa` listens for incoming requests. The default is UDP port 161.

`-r` — Place the MIB into read-only mode.

`-t` *cache-timer* — By default, information fetched from the kernel is considered to be valid for 45 seconds from the time it is retrieved. This cache lifetime may be altered with this parameter. You cannot set *cache-timer* to any value less than 1.

Configuration File

The `snmpd.conf` file is used for configuration information. Each entry in the file consists of a keyword followed by a parameter string. The keyword must begin in the first position. Parameters are separated from the keyword and from one another by white space. Case in keywords is ignored. Each entry must be contained on a single line. All text following (and including) a pound sign (#) is ignored. Keywords currently supported are:

`sysdescr`	The value to be used to answer queries for sysDescr.
`syscontact`	The value to be used to answer queries for sysContact.
`syslocation`	The value to be used to answer queries for sysLocation.
`trap`	The parameter names one or more hosts to receive traps. Only five hosts may be listed.
`system-group-read-community`	The community name to get read access to the system group and Sun's extended system group.
`system-group-write-community`	The community name to get write access to the system group and Sun's extended system group.
`read-community`	The community name to get read access to the entire MIB.
`write-community`	The community name to get write access to the entire MIB (implies read access).
`trap-community`	The community name to be used in traps.
`kernel-file`	The name of the file to use for kernel symbols.
`managers`	The names of hosts that may send SNMP queries. Only five hosts may be listed on any one line. This keyword may be repeated for a total of 32 hosts.

`newdevice` — The additional devices which are not built in SNMPD. The format is as follows: *newdevice* `type` *speed name* where *newdevice* is the keyword, `type` is an integer which has to match your schema file, *speed* is the new device's speed, and *name* is this new device's name.

An example `snmpd.conf` file is shown below:

```
sysdescr        Sun SNMP Agent, Sun Fire 4800, Company
                   Property Number 123456
syscontact      Cliff Claven
sysLocation     Room 1515, building 1
#
system-group-read-community     public
system-group-write-community    private
#
read-community  all_public
write-community all_private
#
trap            localhost
trap-community  SNMP-trap
#
#kernel-file    /vmunix
#
managers        lvs golden
managers        swap
```

Installation The `mibiisa` utility and its configuration file, `snmpd.conf`, may be placed in any directory. However for Solaris 2.4 and subseqent releases, use `/usr/lib/snmp` for `mibiisa` itself and `/etc/snmp/conf` for the configuration file. You can modify the configuration file as appropriate. If you make any changes to `snmpd.conf` file keyword values, you must kill and restart `mibiisa` for the changes to take effect.

Your `/etc/services` file (or NIS equivalent) should contain the following entries:

```
snmp                161/udp                             # Simple Network Mgmt
                                                        Protocol
snmp-trap           162/udp             snmptrap        # SNMP trap (event)
                                                        messages
```

The following is an example for Solaris 2.*x* and releases compatible with Solaris 2.*x*, such as Solaris 9:

```
#
# Start the SNMP agent
#
```

```
if [ -f /etc/snmp/conf/snmpd.conf -a -x
   /usr/lib/snmp/mibiisa ];
then
/opt/SUNWconn/snm/agents/snmpd
echo 'Starting SNMP-agent.'
```

Security SNMP, as presently defined, offers relatively little security. The `mibiisa` utility accepts requests from other machines, which can have the effect of disabling the network capabilities of your computer. To limit the risk, the configuration file lets you specify a list of up to 32 manager stations from which `mibiisa` will accept requests. If you do not specify any such manager stations, `mibiisa` accepts requests from anywhere.

The `mibiisa` utility also allows you to mark the MIB as "read-only" by using the `-r` option.

`mibiisa` supports four different community strings. These strings, however, are visible in the configuration file and within the SNMP packets as they flow on the network.

The configuration file should be owned by, and readable only by super-user. In other words the mode should be:

```
−rw−−−−−−−−  1 root         2090 Oct 17 15:04 /etc/snmp/conf/snmpd.conf
```

Managers can be restricted based on the community strings. This can be configured by creating an optional secondary configuration file `/etc/snmp/conf/mibiisa.acl`. To enable such a restriction, add the security line in the `/etc/snmp/conf/mibiisa.rsrc` file.

An example `mibiisa.acl` file is as follows:

```
acl = {
        {
                communities = public
                access = read-only
                managers = xyz
        }
        {
                communities = private
                access = read-write
                managers = abc,pqrs
        }
}
```

An example `mibiisa.rsrc` file is as follows:

```
resource =
{
        {
                registration_file = "/etc/snmp/conf/mibiisa.reg"
                security = "/etc/snmp/conf/mibiisa.acl"
                policy = "spawn"
                type = "legacy"
```

```
                command = "/usr/lib/snmp/mibiisa -r -p $PORT"
        }
}
```

Mib This section discusses some of the differences between the `mibiisa` MIB and the standard MIB-II (as defined in RFC 1213).

The following variables are read-only in the `mibiisa` MIB:

```
sysName
atIfIndex
ipDefaultTTL
```

These variables are read-write in the standard MIB-II.

The `mibiisa` MIB Address Translation tables support limited write access: only atPhysAddress may be written, either to change the physical address of an existing entry or to delete an entire ARP table entry.

The `mibiisa` MIB IP Net to Media table supports limited write access: only ipNetToMediaPhysAddress and ipNetToMediaType may be written, either to change the physical address of an existing entry or to delete an entire ARP table entry.

The following variables are read-write in the `mibiisa` MIB; however, these variables have fixed values. Any new values "set" to them are accepted, but have no effect:

```
ipRoutIfIndex
ipRouteMetric1
ipRouteMetric2
ipRouteMetric3
ipRouteMetric4
ipRouteType
ipRouteAge
ipRouteMask
ipRouteMetric5
```

The following `mibiisa` MIB variable reflects the actual state of the related table entry. "Sets" are accepted but have no effect:

```
tcpConnState
```

The following `mibiisa` MIB variables are readable, but return a fixed value:

`icmpInDestUnreachs`	Returns 1
`icmpInTimeExcds`	Returns 1
`icmpInParmProbs`	Returns 1
`icmpInSrcQuenchs`	Returns 1

icmpInRedirects	Returns 1
icmpInEchos	Returns 1
icmpInEchoReps	Returns 1
icmpInTimestamps	Returns 1
icmpInTimestampReps	Returns 1
icmpInAddrMasks	Returns 1
icmpInAddrMaskReps	Returns 1
icmpOutDestUnreachs	Returns 1
icmpOutTimeExcds	Returns 1
icmpOutParmProbs	Returns 1
icmpOutSrcQuenchs	Returns 1
icmpOutRedirects	Returns 1
icmpOutEchos	Returns 1
icmpOutEchoReps	Returns 1
icmpOutTimestamps	Returns 1
icmpOutTimestampReps	Returns 1
icmpOutAddrMasks	Returns 1
icmpOutAddrMaskReps	Returns 1
ifInUnknownProtos	Returns 0
ipAdEntBcastAddr	Returns 1
ipAdEntReasmMaxSiz	Returns 65535
ipRouteMetric1	Returns −1
ipRouteMetric2	Returns −1
ipRouteMetric3	Returns −1
ipRouteMetric4	Returns −1
ipRouteAge	Returns 0
ipRouteMetric5	Returns −1
ipNetToMediaType	Returns (3) dynamic
ipRoutingDiscards	Returns 0

The following variables return a fixed value of `0` for drivers not conforming to the GLD framework (see `gld(7D)`), including the old LAN drivers on SPARC machines:

`ifInOctets`	Returns 0
`ifInNUcastPkts`	Returns 0
`ifInDiscards`	Returns 0
`ifOutOctets`	Returns 0
`ifOutNUcastPkts`	Returns 0
`ifOutDiscards`	Returns 0

Schema Attributes

The following describes the attributes in the group and table definitions in the `/var/snmp/mib/sun.mib` file.

system

The `system` group reports statistics about a particular system (for example, a workstation or a printer).

`sysDescr` – A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. This value must only contain printable ASCII characters. (string[255])

`sysObjectID` – The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining what type of equipment is being managed. For example, if vendor “Flintstones, Inc.” was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its “Fred Router.” (objectid)

`sysUpTime` – Time (in hundredths of a second) since the network management portion of the system was last reinitialized. (timeticks)

`sysContact` – The textual identification of the contact person for this managed node, together with information on how to contact this person. (string[255])

`sysName` – An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name. (string[255])

`sysLocation` – The physical location of this node (for example, “telephone closet, 3rd floor” (string[255]))

`sysServices` – A value indicating the set of services that this entity primarily offers. (int) The value is a sum. This sum initially takes the value zero. Then, for each layer L in the range 1 through 7 for which this node performs transactions, 2 raised to (L - 1) is added to the sum. For example, a node that performs primarily routing functions would have a value of 4

(2**(3-1)). In contrast, a node that is a host offering application services would have a value of 72 (2**(4-1) + 2**(7-1)). Note that in the context of the Internet suite of protocols, values should be calculated accordingly:

Layer	Functionality
1	physical (such as repeaters)
2	datalink/subnetwork (such as bridges)
3	internet (such as IP gateways)
4	end-to-end (such as IP hosts)
7	applications (such as mail relays)

For systems including OSI protocols, Layers 5 and 6 may also be counted.

interfaces The `interfaces` group reports the number of interfaces handled by the agent.

`ifNumber` – The number of network interfaces, regardless of their current state, present on this system. (int)

ifTable The `ifTable` is a table of interface entries. The number of entries is given by the value of ifNumber.

`ifIndex` – A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one reinitialization of the entity's network management system to the next reinitialization. (int)

`ifDescr` – A textual string containing information about the interface. This string should include the name of the manufacturer, the product name, and the version of the hardware interface. (string[255])

`ifType` – The type of interface, distinguished according to the physical/link protocol(s) immediately below the network layer in the protocol stack. (enum)

`ifMtu` – The size of the largest datagram that can be sent/received on the interface, specified in octets. For interfaces used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface. (int)

`ifSpeed` – An estimate of the interface's current bandwidth in bits-per-second. For interfaces that do not vary in bandwidth, or for those where no accurate estimation can be made, this object should contain the nominal bandwidth. (gauge)

`if1hysAddress` – The interface's address at the protocol layer immediately below the network layer in the protocol stack. For interfaces without such an address (for example, a serial line), this object should contain an octet string of zero length. (octet[128])

`ifAdminStatus` – The desired state of the interface. The testing(3) state indicates that no operational packets can be passed. (enum)

`if OperStatus` – The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed. (enum)

`ifLastChange` – The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last reinitialization of the local network management subsystem, then this object contains a zero value. (timeticks)

`ifInOctets` – The total number of octets received on the interface, including framing characters. (counter) Returns a fixed value of 0.

`ifInUcastPkts` – The number of subnetwork-unicast packets delivered to a higher-layer protocol. (counter)

`ifInNUcastPkts` – The number of non-unicast (that is, subnetwork- broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol. (counter) Returns a fixed value of 0.

`ifInDiscards` – The number of inbound packets chosen to be discarded, even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space. (counter) Returns a fixed value of 0.

`ifInErrors` – The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. (counter)

`ifInUnknownProtos` – The number of packets received via the interface that were discarded because of an unknown or unsupported protocol. (counter) Returns a fixed value of 0.

`ifOutOctets` – The total number of octets transmitted out of the interface, including framing characters. (counter) Returns a fixed value of 0.

`ifOutUcastPkts` – The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent. (counter)

`ifOutNUcastPkts` – The total number of packets that higher-level protocols requested be transmitted to a non- unicast (that is, a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent. (counter) Returns a fixed value of 0.

`ifOutDiscards` – The number of outbound packets that were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space. (counter) Returns a fixed value of 0.

`ifOutErrors` – The number of outbound packets that could not be transmitted because of errors. (counter)

`ifOutQLen` – The length of the output packet queue (in packets). (gauge)

`ifSpecific` – A reference to MIB definitions specific to the particular media being used to realize the interface. For example, if the interface is realized by an Ethernet, then the value of this object refers to a document defining objects specific to Ethernet. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier. Any conformant implementation of ASN.1 and BER must be able to generate and recognize this value. (objectid)

atTable `atTable` Address Translation tables contain the NetworkAddress to physical address equivalences. Some interfaces do not use translation tables for determining address equivalences (for example, DDN-X.25 has an algorithmic method). If all interfaces are of this type, then the Address Translation table is empty, that is, has zero entries.

`atIfIndex` – The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

`atPhysAddress` – The media-dependent physical address. (octet[128]) Setting this object to a null string (one of zero length) has the effect of invaliding the corresponding entry in the atTable object. That is, it effectively dissociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant atPhysAddress object.

`atNetAddress` – The NetworkAddress (that is, the IP address) corresponding to the media-dependent physical address. (netaddress)

ip The `ip` group reports statistics about the Internet Protocol (IP) group.

`ipForwarding` – The indication of whether this entity is acting as an IP gateway in respect to the forwarding of datagrams received by, but not addressed to, this entity. IP gateways forward datagrams. IP hosts do not— except those source-routed via the host. (enum)

Note that for some managed nodes, this object may take on only a subset of the values possible. Accordingly, it is appropriate for an agent to return a “badValue” response if a management station attempts to change this object to an inappropriate value.

`ipDefaultTTL` – The default value inserted into the Time-To-Live field of the IP header of datagrams originated at this entity, whenever a TTL value is not supplied by the transport layer protocol. (int)

`ipInReceives` – The total number of input datagrams received from interfaces, including those received in error. (counter)

`ipInHdrErrors` – The number of input datagrams discarded due to errors in their IP headers, including bad checksums, version number mismatch, other format errors, time-to-live exceeded, errors discovered in processing their IP options, and so on. (counter)

`ipInAddrErrors` – The number of input datagrams discarded because the IP address in their IP header's destination field was not a valid address to be received at this entity. This count includes invalid addresses (for example, 0.0.0.0) and addresses of unsupported Classes (for example, Class E). For entities that are not IP Gateways and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address. (counter)

`ipForwDatagrams` – The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to that final destination. In entities that do not act as IP Gateways, this counter will include only those packets that were Source-Routed via this entity, and the Source- Route option processing was successful. (counter)

`ipInUnknownProtos` – The number of locally-addressed datagrams received successfully but discarded because of an unknown or unsupported protocol. (counter)

`ipInDiscards` – The number of input IP datagrams for which no problems were encountered to prevent their continued processing, but which were discarded, for example, for lack of buffer space. Note that this counter does not include any datagrams discarded while awaiting reassembly. (counter)

`ipInDelivers` – The total number of input datagrams successfully delivered to IP user-protocols (including ICMP). (counter)

`ipOutRequests` – The total number of IP datagrams that local IP user-protocols (including ICMP) supplied to IP in requests for transmission. Note that this counter does not include any datagrams counted in ipForwDatagrams. (counter)

`ipOutDiscards` – The number of output IP datagrams for which no problem was encountered to prevent their transmission to their destination, but which were discarded (for example, for lack of buffer space). Note that this counter would include datagrams counted in ipForwDatagrams if any such packets met this (discretionary) discard criterion. (counter)

`ipOutNoRoutes` – The number of IP datagrams discarded because no route could be found to transmit them to their destination. Note that this counter includes any packets counted in ipForwDatagrams which meet this “no-route” criterion. Note that this includes any datagrams that a host cannot route because all its default gateways are down. (counter)

`ipReasmTimeout` – The maximum number of seconds that received fragments are held while they are awaiting reassembly at this entity. (int)

`ipReasmReqds` – The number of IP fragments received that needed to be reassembled at this entity. (counter)

`ipReasmOKs` – The number of IP datagrams successfully reassembled. (counter)

`ipReasmFails` – The number of failures detected by the IP reassembly algorithm, for whatever reason: timed out, errors, and the like. Note that this is not necessarily a count of discarded IP fragments since some algorithms (notably the algorithm in RFC 815) can lose track of the number of fragments by combining them as they are received. (counter)

`ipFragOKs` – The number of IP datagrams that have been successfully fragmented at this entity. (counter)

`ipFragFails` – The number of IP datagrams that have been discarded because they needed to be fragmented at this entity but could not be, for example, because their "Don't Fragment" flag was set. (counter)

`ipFragCreates` – The number of IP datagram fragments that have been generated as a result of fragmentation at this entity. (counter)

`ipRoutingDiscards` – The number of routing entries that were chosen to be discarded even though they were valid. One possible reason for discarding such an entry could be to free-up buffer space for other routing entries. (counter) Returns a fixed value of 0.

ipAddrTable `ipAddrTable` is a table of addressing information relevant to this entity's IP addresses.

`ipAdEntAddr` – The IP address to which this entry's addressing information pertains. (netaddress)

`ipAdEntIfIndex` – The index value that uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

`ipAdEntNetMask` – The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1, and all the hosts bits set to 0. (netaddress)

`ipAdEntBcastAddr` – The value of the least-significant bit in the IP broadcast address used for sending datagrams on the (logical) interface associated with the IP address of this entry. For example, when the Internet standard all-ones broadcast address is used, the value will be 1. This value applies to both the subnet and network broadcasts addresses used by the entity on this (logical) interface. (int) Returns a fixed value of 1.

`ipAdEntReasmMaxSize` – The size of the largest IP datagram that this entity can reassemble from incoming IP fragmented datagrams received on this interface. (int) Returns a fixed value of 65535.

ipRouteTable The `ipRouteTable` is this entity's IP Routing table.

`ipRouteDest` – The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple routes to a single destination can appear in the table, but

access to such multiple entries is dependent on the table- access mechanisms defined by the network management protocol in use. (netaddress)

`ipRouteIfIndex` – The index value that uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

`ipRouteMetric1` – The primary routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

`ipRouteMetric2` – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

`ipRouteMetric3` – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

`ipRouteMetric4` – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

`ipRouteNextHop` – The IP address of the next hop of this route. (In the case of a route bound to an interface that is realized via a broadcast media, the value of this field is the agent's IP address on that interface.) (netaddress)

`ipRouteType` – The type of route. Note that the values direct (3) and indirect (4) refer to the notion of direct and indirect routing in the IP architecture. (enum)

Setting this object to the value invalid (2) has the effect of invalidating the corresponding entry in the ipRouteTable object. That is, it effectively dissociates the destination identified with said entry from the route identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipRouteType object.

`ipRouteProto` – The routing mechanism through which this route was learned. Inclusion of values for gateway routing protocols is not intended to imply that hosts should support those protocols. (enum)

`ipRouteAge` – The number of seconds since this route was last updated or otherwise determined to be correct. Note that no semantics of "too old" can be implied except through knowledge of the routing protocol by which the route was learned. (int) Returns a fixed value of 0.

`ipRouteMask` – Indicate the mask to be logical-ANDed with the destination address before being compared to the value in the ipRouteDest field. For those systems that do not support arbitrary subnet masks, an agent constructs the value of the ipRouteMask by determining whether the value of the correspondent ipRouteDest field belongs to a class-A, B, or C network, and then using one of:

Mask	Network
255.0.0.0	class-A
255.255.0.0	class-B
255.255.255.0	class-C

If the value of the ipRouteDest is 0.0.0.0 (a default route), then the mask value is also 0.0.0.0. It should be noted that all IP routing subsystems implicitly use this mechanism. (netaddress)

`ipRouteMetric5` – An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to –1. (int) Returns a fixed value of –1.

`ipRouteInfo` – A reference to MIB definitions specific to the particular routing protocol responsible for this route, as determined by the value specified in the route's ipRouteProto value. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier. Any conformant implementation of ASN.1 and BER must be able to generate and recognize this value. (objectid)

ipNetToMediaTable

The `ipNetToMediaTable` is the IP Address Translation table used for mapping from IP addresses to physical addresses.

`ipNetToMediaIfIndex` – The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex. (int)

`ipNetToMediaPhysAddress` – The media-dependent physical address. (octet[128])

`ipNetToMediaNetAddress` – The IpAddress corresponding to the media- dependent physical address. (netaddress)

`ipNetToMediaType` – The type of mapping. (enum) Returns a fixed value of (3)dynamic. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipNetToMediaTable. That is, it effectively dissociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipNetToMediaType object.

icmp The `icmp` group reports statistics about the ICMP group.

`icmpInMsgs` – The total number of ICMP messages that the entity received. Note that this counter includes all those counted by icmpInErrors. (counter)

`icmpInErrors` – The number of ICMP messages that the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, and the like.). (counter)

`icmpInDestUnreachs` – The number of ICMP Destination Unreachable messages received. (counter)

`icmpInTimeExcds` – The number of ICMP Time Exceeded messages received. (counter)

`icmpInParmProbs` – The number of ICMP Parameter Problem messages received. (counter)

`icmpInSrcQuenchs` – The number of ICMP Source Quench messages received. (counter)

`icmpInRedirects` – The number of ICMP Redirect messages received. (counter)

`icmpInEchos` – The number of ICMP Echo (request) messages received. (counter)

`icmpInEchoReps` – The number of ICMP Echo Reply messages received. (counter)

`icmpInTimestamps` – The number of ICMP Timestamp (request) messages received. (counter)

`icmpInTimestampReps` – The number of ICMP Timestamp Reply messages received. (counter)

`icmpInAddrMasks` – The number of ICMP Address Mask Request messages received. (counter)

`icmpInAddrMaskReps` – The number of ICMP Address Mask Reply messages received. (counter)

`icmpOutMsgs` – The total number of ICMP messages that this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors. (counter)

`icmpOutErrors` – The number of ICMP messages that this entity did not send due to problems discovered within ICMP, such as a lack of buffers. This value should not include errors discovered outside the ICMP layer, such as the inability of IP to route the resultant datagram. In some implementations there may be no types of errors that contribute to this counter's value. (counter)

`icmpOutDestUnreachs` – The number of ICMP Destination Unreachable messages sent. (counter)

`icmpOutTimeExcds` – The number of ICMP Time Exceeded messages sent. (counter)

`icmpOutParmProbs` – The number of ICMP Parameter Problem messages sent. (counter)

`icmpOutSrcQuenchs` – The number of ICMP Source Quench messages sent. (counter)

`icmpOutRedirects` – The number of ICMP Redirect messages sent. For a host, this object will always be zero, since hosts do not send redirects. (counter)

`icmpOutEchos` – The number of ICMP Echo (request) messages sent. (counter)

`icmpOutEchoReps` – The number of ICMP Echo Reply messages sent. (counter)

`icmpOutTimestamps` – The number of ICMP Timestamp (request) messages sent. (counter)

`icmpOutTimestampReps` – The number of ICMP Timestamp Reply messages sent. (counter)

`icmpOutAddrMasks` – The number of ICMP Address Mask Request messages sent. (counter)

`icmpOutAddrMaskReps` – The number of ICMP Address Mask Reply messages sent. (counter)

tcp The `tcp` group reports statistics about the TCP group.

`tcpRtoAlgorithm` – The algorithm used to determine the timeout value used for retransmitting unacknowledged octets. (enum)

`tcpRtoMin` – The minimum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the LBOUND quantity described in RFC 793. (int)

`tcpRtoMax` – The maximum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the UBOUND quantity described in RFC 793. (int)

`tcpMaxConn` – The limit on the total number of TCP connections that the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value –1. (int)

`tcpActiveOpens` – The number of times that TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state. (counter)

`tcpPassiveOpens` – The number of times that TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state. (counter)

`tcpAttemptFails` – The number of times that TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state. (counter)

`tcpEstabResets` – The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state. (counter)

`tcpCurrEstab` – The number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT. (gauge)

`tcpInSegs` – The total number of segments received, including those received in error. This count includes segments received on currently established connections. (counter)

`tcpOutSegs` – The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets. (counter)

`tcpRetransSegs` – The total number of segments retransmitted - that is, the number of TCP segments transmitted containing one or more previously transmitted octets. (counter)

`tcpInErrs` – The total number of segments received in error (for example, bad TCP checksums). (counter)

`tcpOutRsts` – The number of TCP segments sent containing the RST flag. (counter)

tcpConnTable The `tcpConnTable` is a table containing TCP connection-specific information.

`tcpConnState` – The state of this TCP connection. (enum)

The only value that may be set by a management station is deleteTCB(12). Accordingly, it is appropriate for an agent to return a "badValue" response if a management station attempts to set this object to any other value.

If a management station sets this object to the value deleteTCB(12), then this has the effect of deleting the TCB (as defined in RFC 793) of the corresponding connection on the managed node. This results in immediate termination of the connection.

As an implementation-specific option, an RST segment may be sent from the managed node to the other TCP endpoint. (Note, however, that RST segments are not sent reliably.)

`tcpConnLocalAddress` – The local IP address for this TCP connection. For a connection in the listen state that is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0 is used. (netaddress)

`tcpConnLocalPort` – The local port number for this TCP connection. (int)

`tcpConnRemAddress` – The remote IP address for this TCP connection. (netaddress)

`tcpConnRemPort` – The remote port number for this TCP connection. (int)

upd The `udp` group reports statistics about the UDP group.

`udpInDatagrams` – The total number of UDP datagrams delivered to UDP users. (counter) Returns a fixed value of 0.

`udpNoPorts` – The total number of received UDP datagrams for which there was no application at the destination port. (counter) Returns a fixed value of 0.

`udpInErrors` – The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port. (counter)

`udpOutDatagrams` – The total number of UDP datagrams sent from this entity. (counter) Returns a fixed value of 0.

udpTable The `udpTable` is a table containing UDP listener information.

`udpLocalAddress` – The local IP address for this UDP listener. For a UDP listener that is willing to accept datagrams for any IP interface associated with the node, the value 0.0.0.0 is used. (netaddress)

`udpLocalPort` – The local port number for this UDP listener. (int)

snmp The `snmp` group reports statistics about the SNMP group.

`snmpInPkts` – The total number of Messages delivered to the SNMP entity from the transport service. (counter)

`snmpOutPkts` – The total number of SNMP Messages passed from the SNMP protocol entity to the transport service. (counter)

`snmpInBadVersions` – The total number of SNMP Messages delivered to the SNMP protocol entity that were for an unsupported SNMP version. (counter)

`snmpInBadCommunityNames` – The total number of SNMP Messages delivered to the SNMP protocol entity that used a SNMP community name not known to said entity. (counter)

`snmpInBadCommunityUses` – The total number of SNMP Messages delivered to the SNMP protocol entity, which represented an SNMP operation not allowed by the SNMP community named in the Message. (counter)

`snmpInASNParseErrs` – The total number of ASN.1 or BER errors encountered by the SNMP protocol entity when decoding received SNMP Messages. (counter)

`snmpInTooBigs` – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “tooBig.” (counter)

`snmpInNoSuchNames` – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “noSuchName.” (counter)

`snmpInBadValues` – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is “badValue.” (counter)

`snmpInReadOnlys` – The total number valid SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is "readOnly." It should be noted that it is a protocol error to generate an SNMP PDU that contains the value "readOnly" in the error-status field. This object is provided as a means of detecting incorrect implementations of the SNMP. (counter)

`snmpInGenErrs` – The total number of SNMP PDUs delivered to the SNMP protocol entity for which the value of the error-status field is "genErr." (counter)

`snmpInTotalReqVars` – The total number of MIB objects successfully retrieved by the SNMP protocol entity as the result of receiving valid SNMP Get-Request and Get-Next PDUs. (counter)

`snmpInTotalSetVars` – The total number of MIB objects successfully altered by the SNMP protocol entity as the result of receiving valid SNMP Set-Request PDUs. (counter)

`snmpInGetRequests` – The total number of SNMP Get-Request PDUs accepted and processed by the SNMP protocol entity. (counter)

`snmpInGetNexts` – The total number of SNMP Get-Next PDUs accepted and processed by the SNMP protocol entity. (counter)

`snmpInSetRequests` – The total number of SNMP Set-Request PDUs accepted and processed by the SNMP protocol entity. (counter)

`snmpInGetResponses` – The total number of SNMP Get-Response PDUs accepted and processed by the SNMP protocol entity. (counter)

`snmpInTraps` – The total number of SNMP Trap PDUs accepted and processed by the SNMP protocol entity. (counter)

`snmpOutTooBigs` – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status field is "tooBig." (counter)

`snmpOutNoSuchNames` – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status is "noSuchName." (counter)

`snmpOutBadValues` – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status field is "badValue." (counter)

`snmpOutGenErrs` – The total number of SNMP PDUs generated by the SNMP protocol entity for which the value of the error-status field is "genErr." (counter)

`snmpOutGetRequests` – The total number of SNMP Get-Request PDUs which have been generated by the SNMP protocol entity. (counter)

`snmpOutGetNexts` – The total number of SNMP Get-Next PDUs generated by the SNMP protocol entity. (counter)

`snmpOutSetRequests` – The total number of SNMP Set-Request PDUs generated by the SNMP protocol entity. (counter)

`snmpOutGetResponses` – The total number of SNMP Get-Response PDUs generated by the SNMP protocol entity. (counter)

`snmpOutTraps` – The total number of SNMP Trap PDUs generated by the SNMP protocol entity. (counter)

`snmpEnableAuthenTraps` – Indicates whether the SNMP agent process is permitted to generate authentication-failure traps. The value of this object overrides any configuration information. As such, it provides a means whereby all authentication-failure traps may be disabled. (enum)

Note that this object must be stored in non-volatile memory, so that it remains constant between reinitializations of the network management system.

The following are Sun-specific group and table definitions.

sunSystem The `sunSystem` group reports general system information.

`agentDescr` – The SNMP agent's description of itself. (string[255])

`hostID` – The unique Sun hardware identifier. The value returned is four byte binary string. (octet[4])

`motd` – The first line of `/etc/motd`. (string[255])

`unixTime` – The UNIX system time. Measured in seconds since January 1, 1970 GMT. (counter)

sunProcessTable The `sunProcessTable` table reports UNIX process table information.

`psProcessID` – The process identifier for this process. (int)

`psParentProcessID` – The process identifier of this process's parent. (int)

`psProcessSize` – The combined size of the data and stack segments (in kilobytes.) (int)

`psProcessCpuTime` – The CPU time (including both user and system time) consumed so far. (int)

`psProcessState` – The run-state of the process. (octet[4])

R	Runnable
T	Stopped
P	In page wait

D	Non-interruptable wait
S	Sleeping (less than 20 seconds)
I	Idle (more than 20 seconds)
Z	Zombie

`psProcessWaitChannel` – Reason process is waiting. (octet[16])

`psProcessTTY` – Terminal, if any, controlling this process. (octet[16])

`psProcessUserName` – Name of the user associated with this process. (octet[16])

`psProcessUserID` – Numeric form of the name of the user associated with this process. (int)

`psProcessName` – Command name used to invoke this process. (octet[64])

`psProcessStatus` – Setting this variable will cause a signal of the set value to be sent to the process. (int)

sunHostPerf The `sunHostPerf` group reports hostperf information.

`rsUserProcessTime` – Total number of timeticks used by user processes since the last system boot. (counter)

`rsNiceModeTime` – Total number of timeticks used by "nice" mode since the last system boot. (counter)

`rsSystemProcessTime` – Total number of timeticks used by system processes since the last system boot. (counter)

`rsIdleModeTime` – Total number of timeticks in idle mode since the last system boot. (counter)

`rsDiskXfer1` – Total number of disk transfers since the last boot for the first of four configured disks. (counter)

`rsDiskXfer2` – Total number of disk transfers since the last boot for the second of four configured disks. (counter)

`rsDiskXfer3` – Total number of disk transfers since the last boot for the third of four configured disks. (counter)

`rsDiskXfer4` – Total number of disk transfers since the last boot for the fourth of four configured disks. (counter)

`rsVPagesIn` – Number of pages read in from disk. (counter)

`rsVPagesOut` – Number of pages written to disk. (counter)

`rsVSwapIn` – Number of pages swapped in. (counter)

`rsVSwapOut` – Number of pages swapped out. (counter)

`rsVIntr` – Number of device interrupts. (counter)

`rsIfInPackets` – Number of input packets. (counter)

`rsIfOutPackets` – Number of output packets. (counter)

`rsIfInErrors` – Number of input errors. (counter)

`rsIfOutErrors` – Number of output errors. (counter)

`rsIfCollisions` – Number of output collisions. (counter)

Files

`/etc/snmp/conf/snmpd.conf`	configuration information
`/etc/snmp/conf/mibiisa.acl`	access control file
`/var/snmp/mib/sun.mib`	standard SNMP MIBII file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmibii
Interface Stability	Obsolete

See Also `inetd(1M)`, `select(3C)`, `recvfrom(3SOCKET)`, `sendto(3SOCKET)`, `attributes(5)`, `gld(7D)`,

Diagnostics

`cannot dispatch request`	The proxy cannot dispatch the request. The rest of the message indicates the cause of the failure.
`select(3C) failed`	A `select(3C)` call failed. The rest of the message indicates the cause of the failure.
`sendto(3SOCKET) failed`	A `sendto(3SOCKET)` call failed. The rest of the message indicates the cause of the failure.
`recvfrom(3SOCKET) failed`	A `recvfrom(3SOCKET)` call failed. The rest of the message indicates the cause of the failure.

`no response from system`	The SNMP agent on the target system does not respond to SNMP requests. This error might indicate that the SNMP agent is not running on the target system, the target system is down, or the network containing the target system is unreachable.
`response too big`	The agent could not fit the results of an operation into a single SNMP message. Split large groups or tables into smaller entities.
`missing attribute`	An attribute is missing from the requested group.
`bad attribute type`	An object attribute type received from the SNMP agent that does not match the attribute type specified by the proxy agent schema. The rest of the message indicates the expected type and received type.
`cannot get sysUpTime`	The proxy agent cannot get the variable *sysUpTime* from the SNMP agent.
`sysUpTime type bad`	The variable *sysUpTime* received from the SNMP agent has the wrong data type.
`unknown SNMP error`	An unknown SNMP error was received.
`bad variable value`	The requested specified an incorrect syntax or value for a set operation.
`variable is read only`	The SNMP agent did not perform the set request because a variable to set may not be written.
`general error`	A general error was received.
`cannot make request PDU`	An error occurred building a request PDU.
`cannot make request varbind list`	An error occurred building a request variable binding list.

`cannot parse response PDU`	An error occurred parsing a response PDU.
`request ID - response ID mismatch`	The response ID does not match the request ID.
`string contains non-displayable characters`	A displayable string contains non-displayable characters.
`cannot open schema file`	An error occurred opening the proxy agent schema file.
`cannot parse schema file`	The proxy agent couldn't parse the proxy agent schema file.
`cannot open host file`	An error occurred opening the file associated with the *na.snmp.hostfile* keyword in `/etc/snmp/conf/snmpd.conf`
`cannot parse host file`	The proxy agent was unable to parse the file associated with the *na.snmp.hostfile* keyword in `/etc/snmp/conf/snm.conf`.
`attribute unavailable for set operations`	The set could not be completed because the attribute was not available for set operations.

Bugs The `mibiisa` utility returns the wrong interface speed for the SBUS FDDI interface (for example, “bf0”).

The `mibiisa` utility does not return a MAC address for the SBUS FDDI interface (for example, “bf0”).

Process names retrieved from `mibiisa` contain a leading blank space.

When you change attribute values in the system group with an SNMP set request, the change is effective only as long as `mibiisa` is running. `mibiisa` does not save the changes to `/etc/snmp/conf/snmpd.conf`.

Name mipagent – Mobile IP agent

Synopsis

```
/usr/lib/inet/mipagent
```

Description The `mipagent` utility implements the Mobile IP home agent and foreign agent functionality described in *RFC 2002, IP Mobility Support*. The term "mobility agent" is used to refer to the home agent and foreign agent functionality collectively. `mipagent` responds to Mobile IP registration and deregistration requests and router discovery solicitation messages from a mobile node. Besides responding to external messages, the `mipagent` utility also tasks on a periodic basis, such as aging the mobility bindings and visitor entries and sending agent advertisements. The mobility agent can also handle direct delivery style reverse tunneling as specified in *RFC 2344, Reverse Tunneling for Mobile IP*. Limited private address support for mobile nodes is also available. In addition, separate IPsec policies for registration requests, replies, and tunnel traffic can be configured to protect the datagrams associated with these between two mobility agents.

Run the `mipagent` daemon as root using the start-up script, which has the following syntax:

```
example# /etc/init.d/mipagent [start|stop]
```

`/etc/inet/mipagent.conf` must be present before you start-up the `mipagent` daemon. See `mipagent.conf(4)`. At start up, `mipagent` reads the configuration information from `/etc/inet/mipagent.conf`. The `mipagent` daemon records a continuous log of its activities by means of `syslog()`. See `syslog(3C)`. You can use the *LogVerbosity* parameter in `/etc/inet/mipagent.conf` to control the verbosity level of the log.

The `mipagent` daemon can be terminated either by the script:

```
example# /etc/init.d/mipagent stop
```

or by the `kill` command.

Periodically while running, or if terminated or shutdown, the `mipagent` daemon stores the following internal state information in `/var/inet/mipagent_state`:

- a list of the mobile nodes supported as home agents;
- their current care-of addresses; and
- the remaining registration lifetimes.

If the `mipagent` utility is terminated for maintenance and restarted, `mipagent_state` is used to recreate as much of the mobility agent's internal state as possible. This minimizes service disruption for mobile nodes that may be visiting other networks. If `mipagent_state` exists, it is read immediately after `mipagent.conf` when `mipagent` is restarted. The format of `mipagent_state` is undocumented since it is likely to change and programs other than `mipagent` should not use it for any purpose. A separate utility program `mipagentstat` is provided for monitoring `mipagent`.

Exit Status The following exit values are returned:

`0` The daemon started successfully.

`-1` The daemon failed to start.

Files

`/etc/inet/mipagent.conf`	Configuration file for Mobile IP mobility agent.
`/var/inet/mipagent_state`	File where private state information from `mipagent` is stored.
`/etc/init.d/mipagent [start|stop]`	`mipagent` start-up script.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmipu

See Also `mipagentstat(1M)`, `mipagentconfig(1M)`, `syslog(3C)`, `mipagent.conf(4)`, `attributes(5)`

Montenegro, G., editor.*RFC 2344, Reverse Tunneling for Mobile IP*. Network Working Group. May 1998.

Perkins, C. *RFC 2002, IP Mobility Support*. Network Working Group. October 1996.

Diagnostics The `mipagent` utility exits with an error if the configuration file, `mipagent.conf`, cannot be read successfully. Upon receiving a `SIGTERM` or `SIGINT` signal, `mipagent` cleans its internal state, including any changes to the routing and ARP tables, and exits.

Notes The foreign agent adds host– specific local routes to its routing table for visiting mobile nodes after they are successfully registered. If a visiting mobile node departs without sending a de-registration message through the foreign agent, these routing entries persist until the mobile node's previous registration expires. Any packets that arrive at the foreign agent for the departed mobile node during this time, for example because the foreign agent is also a router for the foreign network, will be lost. System administrators can configure foreign agents to accept only short registration lifetimes. This will automatically restrict the maximum duration for which a departed mobile node will be temporarily unreachable.

Home and foreign agents dynamically add and delete IPsec policies configured with a mobility agent peer. Those pertaining to the tunnel are only added when the tunnel is plumbed. At this time, IPsec tunnel policies must be identical in the forward and reverse direction. IPsec policies pertaining to permiting registration requests on the home agent are added to the IPsec policy file at init time as it must be ready to receive these at any time. Otherwise, IPsec policies pertaining to registration request and reply messages with a mobility agent peer are added as soon as they are needed, and are not removed until all mobile nodes are no longer registered with the mobility agent peer, at which point the tunnels are torn down.

Name mipagentconfig – configure Mobility IP Agent

Synopsis

```
/sbin/mipagentconfig [-f configfile] command dest
     [parameters]...
```

Description The `mipagentconfig` utility is used to configure the Mobility IP Agent. `mipagentconfig` allows the user to change settings. The mipagentconfig user can also add and delete mobility clients, Pools, and SPIs in the mobility agent configuration file.

Options The following options are supported:

`-f` *configfile* Use the specified configuration file instead of the system default, `/etc/inet/mipagent.conf`.

Operands The *command* operand, as well as the parameters for each command are described below. See `mipagent.conf(4)` for the default values of the configuration operands that are described here.

`add` This command adds advertisement parameters, security parameters, SPIs, or addresses to the configuration file, based on the destination *dest*.

`add Address` *ipAddress attr_value*
Add the specified *ipAddress* with the specified `SPI`. To add an NAI address, you must specify the `Pool`.

`add adv` *device*
Enable home and foreign agent functionality on the specified interface.

`add adv` *device* `AdvLifetime` *seconds*
Add `AdvLifetime` to the specified device.

`add adv` *device* `RegLifetime` *seconds*
Add `RegLifetime` to the specified device.

`add adv` *device* `AdvFrequency` *seconds*
Add `AdvFrequency` to the specified device.

`add adv` *device* `AdvInitCount` *count*
Add initial unsolicited advertisement count. *count* should be a small integer.

`add adv` *device* `AdvLimitUnsolicited yes | no`
Enable limited or unlimited unsolicited advertisements for foreign agent. Accepted values are:

`yes` Limit unsolicited advertisement to `AdvInitCount` initial advertisements.

`no` Do not limit unsolicited advertisement. The advertisement should take place periodically at the frquency specified by `AdvFrequency`.

`add adv` *device* `HomeAgent yes | no`
Add the `HomeAgent` flag to the specified device.

`add adv` *device* `ForeignAgent yes` | `no`
Add the `ForeignAgent` flag to the specified device.

`add adv` *device* `PrefixLengthExt yes` | `no`
Add the `PrefixLengthExt` flag to the specified device.

`add adv` *device* `NAIExt yes` | `no`
Add the `NAIExt` flag to the specified device.

`add adv` *device* `Challenge yes` | `no`
Add the `Challenge` flag to the specified device.

`add adv` *device* `ReverseTunnel no` | `neither fa ha yes` | `both`
Add the level of `ReverseTunnel` support that is indicated to the specified device. Possible values include:

`no` Do not support `ReverseTunnel` as either a foreign agent or a home agent on this device. Does not advertise reverse tunneling nor accept a registration requesting reverse tunnel support on this device.

`neither` Do not support `ReverseTunnel` as either a foreign agent or a home agent on this device. Do not advertise reverse tunneling or accept a registration that requests reverse tunnel support on this device.

`fa` When the foreign agent processes a registration request received on this device, check to see if the mobile node requests that a reverse tunnel be set up to its home agent. If so, perform the necessary encapsulation of datagrams to the mobile node's home agent as described in *RFC 3024*. This means that a mobile node must see the agent advertising reverse tunnel support, so the reverse tunnel bit is advertised in the agent advertisement on this device.

`ha` When the home agent processes a registration request received on this device, check to see if the mobile node requests that a reverse tunnel be set up from its care-of address. If so, perform the necessary decapsulation as described in *RFC 3024*. This does not mean the home agent is advertising support of reverse tunneling on this device. Mobile nodes are only interested in the advertisement flags if mobile nodes are going to use foreign agent services. Moreover, reverse tunnels by definition originate at the care-of address. HA support is therefore only of interest to the owner of the care-of address.

`yes` Whenever the mobility agent is processing a registration request received on this device, check to see if the mobile node is

requesting that a reverse tunnel be set up. If so, apply *RFC 3024* as appropriate, either as an encapsulating foreign agent, or as a decapsulating home agent, depending on how this mobility agent is servicing the specific mobile node. As a result, the mobility agent advertises reverse tunnel support on this device.

`both` Whenever the mobility agent is processing a registration request received on this device, check to see if the mobile node is requesting that a reverse tunnel be set up. If so, apply *RFC 3024* as appropriate, either as an encapsulating foreign agent, or a decapsulating home agent, depending on how this mobility agent services the specific mobile node. As a result, the mobility agent advertises reverse tunnel support on this device.

`add adv` *device* `ReverseTunnelRequired no` | `neither fa ha yes` | `both yes` | `both`
Add the requirement that the `ReverseTunnel` flag be set in any registration request received on the indicated device. Possible values include:

`no` Reverse tunneling is not required by the `mipagent` on this device.

`neither` Reverse tunneling is not required by the `mipagent` on this device.

`fa` The `ReverseTunnel` flag is required to be set in registration requests received by the foreign agent on this device.

`ha` The `ReverseTunnel` flag is required to be set in registration requests received by the home agent on this device.

`yes` The `ReverseTunnel` flag is required to be set in all registration requests received by either home and or foreign agents on this device.

`both` The `ReverseTunnel` flag is required to be set in all registration requests received by either home and or foreign agents on this device.

`add Pool` *number startAddr length*
Add the specified `Pool` with the specified start addresses and length.

`add SPI` *number replay Key*
Add the specified `SPI` with the given replay type and key. The replay type can have a value of `none` or `timestamps`.

`add HA-FAAuth yes` | `no`
Add the `HA-FAAuth` flag.

`add MN-FAAuth yes` | `no`
Add the `MN-FAAuth` flag.

`add MaxClockSkew` *seconds*
Add the `MaxClockSkew`.

`add KeyDistribution` *type*
Add the `KeyDistribution` type. The only value for `KeyDistribution` that is supported at this time is `file`.

`change` Depending on the destination *dest*, this command will change advertisement parameters, security parameters, SPIs, or addresses in the configuration file. Any of the above destinations are valid.

`delete` Depending on the destination *dest*, this command will delete advertisement parameters, security parameters, SPIs, or addresses from the configuration file. Any destination discussed above is valid.

`get` Display all of the parameters associated with *dest*. Any destination discussed above is valid.

Examples

EXAMPLE 1 Adding an SPI, a Pool, and a Mobile Node and Requiring Reverse Tunneling on a Device to the *configfile*

The following example adds an SPI, a `Pool`, a `mobile node`, and requires reverse tunneling for the foreign agent in the *configfile*. First, the SPI of 250 is added. Then, a Pool of 200 addresses starting at `192.168.168.1` is added. `joe@mobile.com` is added with an SPI of 250 and using Pool 1. Finally, reverse tunneling is required for the foreign agent on device `eri0`.

```
example# mipagentconfig add SPI 250 ReplayMethod none
example# mipagentconfig add SPI 250 Key 00ff00ff00ff
example# mipagentconfig add Pool 1 192.168.168.1 200
example# mipagentconfig add Address joe@mobile.com 250 1
example# mipagentconfig add adv eri0 reversetunnel fa
example# mipagentconfig add adv eri0 reversetunnelrequired fa
```

EXAMPLE 2 Adding Dynamic Interface Mobility Support on PPP Interfaces

The following example adds dynamic interface mobility support on PPP interfaces. Note that in some shells the backslash (\) escape character is required to bypass the expansion of the asterix ("*") and pass the "*" character to `mipagentconfig`. The example also indicates that all the new PPP interfaces offer reverse tunnel service.

```
example# mipagentconfig add adv sppp\* reversetunnel yes
example# mipagentconfig add adv sppp\* AdvLimitUnsolicited yes
example# mipagentconfig add adv sppp\* AdvInitCount 3
example# mipagentconfig add adv sppp\* AdvFrequency 1
```

EXAMPLE 3 Adding IPsec Policies to an Agent-Peer Entry

The following example adds IPsec policies to an existing mobility agent entry, then displays the configuration for the mobility agent peer. The backslash (\) character denotes a line continuation for the formatting of this example.

EXAMPLE 3 Adding IPsec Policies to an Agent-Peer Entry *(Continued)*

```
example# mipagentconfig add Address 192.168.10.1 \
           IPsecRequest apply {auth_algs md5 sa shared}
example# mipagentconfig add Address 192.168.10.1 \
           IPsecReply permit {auth_algs md5}
example# mipagentconfig add Address 192.168.10.1 \
           IPsecTunnel permit {encr_auth_algs md5 encr_algs 3des}
example# mipagentconfig get Address 192.168.10.1

[Address 192.168.10.1]
     Type = agent
     SPI = 137
     IPsecRequest = apply {auth_algs md5 sa shared}
     IPsecReply = permit {auth_algs md5}
     IPsecTunnel = \
     permit {encr_auth_algs md5 encr_algs 3des}
```

EXAMPLE 4 Modifying an SPI

To modify the SPI associated with joe, first, use the command `get` to verify the existing settings, then change the SPI from 250 to 257.

```
example# mipagentconfig get Address joe@mobile.com
Address: joe@mobile.com
SPI:  250
Pool: 1
example# mipagentconfig change Address joe@mobile.com 257 1
```

EXAMPLE 5 Deleting a Pool

Use the following example to delete Pool 3:

```
example# mipagentconfig delete Pool 3
```

EXAMPLE 6 Using the `mipagentconfig` command

Use the following example to delete Pool 3:

```
example# mipagentconfig delete Pool 3
```

Exit Status

The following exit values are returned:

`0`	Successful completion
`non-zero`	An error occurred

Files

`/etc/inet/mipagent.conf`	Configuration file for Mobile IP mobility agent
`/etc/inet/mipagent.conf-sample`	Sample configuration file for mobility agents

`/etc/inet/mipagent.conf.ha-sample`	Sample configuration file for home agent functionality.
`/etc/inet/mipagent.conf.fa-sample`	Sample configuration file for foreign agent functionality.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmipu

See Also `mipagent(1M)`, `mipagent.conf(4)`, `attributes(5)`

Montenegro, G., editor. *RFC 3024, Reverse Tunneling for Mobile IP, revised*. The Internet Society. January, 2001.

Perkins, C. *RFC 2002, IP Mobility Support*. Network Working Group. October 1996.

Name mipagentstat – show Mobile IP Mobility Agent status

Synopsis

```
mipagentstat [-fhp]
```

Description Use the `mipagentstat` utility to display the content of various Mobile-IP related data structures.

Visitor Table (First Form)

The visitor table display lists information for all mobile nodes registered with the foreign agent, one mobile node per line. This list consists of the mobile node's home address or Network Access Identifier (NAI), home agent address, total registration lifetime and the number of seconds remaining before the registration expires.

The following command line shows the output from a foreign agent with two mobile nodes registered:

```
example# mipagentstat -f


Mobile Node     Foreign Agent  Time Granted    Time Remaining     Flags
                               (in secs)       (in secs)
foobar@xyz.com  fa1@tuv.com    600             125
10.1.5.23       123.2.5.12     1000            10                 R
```

An "R" in the flags column indicates a reverse tunnel is present. No reverse tunnel is configured for the mobile node `foobar@xyz.com`. A reverse tunnel is configured from mobile node `10.1.5.23`.

Binding Table (Second Form)

The binding table display lists information for all mobile nodes registered with the home agent, one mobile node per line. This list consists of the mobile node's home address or NAI, foreign agent address, total registration lifetime and the number of seconds remaining before the registration expires.

Use the following command line to show the output from a home agent with two active mobile nodes:

```
example# mipagentstat -h


Mobile Node     Home Agent     Time Granted    Time Remaining     Flags
                               (in secs)       (in secs)
foobar@xyz.com  ha1@xyz.com    600             125
10.1.5.23       10.1.5.1       1000            10                 R
```

An "R" in the flags column indicates a reverse tunnel is present. No reverse tunnel is configured for the mobile node `foobar@xyz.com`. A reverse tunnel is configured for mobile node `10.1.5.23`.

Agent Table (Third Form)

The agent table display lists information for all current mobility agent-peers, that is all mobility agents with which mobile-nodes we are servicing are trying to obtain service.

Provided in this display are the IPsec protection mechanisms being used with registration requests, replies, and tunnels.

Use the following command line to show the output from a home agent with two (foreign) mobility agent peers:

```
example# mipagentstat -hp

Foreign                   ..... Security Association(s).....
Agent                     Requests Replies  FTunnel  RTunnel
-----------------------   -------- -------- -------- --------
fa.eng.example.com        AH,ESP   AH,ESP   AH,ESP   AH,ESP
fa.central.example.com    AH       AH       ESP      ESP
```

Use the following command line to show the output from a home agent with two (foreign) mobility agent peers:

```
example# mipagentstat -fp


Home                      ..... Security Association(s) .....
Agent                     Requests Replies  FTunnel  RTunnel
-----------------------   -------- -------- -------- --------
ha.eng.example.com        AH,ESP   AH,ESP   AH,ESP   AH,ESP
ha.central.example.com
```

Use of the `-p` option without specifying the agent results in both displays described above, that is one display for each agent.

An `AH` in any column indicates the IPsec `AH` mechanism is in place for those datagrams.

An `ESP` in any column indicates the IPsec `ESP` mechanism is in place for those datagrams.

Options The following options are supported:

`-f` Display the list of active mobile nodes in the foreign agent's visitor's list.

`-h` Display the list of active mobile nodes in the home agent's binding table.

`-p` Display the list of mobility agent peers, and the IPsec protection mechanisms currently in use for registration and tunnel traffic.

Exit Status The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmipu

See Also mipagent(1M), mipagentconfig(1M), mipagent.conf(4), attributes(5)

Aboda, B., and Beadles, M. *RFC 2486, The Network Access Identifier*. The Internet Society, 1999.

Name mkbootmedia – create bootable Solaris ISO image

Synopsis

```
/usr/bin/mkbootmedia -v [-l label] media-root iso
```

Description The `mkbootmedia` utility takes *media-root* (the root of an on-disk Solaris install media) as input and creates a bootable Solaris ISO image in the file *iso*, using `mkisofs(8)`. The file can then be burned onto a CD/DVD with utilities such as `cdrw(1)` or `cdrecord(1)`. (Neither `mkisofs(8)` nor `cdrecord(1)` are SunOS man pages.)

Caution – The directory tree *media-root* must contain the file `boot/grub/stage2_eltorito`, which will be written to the media boot sectors. This file will be modified with some boot information, thus it must be writable. If necessary, first save a copy prior to running this utility.

Options The following options are supported:

`-l` *label*
Sets *label* as the label/volume name of the ISO image.

`-v`
Verbose. Multiple `-v` options increase verbosity.

Operands The following operands are supported:

media-root
Top-level directory of an on-disk Solaris install media.

iso
Name of the output file which will contain the resulting ISO image.

Examples **EXAMPLE 1** Creating an ISO Image and Burning a CD/DVD

The following commands create an ISO image from the content of `s10u1` and burn the image to a CD/DVD.

```
# /usr/bin/mkbootmedia s10u1 s10u1.iso
# /usr/bin/cdrw -i s10u1.iso
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed

See Also `cdrw(1)`, `attributes(5)`

`mkisofs(8)`, (`/usr/share/man/man8/mkisofs.8`), in the `SUNWfsman` package (not a SunOS man page)

Name mkdevalloc – Make device_allocate entries

Synopsis

```
/usr/sbin/mkdevalloc
```

Description The `mkdevalloc` command writes to standard out a set of `device_allocate(4)` entries describing the system's frame buffer, audio and removable media devices.

The `mkdevalloc` command is used by the `init.d(4)` scripts to create or update the `/etc/security/device_allocate` file.

Entries are generated based on the device special files found in `/dev`. For the different categories of devices, the `mkdevalloc` command checks for the following files under `/dev`:

```
audio           /dev/audio, /dev/audioctl, /dev/sound/...
tape            /dev/rst*, /dev/nrst*, /dev/rmt/...
floppy          /dev/diskette, /dev/fd*, /dev/rdiskette, /dev/rfd*
removable disk  /dev/sr*, /dev/nsr*, /dev/dsk/c0t?d0s?, /dev/rdsk/c0t?d0s?
frame buffer    /dev/fb
```

All entries set the *device-minimum* and *device-maximum* fields to the hex representations of `ADMIN_LOW` and `ADMIN_HIGH`, respectively. The *device-authorization* field is set to `solaris.device.allocate`, except for the `framebuffer` entry, where it is set to `*`. The *device-name*, *device-type* and *device-clean* fields are set to the following values:

```
                device-name      device-type   device-clean

audio           audio            audio         audio_clean_wrapper
tape            mag_tape_0,1,... st            st_clean
floppy          floppy_0,1,...   fd            disk_clean
removable disk  cdrom_0,1,...    sr            disk_clean
frame buffer    framebuffer      fb            /bin/true
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Obsolete

See Also `allocate(1)`, `bsmconv(1M)`, `attributes(5)`

Notes `mkdevalloc` might not be supported in a future release of the Solaris operating system.

Name mkdevmaps – make device_maps entries

Synopsis

```
/usr/sbin/mkdevmaps
```

Description The `mkdevmaps` command writes to standard out a set of `device_maps(4)` entries describing the system's frame buffer, audio, and removable media devices.

The `mkdevmaps` command is used by the `init.d(4)` scripts to create or update the `/etc/security/device_maps` file.

Entries are generated based on the device special files found in `/dev`. For the different categories of devices, the `mkdevmaps` command checks for the following files under `/dev`:

```
audio           /dev/audio, /dev/audioctl, /dev/sound/...
tape            /dev/rst*, /dev/nrst*, /dev/rmt/...
floppy          /dev/diskette, /dev/fd*, /dev/rdiskette, /dev/rfd*
removable disk  /dev/dsk/c0t?d0s?, /dev/rdsk/c0t?d0s?
frame buffer    /dev/fb
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Obsolete

See Also `allocate(1)`, `bsmconv(1M)`, `attributes(5)`

Notes `mkdevmaps` might not be supported in a future release of the Solaris operating system.

Name mkfifo – make FIFO special file

Synopsis

```
/usr/bin/mkfifo [-m mode] path...
```

Description The `mkfifo` utility creates the FIFO special files named by its argument list. The arguments are taken sequentially, in the order specified; and each FIFO special file is either created completely or, in the case of an error or signal, not created at all.

If errors are encountered in creating one of the special files, `mkfifo` writes a diagnostic message to the standard error and continues with the remaining arguments, if any.

The `mkfifo` utility calls the library routine `mkfifo(3C)`, with the *path* argument is passed as the *path* argument from the command line, and *mode* is set to the equivalent of `a=rw`, modified by the current value of the file mode creation mask `umask(1)`.

Options The following option is supported:

`-m` *mode* Sets the file permission bits of the newly-created FIFO to the specified *mode* value. The *mode* option-argument will be the same as the *mode* operand defined for the `chmod(1)` command. In <*symbolic*mode> strings, the *op* characters + and – will be interpreted relative to an assumed initial mode of `a=rw`.

Operands The following operand is supported:

file A path name of the FIFO special file to be created.

Usage See `largefile(5)` for the description of the behavior of `mkfifo` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Environment Variables See `environ(5)` for descriptions of the following environment variables that affect the execution of `mkfifo`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Exit Status The following exit values are returned:

`0` All the specified FIFO special files were created successfully.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu
Interface Stability	Standard

See Also `mkfifo(3C)`, `attributes(5)`, `environ(5)`, `largefile(5)`, `standards(5)`

Name mkfile – create a file

Synopsis

```
mkfile [-nv] size [g | k | b | m] filename...
```

Description `mkfile` creates one or more files that are suitable for use as NFS-mounted swap areas, or as local swap areas. When a root user executes `mkfile()`, the sticky bit is set and the file is padded with zeros by default. When non-root users execute `mkfile()`, they must manually set the sticky bit using `chmod(1)`. The default `size` is in bytes, but it can be flagged as gigabytes, kilobytes, blocks, or megabytes, with the `g`, `k`, `b`, or `m` suffixes, respectively.

Options

`-n` Create an empty *filename*. The size is noted, but disk blocks are not allocated until data is written to them. Files created with this option cannot be swapped over local UFS mounts.

`-v` Verbose. Report the names and sizes of created files.

Exit Status The following exit values are returned:

`0` Success.

`>0` An error occurred.

Usage See `largefile(5)` for the description of the behavior of `mkfile` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed

See Also `chmod(1)`, `swap(1M)`, `attributes(5)`, `largefile(5)`

Name mkfs – construct a file system

Synopsis

```
mkfs [-F FSType] [generic_options]
     [-o FSType-specific_options] raw_device_file
     [operands]
```

Description The `mkfs` utility constructs a file system on the *raw_device_file* by calling the specific `mkfs` module indicated by `-F` *FSType*.

`Note`: ufs file systems are normally created with the `newfs(1M)` command.

generic_options are independent of file system type. *FSType-specific_options* is a comma-separated list of *keyword=value* pairs (with no intervening spaces), which are *FSType*-specific. *raw_device_file* specifies the disk partition on which to write the file system. It is required and must be the first argument following the *specific_options* (if any). *operands* are *FSType*-specific. See the *FSType*-specific manual page of `mkfs` (for example, `mkfs_ufs(1M)`) for a detailed description.

Options The following are the generic options for `mkfs`:

`-F` Specify the *FSType* to be constructed. If `-F` is not specified, the *FSType* is determined from `/etc/vfstab` by matching the *raw_device_file* with a `vfstab` entry, or by consulting the `/etc/default/fs` file.

`-V` Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided and adding to them information derived from `/etc/vfstab` or `/etc/default/fs`. This option may be used to verify and validate the command line.

`-m` Return the command line which was used to create the file system. The file system must already exist. This option provides a means of determining the command used in constructing the file system.

`-o` Specify *FSType*-specific options. See the manual page for the `mkfs` module specific to the file system type.

Usage See `largefile(5)` for the description of the behavior of `mkfs` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/default/fs` Default file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`

- `LOCAL` The default partition for a command if no *FSType* is specified.

`/etc/vfstab` List of default parameters for each file system

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also mkfs_ufs(1M), newfs(1M), vfstab(4), attributes(5), largefile(5)

Manual pages for the *FSType*-specific modules of mkfs.

Notes This command might not be supported for all *FSType*s.

You can use lofiadm to create a file that appears to a mkfs command as a raw device. You can then use a mkfs command to create a file system on that device. See lofiadm(1M) for examples of creating a UFS and a PC (FAT) file system (using mkfs_ufs(1M) and mkfs_pcfs(1M)) on a device created by lofiadm.

Name mkfs_pcfs – construct a FAT file system

Synopsis `mkfs -F pcfs [`*generic_options*`] [-o` *FSType_specific_options*`] `*raw_device_file*

Description The `pcfs`-specific module of `mkfs` constructs a File Allocation Table (FAT) on removable media (diskette, JAZ disk, ZIP disk, PCMCIA card), a hard disk, or a file (see NOTES). FATs are the standard MS-DOS and Windows file system format. Note that you can use `fdformat(1)` to construct a FAT file system only on a diskette or PCMCIA card.

`mkfs` for `pcfs` determines an appropriate FAT size for the medium, then it installs an initial boot sector and an empty FAT. A sector size of 512 bytes is used. `mkfs` for `pcfs` can also install the initial file in the file system (see the `pcfs`-specific `-o i` option). This first file can optionally be marked as read-only, system, and/or hidden.

If you want to construct a FAT with `mkfs` for `pcfs` on a medium that is not formatted, you must first perform a low-level format on the medium with `fdformat(1)` or `format(1M)`. Non-diskette media must also be partitioned with the `fdisk(1M)` utility. Note that all existing data on the diskette or disk partition, if any, is destroyed when a new FAT is constructed.

generic_options are supported by the generic `mkfs` command. See `mkfs(1M)` for a description of these options.

raw_device_file indicates the device on which to write unless the `-o N` option has been specified, or if the `-V` or `-m` generic options are passed from the generic `mkfs` module.

Options See `mkfs(1M)` for the list of supported generic options.

The following options are supported:

`-o` *FSType_specific_options* Specify `pcfs` file system-specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored.

- b=*label* Label the media with volume label. The volume label is restricted to 11 uppercase characters.
- B=*filename* Install *filename* as the boot loader in the file system's boot sector. If you don't specify a boot loader, an MS-DOS boot loader is installed. The MS-DOS boot loader requires specific MS-DOS system files to make the diskette bootable. See `NOTES` for more information.
- fat=*n* The size of a FAT entry. Currently, 12, 16, and 32 are valid values. The default is 12 for diskettes, 16 for larger media.

h	Mark the first file installed as a hidden file. The `-i` option must also be specified.
hidden=*n*	Set the number of hidden sectors to *n*. This is the number of sectors on the physical disk preceding the start of the volume (which is the boot sector itself). This defaults to 0 for diskettes or a computed valued (based on the fdisk table) for disks. This option may be used only in conjunction with the `nofdisk` option.
i=*filename*	Install *filename* as the initial file in the new file system. The initial file's contents are guaranteed to occupy consecutive clusters at the start of the files area. When creating bootable media, a boot program should be specified as the initial file.
nofdisk	Do not attempt to find an `fdisk` table on the medium. Instead rely on the `size` option for determining the partition size. By default, the created FAT is 16 bits and begins at the first sector of the device. This origination sector can be modified with the hidden option (`-h`).
nsect=*n*	The number of sectors per track on the disk. If not specified, the value is determined by using a `dkio(7I)` ioctl to get the disk geometry, or (for diskette) from the results of an `FDIOGCHAR` ioctl.
ntrack=*n*	The number of tracks per cylinder on the disk. If not specified, the value is determined by using a `dkio(7I)` ioctl to get the disk geometry, or (for diskette) from the results of an `FDIOGCHAR` ioctl.
N	No execution mode. Print normal output, but do not actually write the file system to the medium. This is most useful when used in conjunction with the verbose option.
r	Mark the first file installed as read-only. The `-i` option must also be specified.
reserve=*n*	Set the number of reserved sectors to *n*. This is the number of sectors in the volume,

preceding the start of the first FAT, including the boot sector. The value should always be at least 1, and the default value is exactly 1.

s	Mark the first file installed as a system file. The `-i` option must also be specified.
size=*n*	The number of sectors in the file system. If not specified, the value is determined from the size of the partition given in the fdisk table or (for diskette) by way of computation using the `FDIOGCHAR` ioctl.
spc=*n*	The size of the allocation unit for space within the file system, expressed as a number of sectors. The default value depends on the FAT entry size and the size of the file system.
v	Verbose output. Describe, in detail, operations being performed.

Files *raw_device_file* The device on which to build the FAT. The device name for a diskette must be specified as `/dev/rdiskette0` for the first diskette drive, or `/dev/rdiskette1` for a second diskette drive. For non-diskette media, a disk device name must be qualified with a suffix to indicate the proper partition. For example, in the name `/dev/rdsk/c0t0d0p0:c`, the `:c` suffix indicates that the first partition on the disk should receive the new FAT.

For a file, *raw_device_file* is the block device name returned by `lofiadm(1M)`.

Examples The media in these examples must be formatted before running `mkfs` for `pcfs`. See DESCRIPTION for more details.

EXAMPLE 1 Creating a FAT File System on a Diskette

The following command creates a FAT file system on a diskette:

```
mkfs -F pcfs /dev/rdiskette
```

EXAMPLE 2 Creating a FAT File System on a Disk

The following command creates a FAT file system on the second fdisk partition of a disk attached to an x86 based system:

```
mkfs -F pcfs /dev/rdsk/c0d0p0:d
```

EXAMPLE 3 Creating a FAT File System on a ZIP Disk

The following command creates a FAT file system on a ZIP disk located on a SPARC based system:

```
mkfs -F pcfs /dev/rdsk/c0t4d0s2:c
```

EXAMPLE 4 Creating a FAT File System on a JAZ Disk

The following command creates a FAT file system on a JAZ disk located on a SPARC based system and overrides the sectors/track and tracks/cylinder values obtained from the device's controller:

```
mkfs -F pcfs -o nsect=32,ntrack=64 /dev/rdsk/c0t3d0s2:c
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu
Interface Stability	Stable

See Also

`fdformat(1)`, `fdisk(1M)`, `format(1M)`, `lofiadm(1M)`, `mkfs(1M)`, `attributes(5)`, `fd(7D)`, `dkio(7I)`, `fdio(7I)`

Notes

The default MS-DOS boot loader, which is installed by default if `-o B` is not specified, requires specific MS-DOS system files to make the diskette bootable. These MS-DOS files are not installed when you format a diskette with `mkfs` for `pcfs`, which makes a diskette formatted this way not bootable. Trying to boot from it on an x86 based system will result in the following message:

```
Non-System disk or disk error
Replace and strike any key when ready
```

You must format a diskette with the DOS `format` command to install the specific MS-DOS system files required by the default boot loader.

You can use `lofiadm` to create a file that appears to a `mkfs` command (for example, `mkfs_pcfs` or `mkfs_ufs`) as a raw device. You can then use a `mkfs` command to create a file system on that device. See `lofiadm(1M)` for examples of creating a UFS and a PC (FAT) file system on a device created by `lofiadm`.

Name mkfs_udfs – construct a udfs file system

Synopsis

```
mkfs -F udfs [generic_options] [-o specific_options] raw_device_file
     [size]
```

Description This is the universal disk format file system (udfs) -specific module of the `mkfs` command. `mkfs` constructs a `udfs` file system with a root directory.

Options See `mkfs`(1M) for the list of supported *generic_options*.

The following options are supported:

`-o` *specific_options* Specify a `udfs`-specific option. Specify `udfs` file system specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored.

The following *specific_options* are available:

N — Print the file system parameters without actually creating the file system.

label=*string* — Specify the label to be written into the volume header structures. Specify *string* as the name of the label. If *string* is not specified, a default *string* is generated in the form of `*NoLabel*`.

Operands The following operands are supported:

raw_device_file Specify the disk partition on which to write.

size Specify the number of 512-byte blocks in the file system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWudf

See Also `fsck`(1M),`mkfs`(1M), `attributes(5)`

Diagnostics

`not currently a valid file system`

The specified device does not contain a valid `udfs` file system.

`Invalid size: larger than the partition size`

Number of blocks given as parameter to create the file system is larger than the size of the device specified.

`is mounted can't mkfs`

Device is in use, cannot create file system when the device is in use.

```
preposterous size
```

Negative size parameter provided is invalid.

```
sector size must be between 512, 8192 bytes
```

Sector size given is not in the valid range.

```
Volume integrity sequence descriptors too long
File set descriptor too long.
```

Not enough space to create volume integrity sequence or file set descriptor.

```
mkfs: argument out of range
```

One of the arguments is out of range.

```
mkfs: bad numeric arg
```

One of the arguments is potentially a bad numeric.

Notes You can use `lofiadm` to create a file that appears to a `mkfs` command (for example, `mkfs_pcfs` or `mkfs_ufs`) as a raw device. You can then use a `mkfs` command to create a file system on that device. See `lofiadm(1M)` for examples of creating a UFS and a PC (FAT) file system on a device created by `lofiadm`.

Name mkfs_ufs – construct a UFS file system

Synopsis

```
mkfs -F ufs [generic_options] [-o FSType_specific_options] raw_device_file
     [size]
```

Description The UFS-specific module of `mkfs` builds a UFS file system with a root directory and a `lost+found` directory (see `fsck`(1M)).

The UFS-specific `mkfs` is rarely run directly. Use the `newfs`(1M) command instead.

raw_device_file indicates the disk partition on which to create the new file system. If the `-o N`, `-V`, or `-m` options are specified, the *raw_device_file* is not actually modified. *size* specifies the number of disk sectors in the file system, where a disk sector is usually 512 bytes. This argument must follow the *raw_device_file* argument and is required (even with `-o N`), unless the `-V` or `-m` generic options are specified.

generic_options are supported by the generic `mkfs` command. See `mkfs`(1M) for a description of these options.

Options The following generic options are supported:

`-m` Print the command line that was used to create the existing file system.

`-V` Print the current `mkfs` command line.

Options The following UFS-specific options are supported:

`-o` Use one or more of the following values separated by commas (with no intervening spaces) to specify UFS-specific options:

`apc=`*n* The number of alternate sectors per cylinder to reserve for bad block replacement for SCSI devices only. The default is `0`.

This option is not applicable for disks with EFI labels and is ignored.

`bsize=`*n* The logical block size of the file system in bytes, either `4096` or `8192`. The default is `8192`. The sun4u architecture does not support the `4096` block size.

`calcbinsb` Sends to stdout a binary (machine-readable) version of the superblock that would be used to create a file system with the specified configuration parameters.

`calcsb` Sends to stdout a human-readable version of the superblock that would be used to create a file system with the specified configuration parameters.

`cgsize=`*n* The number of cylinders per cylinder group, ranging from `16` to `256`. The default is calculated by dividing the number of sectors in the file system by the number of sectors in a gigabyte. Then, the result is multiplied by `32`. The default value is always between `16` and `256`.

The per-cylinder-group meta data must fit in a space no larger than what is available in one logical file system block. If too large a `cgsize` is requested, it is changed by the minimum amount necessary.

`fragsize=`*n* The smallest amount of disk space in bytes that can be allocated to a file. *fragsize* must be a power of 2 divisor of *bsize*, where:

bsize / *fragsize* is 1, 2, 4, or 8.

This means that if the logical block size is `4096`, legal values for *fragsize* are `512`, `1024`, `2048`, and `4096`. When the logical block size is `8192`, legal values are `1024`, `2048`, `4096`, and `8192`. The default value is `1024`.

For file systems greater than 1 terabyte or for file systems created with the `mtb=y` option, *fragsize* is forced to match block size (*bsize*).

`free=`*n* The minimum percentage of free space to maintain in the file system between 0% and 99%, inclusively. This space is off-limits to users. Once the file system is filled to this threshold, only the superuser can continue writing to the file system.

The default is ((64 Mbytes/partition size) * 100), rounded down to the nearest integer and limited between 1% and 10%, inclusively.

This parameter can be subsequently changed using the `tunefs(1M)` command.

`gap=`*n* Rotational delay. This option is obsolete in the Solaris 10 release. The value is always set to `0`, regardless of the input value.

`maxcontig=`*n* The maximum number of logical blocks, belonging to one file, that are allocated contiguously. The default is calculated as follows:

`maxcontig =` *disk drive maximum transfer size / disk block size*

If the disk drive's maximum transfer size cannot be determined, the default value for `maxcontig` is calculated from kernel parameters as follows:

If `maxphys` is less than `ufs_maxmaxphys`, which is typically 1 Mbyte, then `maxcontig` is set to `maxphys`. Otherwise, `maxcontig` is set to `ufs_maxmaxphys`.

You can set `maxcontig` to any positive integer value.

The actual value will be the lesser of what has been specified and what the hardware supports.

You can subsequently change this parameter by using tunefs(1M).

`mtb=y` Set the parameters of the file system to allow eventual growth to over a terabyte in total file system size. This option sets *fragsize* to be the same as *bsize*, and sets *nbpi* to 1 Mbyte, unless the `-i` option is used to make it even larger. If you explicitly set the *fragsize* or *nbpi* parameters to values that are incompatible with this option, the user-supplied value of *fragsize* or *nbpi* is ignored.

`N` Print out the file system parameters that would be used to create the file system without actually creating the file system.

`nbpi=`*n* The number of bytes per inode, which specifies the density of inodes in the file system. The number is divided into the total size of the file system to determine the number of inodes to create.

This value should reflect the expected average size of files in the file system. If fewer inodes are desired, a larger number should be used. To create more inodes, a smaller number should be given. The default is `2048`.

The number of inodes can increase if the file system is expanded with the `growfs` command.

`nrpos=`*n* The number of different rotational positions in which to divide a cylinder group. The default is `8`.

This option is not applicable for disks with EFI labels and is ignored.

`nsect=`*n* The number of sectors per track on the disk. The default is `32`.

`ntrack=`*n* The number of tracks per cylinder on the disk. The default is `16`.

This option is not applicable for disks with EFI labels and is ignored.

`opt=`*s* | *t* The file system can either be instructed to try to minimize the *time* spent allocating blocks, or to try to minimize the *space* fragmentation on the disk. The default is *time*.

This parameter can be subsequently changed with the tunefs(1M) command.

`rps=`*n* The rotational speed of the disk, in revolutions per second. The default is `60`.

Note that you specify *rps* for `mkfs` and *rpm* for `newfs`.

This option is not applicable for disks with EFI labels and is ignored.

Alternatively, parameters can be entered as a list of space-separated values (without keywords) whose meaning is positional. In this case, the `-o` option is omitted and the list follows the size operand. This is the way `newfs` passes the parameters to `mkfs`.

Operands The following operands are supported:

raw_device_file The disk partition on which to write.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `fsck(1M)`, `mkfs(1M)`, `newfs(1M)`, `tunefs(1M)`, `dir_ufs(4)`, `attributes(5)`, `ufs(7FS)`

Diagnostics The following error message typically occurs with very high density disks. On such disks, the file system structure cannot encode the proper disk layout information. However, such disks have enough onboard intelligence to make up for any layout deficiencies, so there is no actual impact on performance. The warning that performance might be impaired can be safely ignored.

```
Warning: insufficient space in super block for
rotational layout tables with nsect sblock.fs_nsect
and ntrak sblock.fs_ntrak. (File system performance may be impaired.)
```

The following error message occurs when the disk geometry results in a situation where the last truncated cylinder group cannot contain the correct number of data blocks. Some disk space is wasted.

```
Warning: inode blocks/cyl group (grp) >= data blocks (num) in last cylinder
```

If there is only one cylinder group and if the above condition holds true, `mkfs` fails with the following error:

```
File system creation failed. There is only one cylinder group and that is
not even big enough to hold the inodes.
```

The following error message occurs when the best calculated file system layout is unable to include the last few sectors in the last cylinder group. This is due to the interaction between how much space is used for various pieces of meta data and the total blocks available in a cylinder group. Modifying `nbpi` and `cpg` might reduce this number, but it is rarely worth the effort.

```
Warning: num sector(s) in last cylinder group unallocated
```

Notes You can use `lofiadm` to create a file that appears to the `mkfs` command (for example, `mkfs_pcfs` or `mkfs_ufs`) as a raw device. You can then use the `mkfs` command to create a file system on that device. See `lofiadm(1M)` for examples of creating a UFS and a PC (FAT) file system on a device created by `lofiadm`.

Both the block and character devices, such as devices in `/dev/dsk` and `/dev/rdsk`, must be available prior to running the `mkfs` command.

Name mknod – make a special file

Synopsis

```
mknod name b major minor
mknod name c major minor
mknod name p
```

Description `mknod` makes a directory entry for a special file.

Options The following options are supported:

`b` Create a block-type special file.

`c` Create a character-type special file.

`p` Create a FIFO (named pipe).

Operands The following operands are supported:

major The *major* device number.

minor The *minor* device number; can be either decimal or octal. The assignment of major device numbers is specific to each system. You must be the super-user to use this form of the command.

name A special file to be created.

Usage See `largefile(5)` for the description of the behavior of `mknod` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `ftp(1)`, `in.ftpd(1M)`, `mknod(2)`, `symlink(2)`, `attributes(5)`, `largefile(5)`

Notes If `mknod(2)` is used to create a device, the major and minor device numbers are always interpreted by the kernel running on that machine.

With the advent of physical device naming, it would be preferable to create a symbolic link to the physical name of the device (in the `/devices` subtree) rather than using `mknod`.

Name mkpwdict – maintain password-strength checking database

Synopsis

```
/usr/sbin/mkpwdict [-s dict1,... ,dictN]
     [-d destination-path]
```

Description The `mkpwdict` command adds words to the dictionary-lookup database used by `pam_authtok_check(5)` and `passwd(1)`.

Files containing words to be added to the database can be specified on the command-line using the `-s` flag. These source files should have a single word per line, much like `/usr/share/lib/dict/words`.

If `-s` is omitted, `mkpwdict` will use the value of `DICTIONLIST` specified in `/etc/default/passwd` (see `passwd(1)`).

The database is created in the directory specified by the `-d` option. If this option is omitted, `mkpwdict` uses the value of `DICTIONDBDIR` specified in `/etc/default/passwd` (see `passwd(1)`). The default location is `/var/passwd`.

Options The following options are supported:

`-s` Specifies a comma-separated list of files containing words to be added to the dictionary-lookup database.

`-d` Specifies the target location of the dictionary-database.

Files

`/etc/default/passwd`	See `passwd(1)`.
`/var/passwd`	default destination directory

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also `passwd(1)`, `attributes(5)`, `pam_authtok_check(5)`

Name modinfo – display information about loaded kernel modules

Synopsis

```
/usr/sbin/modinfo [-c] [-w] [-i module-id]
```

Description The `modinfo` utility displays information about the loaded modules. The format of the information is as follows:

Id Loadaddr Size Info Rev Module Name

where *Id* is the module ID, *Loadaddr* is the starting text address in hexadecimal, *Size* is the size of text, data, and bss in hexadecimal bytes, *Info* is module specific information, *Rev* is the revision of the loadable modules system, and *Module Name* is the filename and description of the module.

The module specific information is the block and character major numbers for drivers, the system call number for system calls, and unspecified for other module types.

Options The following options are supported:

`-c` Display the number of instances of the module loaded and the module's current state.

`-i` *module-id* Display information about this module only.

`-w` Do not truncate module information at `80` characters.

Examples **EXAMPLE 1** Displaying the Status of a Module

The following example displays the status of module 2:

```
example% modinfo -i 2
Id   Loadaddr   Size Info Rev Module Name
 2    ff08e000   1734   -   1  swapgeneric (root and swap configuration)
```

EXAMPLE 2 Displaying the Status of Kernel Modules

The following example displays the status of some kernel modules:

```
example% modinfo
Id Loadaddr   Size Info Rev Module Name
 2  ff08e000   1734   -   1  swapgeneric
 4  ff07a000   3bc0   -   1  specfs (filesystem for specfs)
 6  ff07dbc0   2918   -   1  TS (time sharing sched class)
 7  ff0804d8    49c   -   1  TS_DPTBL (Time sharing dispatch table)
 8  ff04a000  24a30   2   1  ufs (filesystem for ufs)
 9  ff080978   c640 226   1  rpcmod (RPC syscall)
 9  ff080978   c640   -   1  rpcmod (rpc interface str mod)
10  ff08cfb8  2031c   -   1  ip (IP Streams module)
10  ff08cfb8  2031c   2   1  ip (IP Streams device)
```

EXAMPLE 3 Using the -c Option

Using the `modinfo` command with the -c option displays the number of instances of the module loaded and the module's current state.

```
example% modinfo -c
Id    Loadcnt Module Name                          State
  1          0 krtld                          UNLOADED/UNINSTALLED
  2          0 genunix                        UNLOADED/UNINSTALLED
  3          0 platmod                        UNLOADED/UNINSTALLED
  4          0 SUNW,UltraSPARC-IIi            UNLOADED/UNINSTALLED
  5          0 cl_bootstrap                   UNLOADED/UNINSTALLED
  6          1 specfs                         LOADED/INSTALLED
  7          1 swapgeneric                    UNLOADED/UNINSTALLED
  8          1 TS                             LOADED/INSTALLED
  9          1 TS_DPTBL                       LOADED/INSTALLED
 10          1 ufs                            LOADED/INSTALLED
 11          1 fssnap_if                      LOADED/INSTALLED
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also

`modload(1M)`, `modunload(1M)`, `attributes(5)`

Name modload – load a kernel module

Synopsis

```
modload [-p] [-e exec_file] filename
```

Description The `modload` command loads the loadable module *filename* into the running system.

filename is an object file produced by `ld -r`. If *filename* is an absolute pathname then the file specified by that absolute path is loaded. If *filename* does not begin with a slash (`/`), then the path to load *filename* is relative to the current directory unless the `-p` option is specified.

The kernel's `modpath` variable can be set using the `/etc/system` file. The default value of the kernel's `modpath` variable is set to the path where the operating system was loaded. Typically this is `/kernel /usr/kernel`.

For example, the following command looks for `./drv/foo`:

```
example# modload drv/foo
```

The following command looks for `/kernel/drv/foo` and then `/usr/kernel/drv/foo`:

```
example# modload -p drv/foo
```

Options The following options are supported:

`-e` *exec_file* — Specify the name of a shell script or executable image file that is executed after the module is successfully loaded. The first argument passed is the module ID (in decimal). The other argument is module specific. The module specific information is: the block and character major numbers for drivers, the system call number for system calls, or, for other module types, the index into the appropriate kernel table. See `modinfo(1M)`

`-p` — Use the kernel's internal modpath variable as the search path for the module.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `ld(1)`, `add_drv(1M)`, `kernel(1M)`, `modinfo(1M)`, `modunload(1M)`, `system(4)`, `attributes(5)`, `modldrv(9S)`, `modlinkage(9S)`, `modlstrmod(9S)`, `module_info(9S)`

Writing Device Drivers

Notes Use `add_drv(1M)` to add device drivers, not `modload`. See *Writing Device Drivers* for procedures on adding device drivers.

Name modunload – unload a module

Synopsis

```
modunload -i module_id [-e exec_file]
```

Description `modunload` unloads a loadable module from the running system. The *module_id* is the ID of the module as shown by `modinfo(1M)`. If `ID` is `0`, all modules that were autoloaded which are unloadable, are unloaded. Modules loaded by `modload(1M)` are not affected.

Options The following options are supported:

`-e` *exec_file* Specify the name of a shell script or executable image file to be executed before the module is unloaded. The first argument passed is the module id (in decimal). There are two additional arguments that are module specific. For loadable drivers, the second argument is the driver major number. For loadable system calls, the second argument is the system call number. For loadable exec classes, the second argument is the index into the `execsw` table. For loadable filesystems, the second argument is the index into the `vfssw` table. For loadable streams modules, the second argument is the index into the `fmodsw` table. For loadable scheduling classes, the second argument is the index into the class array. Minus one is passed for an argument that does not apply.

`-i` *module_id* Specify the module to be unloaded.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `modinfo(1M)`, `modload(1M)`, `update_drv(1M)`, `attributes(5)`

Notes The `modunload` command is often used on driver modules to force the system to reread the associated driver configuration file. While this works in the current Solaris release, it is not the supported way to reread the configuration file and is not guaranteed to work in future releases. The supported way for rereading driver configuration file is through the `update_drv(1M)` command.

Name mofcomp – compile MOF files into CIM classes

Synopsis

```
/usr/sadm/bin/mofcomp [-c cimom_hostname ] [-h]
     [-j filename] [-n namespace] [-o dirname]
     [-p password ] [-CIQ] [-u username] [-v ] [-version]
     [-x] file
```

Description The `mofcomp` utility is executed during installation to compile managed object format (MOF) files that describe the Common Information Model (CIM) and Solaris Schemas into the CIM Object Manager Repository, a central storage area for management data. The CIM Schema is a collection of class definitions used to represent managed objects that occur in every management environment. The Solaris Schema is a collection of class definitions that extend the CIM Schema and represent managed objects in a typical Solaris operating environment.

The `mofcomp` utility must be run as root or as a user with write access to the namespace in which you are compiling.

MOF is a language for defining CIM classes and instances. MOF files are ASCII text files that use the MOF language to describe CIM objects. A CIM object is a computer representation or model of a managed resource, such as a printer, disk drive, or CPU.

Many sites store information about managed resources in MOF files. Because MOF can be converted to Java, Java applications that can run on any system with a Java Virtual Machine can interpret and exchange this information. You can also use the `mofcomp` utility to compile MOF files at any time after installation.

Options The following options are supported:

`-c` *cimom_hostname* — Specify a remote system running the CIM Object Manager.

`-C` — Run the compiler set with the class option, which updates a class if it exists, and returns an error if the class does not exist. If you do not specify this option, the compiler adds a CIM class to the connected namespace, and returns an error if the class already exists.

`-h` — List the arguments to the `mofcomp` utility.

`-I` — Run the compiler set with the instance option, which updates an instance if it exists, and returns an error if the instance does not exist. If you do not specify this option, the compiler adds a CIM instance to the connected namespace, and returns an error if the instance already exists.

`-j` *filename* — Generate Java Beans and Java Interfaces to manage the CIM instances related to the CIM classes in the MOF being compiled.

The contents of *filename* are:

`PACKAGE=`*Java package name*
`IMPORTS=`*import1*:...:*importN*

`<EXCEPTIONS=`*exception1*`:...:`*exceptionN*

`PACKAGE` is a valid Java package name to include in all generated Java source. `IMPORTS` is an optional colon separated list of valid Java classes to be imported in all generated Java source. `EXCEPTIONS` is an optional colon separated list of valid Java exceptions to be thrown by the methods in all generated Java source.

`-n` *namespace*
: Requests that the compiler load the MOF file into the namespace specified as *namespace*. The default namespace (`root\cimv2`) is used unless this switch is used or a `#pragma` *namespace* ("`namespace`") statement appears in the MOF file. If both the `-n` *namespace* switch and the `#pragma` *namespace* construct are used, all namespaces are created, but the objects are created only in the `#pragma namespaces`.

`-o` *dirname*
: Run compiler in standalone mode, without the CIM Object Manager. Specify dirname as the directory in which the compiler output is to be stored. In this mode, the CIM Object Manager need not be running.

`-p` *password*
: Specify a password for connecting to the CIM Object Manager. Use this option for compilations that require privileged access to the CIM Object Manager. If you specify both `-p` and `-u`, you must type the password on the command line, which can pose a security risk. A more secure way to specify a password is to specify `-u` but not `-p`, so that the compiler will prompt for the password.

`-Q`
: Run the compiler set with the qualifier types option, which updates a qualifier type if it exists, and returns an error if the qualifier type does not exist. If you do not specify this option, the compiler adds a CIM qualifier type to the connected namespace, and returns an error if the qualifier type already exists.

`-u` *username*
: Specify user name for connecting to the CIM Object Manager. Use this option for compilations that require privileged access to the CIM Object Manager. If you specify both `-p` and `-u`, you must type the password on the command line, which can pose a security risk. A more secure way to specify a password is to specify `-u` but not `-p`, so that the compiler will prompt for the password.

`-v`
: Run the compiler in verbose mode, which displays compiler messages.

`-version`
: Display the version of the MOF compiler.

`-x`
: Generate XML documents for the CIM classes defined in the input MOF file.

Operands The following operands are supported:

file The pathname of the file to be compiled.

Exit Status The `mofcomp` utility exits with `0` upon success and a positive integer upon failure.

Files MOF files are installed in `/usr/sadm/mof`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbcor

See Also `init.wbem(1M)`, `mofreg(1M)`, `wbemadmin(1M)`, `wbemlogviewer(1M)`, `attributes(5)`, `wbem(5)`

Name mofreg – register MOF classes with WBEM services

Synopsis

```
/usr/sadm/bin/mofreg -r tag file

/usr/sadm/bin/mofreg -s

/usr/sadm/bin/mofreg -u tag [file]
```

Description The `mofreg` command is used by package and patch install scripts, or by any applications that wish to register managed object format (MOF) classes with Sun The Web-Based Enterprise Management (WBEM) services.

The WBEM services daemon (Common Information Model or CIM object manager) processes at start up the files that are specified by `mofreg` commands. Files are processed in the order that the individual `mofreg` commands are executed.

As an alternative to using the `mofreg` command, MOFs can be registered or unregistered by manipulating directories in `/var/sadm/wbem/logr`. Instead of running the `mofreg -r tag file` version fo the command you can create a directory named `tag` under `/var/sadm/wbem/logr/preReg` and copy `file` to the `tag` directory.

Similarly, instead of running the `mofreg -u tag [file]` command, you can create a directory named `tag` under `/var/sadm/wbem/logr/preUnreg` and copy the optional `file` to the `tag` directory.

The entries are processed in increasing order of last modification time of the `tag` directories. If you issue `mofreg` commands in rapid succession, the timestamps might be the same. If you have a situation where the timestamp order is critical, you can place appropriate sleeps between the successive registration or unregistration operations. As with the `mofreg` command, processing is done at next restart or by using the `-s` option.

This alternative mechanism is typically used in package install scripts which do not have access to `/usr`, and therefore do not have access to the `mofreg` command. This case arises when packages are installed for diskless clients.

Options The following options are supported:

`-r` *tag file* The *file* argument is the actual MOF registration file. Its form is identical to the MOF syntax as defined by the Distributed Management Task Force (DMTF). The only difference is the addition of the following 3 new pseudo-pragmas, which are variations of the namespace pragma. The name of file cannot end in `.unreg`.

```
#pragma namespace("__create")
#pragma namespace("__delete")
#pragma namespace("__modify")
```

These three pragmas are used specify if the elements following the pragmas should be created, deleted, or modified by the CIM object manager. The `__delete` pragma can currently only be applied for a `mofreg -u` command.

The *tag* argument is a unique string that specifies the identity of the registry action. This tag can be set to the package name or the patch number if the `mofreg` script is being invoked through packages/patches, though any tag can be specified.

Errors and warnings that are encountered when the CIM object manager handles the `mofreg` script are logged. Processing of the `mofreg` script stops at the first error. Specific warnings include:

```
Element already defined - the element already exists and
cannot be created.

Element not found - the element does not exist and
cannot be modified.
```

The error conditions are:

```
Key modification - A class cannot be modified if its keys
are being changed.

Other mod compilation errors.
```

`-s` Forces the CIM object manager to immediately process outstanding registry requests, instead of at the next restart. This currently requires Java.

`-u` *tag* [*file*] Undoes the operations performed during `mof` registry.

The *tag* argument must correspond to the value set during the original `mofreg` invocation. If no `mofreg` was done with the original *tag*, the command does not succeed.

If required, an *unreg* file can be specified. If no *unreg* file is specified, the CIM object manager automatically undoes the actions of the registry. Any class created by the registry process is removed and any classes modified by the registry revert to the old state.

The `mofreg` command does not take care of cases where packages and patches make conflicting changes to classes. This should be taken care of by the standard patch and package conflict resolution.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred. The reason for error is displayed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbcou

See Also `init.wbem(1M)`, `mofcomp(1M)`, `wbemadmin(1M)`, `wbemlogviewer(1M)`, `attributes(5)`, `wbem(5)`

Name monitor – SPARC system PROM monitor

Synopsis

```
STOP–A

BREAK

initial system power-on

exit from a client program, e.g., the Operating System
```

Description The CPU board of a workstation contains one or more EPROMs or EEPROMs. The program which executes from the PROMs is referred to as "the monitor". Among other things, the monitor performs system initialization at power-on and provides a user interface.

Monitor Prompt The monitor of earlier workstations was known as the `SunMON` monitor and displayed the > for its prompt. See the `SunMON MONITOR USAGE` section for further details.

Existing workstations use a monitor which is known as the OpenBoot monitor. The OpenBoot monitor typically displays `ok` as its prompt, but it may also display the > prompt under certain circumstances.

If the `'auto-boot?'` NVRAM parameter is set to `'false'` when the workstation is powered on, the system does not attempt to boot and the monitor issues its prompt. If `'auto-boot'` is set to `'true'`, the system initiates the boot sequence. The boot sequence can be aborted by simultaneously pressing two keys on the system's keyboard: L1 and A (on older keyboards), or Stop and A (on newer keyboards). Either a lower case `a` or an upper case `A` works for the keyboard abort sequence. If a console has been attached by way of one of the system's serial ports then the abort sequence can be accomplished by sending a BREAK. See `tip(1)`.

When the NVRAM `'security-mode'` parameter has been turned on, or when the value of the `'sunmon-compat?'` parameter is true, then the OpenBoot monitor displays the message: `Type b (boot), c (continue), or n (new command mode)`

and the > prompt appears.

Openboot Prom Usage Some of the more useful commands that can be issued from OpenBoot's `ok` prompt are described here. Refer to the *OpenBoot 2.x Command Reference Manual* book for a complete list of commands.

Help Help for various functional areas of the OpenBoot monitor can be obtained by typing `help`. The help listing provides a number of other key words which can then be used in the help command to provide further details.

NVRAM Parameters Each workstation contains one or more NVRAM devices which contains unique system ID information, as well as a set of user-configurable parameters. The NVRAM parameters allow the user a certain level of flexibility in configuring the system to act in a given manner under a specific set of circumstances.

See `eeprom(1M)` for a description of the parameters and information regarding setting the parameters from the OS level.

The following commands can be used at the OpenBoot monitor to access the NVRAM parameters.

`printenv`	Used to list the NVRAM parameters, along with their default values and current values.
`setenv` *pn pv*	Used to set or modify a parameter. The *pn* represents the parameter name, and *pv* represents the parameter value.
`set-default` *pn*	Used to set an individual parameter back to its default value.
`set-defaults`	Used to reset all parameters to their default values. (Note that `'set-defaults'` only affects parameters that have assigned default values.)

Security Parameters

Newer OpenBoot monitors contain user interfaces that support the storage and listing of keys for later use by client programs.

`list-security-keys`	Lists the names of keys currently stored on a machine.
`set-security-key` *keyname* [*keydata*]	Stores key data *keydata* in a key named *keyname*. Actual key data can be up to 32 bytes in length. The maximum length of *keyname* is 64 bytes, which allows for the hex-formatted ASCII used to present the key data. If *keydata* is not present, *keyname* and its corresponding data is deleted.

Hardware Checks and Diagnostics

The following commands are available for testing or checking the system's hardware. If the `'diag-switch?'` NVRAM parameter is set to true when the system is powered on, then a Power-On Self Test (POST) diagnostic is run, if present, sending its results messages to the system's serial port A. Not all of the commands shown are available on all workstations.

`test-all`	Run the diagnostic tests on each device which has provided a self-test.
`test floppy`	Run diagnostics on the system's floppy device.
`test /memory`	Run the main memory tests. If the NVRAM parameter `'diag-switch?'` is set to true, then all of main memory is tested. If the parameter is false then only the amount of memory specified in the `'selftest-#megs'` NVRAM parameter is tested.
`test net`	Test the network connection for the on-board network controller.
`watch-net`	Monitor the network attached to the on-board net controller.
`watch-net-all`	Monitor the network attached to the on-board net controller, as well as the network controllers installed in SBus slots.
`watch-clock`	Test the system's clock function.

System Information

The following commands are available for displaying information about the system. Not all commands are available on all workstations.

`banner`	Display the power-on banner.
`.enet-addr`	Display the system's Ethernet address.
`.idprom`	Display the formatted contents of the IDPROM.
`module-info`	Display information about the system's processor(s).
`probe-scsi`	Identify the devices attached to the on-board SCSI controller.
`probe-scsi-all`	Identify the devices attached to the on-board SCSI controller as well as those devices which are attached to SBus SCSI controllers.
`show-disks`	Display a list of the device paths for installed SCSI disk controllers.
`show-displays`	Display a list of the device paths for installed display devices.
`show-nets`	Display a list of the device paths for installed Ethernet controllers.
`show-sbus`	Display list of installed SBus devices.
`show-tapes`	Display a list of the device paths for installed SCSI tape controllers.
`show-ttys`	Display a list of the device paths for tty devices.
`.traps`	Display a list of the SPARC trap types.
`.version`	Display the version and date of the OpenBoot PROM.

Emergency Commands

These commands must be typed from the keyboard, they do not work from a console which is attached by way of the serial ports. With the exception of the Stop-A command, these commands are issued by pressing and holding down the indicated keys on the keyboard immediately after the system has been powered on. The keys must be held down until the monitor has checked their status. The Stop-A command can be issued at any time after the console display begins, and the keys do not need to be held down once they've been pressed. The Stop-D, Stop-F and Stop-N commands are not allowed when one of the security modes has been set. Not all commands are available on all workstations.

Stop (L1)	Bypass the Power-On Self Test (POST). This is only effective if the system has been placed into the diagnostic mode.
Stop-A (L1-A)	Abort the current operation and return to the monitor's default prompt.
Stop-D (L1-D)	Set the system's `'diag-switch?'` NVRAM parameter to `'true'`, which places the system in diagnostic mode. POST diagnostics, if present, are run, and the messages are displayed by way of the system's serial port A.
Stop-F (L1-F)	Enter the OpenBoot monitor before the monitor has probed the system for devices. Issue the `'fexit'` command to continue with system initialization.

Stop-N (L1-N) Causes the NVRAM parameters to be reset to their default values. Note that not all parameters have default values.

Line Editor Commands

The following commands can be used while the monitor is displaying the `ok` prompt. Not all of these editing commands are available on all workstations.

CTRL-A	Place the cursor at the start of line.
CTRL-B	Move the cursor backward one character.
ESC-B	Move the cursor backward one word.
CTRL-D	Erase the character that the cursor is currently highlighting.
ESC-D	Erase the portion of word from the cursor's present position to the end of the word.
CTRL-E	Place the cursor at the end of line.
CTRL-F	Move the cursor forward one character.
ESC-F	Move the cursor forward one word.
CTRL-H	Erase the character preceding the cursor (also use Delete or Back Space)
ESC-H	Erase the portion of the word which precedes the cursor (use also CTRL-W)
CTRL-K	Erase from the cursor's present position to the end of the line.
CTRL-L	Show the command history list.
CTRL-N	Recall the next command from the command history list
CTRL-P	Recall a previous command from the command history list.
CTRL-Q	Quote the next character (used to type a control character).
CTRL-R	Retype the current line.
CTRL-U	Erase from the cursor's present position to the beginning of the line.
CTRL-Y	Insert the contents of the memory buffer into the line, in front (to the left) of the cursor.

nvramrc

The `nvramrc` is an area of the system's NVRAM where users may store Forth programs. The programs which are stored in the `nvramrc` are executed each time the system is reset, provided that the `'use-nvramrc?'` NVRAM parameter has been set to `'true'`. Refer to the *OpenBoot 2.x Command Reference Manual*book for information on how to edit and use the `nvramrc`.

Restricted Monitor

The command `'old-mode'` is used to move OpenBoot into a restricted monitor mode, causing the > prompt to be displayed. Only three commands are allowed while in the restricted monitor; the `'go'` command (to resume a program which was interrupted with the

Stop-A command), the `'n'` command (to return to the normal OpenBoot monitor), and boot commands. The restricted monitor's boot commands approximate the older `SunMON` monitor's boot command syntax. If a `'security-mode'` has been turned on then the restricted monitor becomes the default monitor environment. The restricted monitor may also become the default environment if the `'sunmon-compat?'` NVRAM parameter is set to true. Not all workstations have the `'sunmon-compat?'` parameter.

SunMON Prom Usage

The following commands are available systems with older `SunMON`-based PROM:

`+|–`
Increment or decrement the current address and display the contents of the new location.

`^C` *source destination n*
(caret-C) Copy, byte-by-byte, a block of length *n* from the `source` address to the *destination* address.

`^I` *program*
(caret-I) Display the compilation date and location of *program*.

`^T` *virtual_address*
(caret-T) Display the physical address to which *virtual_address* is mapped.

`b` [`!`] [*device* [`(`*c*`,`*u*`,`*p* `)`]] [*pathname*] [*arguments_list*]

`b[?]`
Reset appropriate parts of the system and bootstrap a program. A '`!`' (preceding the *device* argument) prevents the system reset from occurring. Programs can be loaded from various devices (such as a disk, tape, or Ethernet). '`b`' with no arguments causes a default boot, either from a disk, or from an Ethernet controller. '`b?`' displays all boot devices and their *device*s.

device one of

`le`	Lance Ethernet
`ie`	Intel Ethernet
`sd`	SCSI disk, CDROM
`st`	SCSI 1/4" or 1/2" tape
`fd`	Diskette
`id`	IPI disk
`mt`	Tape Master 9-track 1/2" tape
`xd`	Xylogics 7053 disk
`xt`	Xylogics 1/2" tape
`xy`	Xylogics 440/450 disk

c A controller number (`0` if only one controller),

u A unit number (`0` if only one driver), and

p A partition.

pathname A pathname for a program such as `/stand/diag`.

arguments_list A list of up to seven arguments to pass to the program being booted.

`c` [*virtual_address*]
Resume execution of a program. When given, *virtual_address* is the address at which execution resumes. The default is the current PC. Registers are restored to the values shown by the `d`, and `r` commands.

`d` [*window_number*]
Display (dump) the state of the processor. The processor state is observable only after:

- An unexpected trap was encountered.
- A user program dropped into the monitor (by calling *abortent*).
- The user manually entered the monitor by typing L1–A or BREAK.

The display consists of the following:

- The special registers: PSR, PC, nPC, TBR, WIM, and Y
- Eight global registers
- 24 window registers (8 *in*, 8 *local*, and 8 *out*), corresponding to one of the 7 available windows. If a Floating-Point Unit is on board, its status register along with 32 floating-point registers are also shown.

window_number Display the indicated *window_number*, which can be any value between `0` and `6`, inclusive. If no window is specified and the PSR's current window pointer contains a valid window number, registers from the window that was active just prior to entry into the monitor are displayed. Otherwise, registers from window 0 are displayed.

`e` [*virtual_address*] [*action*] . . .
Open the 16-bit word at *virtual_address* (default zero). The address is interpreted in the address space defined by the `s` command. See the `a` command for a description of *action*.

`f` *virtual_address1 virtual_address2 pattern* [*size*]
Fill the bytes, words, or long words from *virtual_address1* (lower) to *virtual_address2* (higher) with the constant, *pattern*. The `size` argument can take one of the following values:

`b` byte format (the default)

`w` word format

`l` long word format

For example, the following command fills the address block from `0x1000` to `0x2000` with the word pattern, `0xABCD`:

```
f 1000 2000 ABCD W
```

`g` [*vector*] [*argument*]
`g` [*virtual_address*] [*argument*]
Goto (jump to) a predetermined or default routine (first form), or to a user-specified routine (second form). The value of *argument* is passed to the routine. If the *vector* or *virtual_address* argument is omitted, the value in the PC is used as the address to jump to.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's `g` command, set the variable `*romp->v_vector_cmd` to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

`%x` hexadecimal

`%d` decimal

`g0`
Force a panic and produce a crash dump when the monitor is running as a result of the system being interrupted,

`g4`
(Sun-4 systems only) Force a kernel stack trace when the monitor is running as a result of the system being interrupted,

`h`
Display the help menu for monitor commands and their descriptions. To return to the monitor's basic command level, press ESCAPE or `q` before pressing RETURN.

`i` [*cache_data_offset*] [*action*] . . .
Modify cache data RAM command. Display and/or modify one or more of the cache data addresses. See the `a` command for a description of *action*.

`j` [*cache_tag_offset*] [*action*] . . .
Modify cache tag RAM command. Display and/or modify the contents of one or more of the cache tag addresses. See the `a` command for a description of *action*.

`k` [*reset_level*]
Reset the system, where *reset_level* is:

`0` Reset VMEbus, interrupt registers, video monitor (Sun-4 systems). This is the default.

`1` Software reset.

2 Power-on reset. Resets and clears the memory. Runs the EPROM-based diagnostic self test, which can take several minutes, depending upon how much memory is being tested.

`kb`
Display the system banner.

`l` [*virtual_address*] [*action*] . . .
Open the long word (32 bit) at memory address *virtual_address* (default zero). The address is interpreted in the address space defined by the `s` command (below). See the `a` command for a description of *action*.

`m` [*virtual_address*] [*action*] . . .
Open the segment map entry that maps *virtual_address* (default zero). The address is interpreted in the address space defined by the `s` command. See the `a` command for a description of *action*.

`ne`

`ni`
Disable, enable, or invalidate the cache, respectively.

`o` [*virtual_address*] [action*]* . . .
Open the byte location specified by *virtual_address* (default zero). The address is interpreted in the address space defined by the `s` command. See the `a` command for a description of *action*.

`p` [*virtual_address*] [*action*]. . .
Open the page map entry that maps *virtual_address* (default zero) in the address space defined by the `s` command. See the `a` command for a description of *action*.

`q` [*eeprom_offset*] [*action*]. . .
Open the EEPROM *eeprom_offset* (default zero) in the EEPROM address space. All addresses are referenced from the beginning or base of the EEPROM in physical address space, and a limit check is performed to insure that no address beyond the EEPROM physical space is accessed. This command is used to display or modify configuration parameters, such as: the amount of memory to test during self test, whether to display a standard or custom banner, if a serial port (A or B) is to be the system console, etc. See the `a` command for a description of *action*.

`r` [*register_number*]
`r` [*register_type*]
`r` [*w window_number*]
Display and/or modify one or more of the IU or FPU registers. A hexadecimal *register_number* can be one of:

`0x00`–`0x0f`	window(0,i0)–window(0,i7), window(0,i0)—window(0,i7)
`0x16`–`0x1f`	window(1,i0)–window(1,i7), window(1,i0)—window(1,i7)

`0x20–0x2f`	window(2,i0)–window(2,i7), window(2,i0)—window(2,i7)
`0x30–0x3f`	window(3,i0)–window(3,i7), window(3,i0)—window(3,i7)
`0x40–0x4f`	window(4,i0)–window(4,i7), window(4,i0)—window(4,i7)
`0x50–0x5f`	window(5,i0)–window(5,i7), window(5,i0)—window(5,i7)
`0x60–0x6f`	window(6,i0)–window(6,i7), window(6,i0)—window(6,i7)
`0x70–0x77`	`g0, g1, g2, g3, g4, g5, g6, g7`
`0x78–0x7d`	PSR, PC, nPC, WIM, TBR, Y.
`0x7e–0x9e`	FSR, f0–f31

Register numbers can only be displayed after an unexpected trap, a user program has entered the monitor using the *abortent* function, or the user has entered the monitor by manually typing L1–A or BREAK.

If a *register_type* is given, the first register of the indicated type is displayed. *register_type* can be one of:

`f`	floating-point
`g`	global
`s`	special

If `w` and a *window_number* (`0`—`6`) are given, the first *in*-register within the indicated window is displayed. If *window_number* is omitted, the window that was active just prior to entering the monitor is used. If the PSR's current window pointer is invalid, window 0 is used.

`s [`*asi*`])`

Set or display the Address Space Identifier. With no argument, `s` displays the current Address Space Identifier. The *asi* value can be one of:

`0x2`	control space
`0x3`	segment table
`0x4`	Page table
`0x8`	user instruction
`0x9`	supervisor instruction
`0xa`	user data
`0xb`	supervisor data
`0xc`	flush segment

`0xd` flush page

`0xe` flush context

`0xf` cache data

`u` [`echo`]

`u` [*port*] [*options*] [*baud_rate*]

`u` [`u`] [*virtual_address*]
With no arguments, display the current I/O device characteristics including: current input device, current output device, baud rates for serial ports A and B, an input-to-output echo indicator, and virtual addresses of mapped UART devices. With arguments, set or configure the current I/O device. With the `u` argument (`uu. . .`), set the I/O device to be the *virtual_address* of a UART device currently mapped.

`echo` Can be either `e` to enable input to be echoed to the output device, or `ne`, to indicate that input is not echoed.

port Assign the indicated *port* to be the current I/O device. *port* can be one of:

`a` serial port A

`b` serial port B

`k` the workstation keyboard

`s` the workstation screen

baud_rate Any legal baud rate.

options can be any combination of:

`i` input

`o` output

`u` UART

`e` echo input to output

`ne` do not echo input

`r` reset indicated serial port (`a` and `b` ports only)

If either `a` or `b` is supplied, and no *options* are given, the serial port is assigned for both input and output. If `k` is supplied with no options, it is assigned for input only. If `s` is supplied with no options, it is assigned for output only.

`v` *virtual_address1 virtual_address2* [`size`]
Display the contents of *virtual_address1* (lower) *virtual_address2* (higher) in the format specified by `size`:

`b` byte format (the default)

`w` word format

`l` long word format

Enter return to pause for viewing; enter another return character to resume the display. To terminate the display at any time, press the space bar.

For example, the following command displays the contents of virtual address space from address `0x1000` to `0x2000` in word format:

```
v 1000 2000 W
```

`w` [*virtual_address*] [*argument*]
Set the execution vector to a predetermined or default routine. Pass *virtual_address* and *argument* to that routine.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's `w` command, set the variable `*romp->v_vector_cmd` to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

`%x` hexadecimal

`%d` decimal

`x`
Display a menu of extended tests. These diagnostics permit additional testing of such things as the I/O port connectors, video memory, workstation memory and keyboard, and boot device paths.

`y c` *context_number*

`y p|s` *context_number virtual_address*
Flush the indicated context, context page, or context segment.

`c` flush context *context_number*

`p` flush the page beginning at *virtual_address* within context *context_number*

`s` flush the segment beginning at *virtual_address* within context *context_number*

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC

See Also `tip`(1), `boot`(1M), `eeprom`(1M), `attributes`(5)

OpenBoot 2.x Command Reference Manual

Name mount, umount – mount or unmount file systems and remote resources

Synopsis

```
mount [-p | -v]

mount [-F FSType] [generic_options] [-o specific_options]
     [-O] special | mount_point

mount [-F FSType] [generic_options] [-o specific_options]
     [-O] special mount_point

mount -a [-F FSType] [-V] [current_options]
     [-o specific_options] [mount_point]...

umount [-f] [-V] [-o specific_options] special | mount_point

umount -a [-f] [-V] [-o specific_options] [mount_point]...
```

Description `mount` attaches a file system to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the `mount` operation, these are hidden until the file system is unmounted.

`umount` unmounts a currently mounted file system, which may be specified either as a *mount_point* or as *special*, the device on which the file system resides.

The table of currently mounted file systems can be found by examining the mounted file system information file. This is provided by a file system that is usually mounted on `/etc/mnttab`. The mounted file system information is described in `mnttab(4)`. Mounting a file system adds an entry to the mount table; a `umount` removes an entry from the table.

When invoked with both the *special* and *mount_point* arguments and the `-F` option, `mount` validates all arguments except for *special* and invokes the appropriate *FSType*-specific `mount` module. If invoked with no arguments, `mount` lists all the mounted file systems recorded in the mount table, `/etc/mnttab`. If invoked with a partial argument list (with only one of *special* or *mount_point*, or with both *special* or *mount_point* specified but not *FSType*), `mount` will search `/etc/vfstab` for an entry that will supply the missing arguments. If no entry is found, and the special argument starts with "/", the default local file system type specified in `/etc/default/fs` will be used. Otherwise the default remote file system type will be used. The default remote file system type is determined by the first entry in the `/etc/dfs/fstypes` file. After filling in missing arguments, `mount` will invoke the *FSType*-specific `mount` module.

Only a super-user can mount or unmount file systems using `mount` and `umount`. However, any user can use `mount` to list mounted file systems and resources.

Options

`-F` *FSType*
: Used to specify the *FSType* on which to operate. The *FSType* must be specified or must be determinable from `/etc/vfstab`, or by consulting `/etc/default/fs` or `/etc/dfs/fstypes`.

`-a` [*mount_points*...]
: Perform mount or umount operations in parallel, when possible.

If mount points are not specified, `mount` will mount all file systems whose `/etc/vfstab` "mount at boot" field is "yes". If mount points are specified, then `/etc/vfstab` "mount at boot" field will be ignored.

If mount points are specified, `umount` will only `umount` those mount points. If none is specified, then `umount` will attempt to `unmount` all file systems in `/etc/mnttab`, with the exception of certain system required file systems: `/`, `/usr`, `/var`, `/var/adm`, `/var/run`, `/proc`, `/dev/fd` and `/tmp`.

`-f`
Forcibly unmount a file system.

Without this option, `umount` does not allow a file system to be unmounted if a file on the file system is busy. Using this option can cause data loss for open files; programs which access files after the file system has been unmounted will get an error (`EIO`).

`-p`
Print the list of mounted file systems in the `/etc/vfstab` format. Must be the only option specified. See `BUGS`.

`-v`
Print the list of mounted file systems in verbose format. Must be the only option specified.

`-V`
Echo the complete command line, but do not execute the command. `umount` generates a command line by using the options and arguments provided by the user and adding to them information derived from `/etc/mnttab`. This option should be used to verify and validate the command line.

generic_options
Options that are commonly supported by most *FSType*-specific command modules. The following options are available:

`-m`
Mount the file system without making an entry in `/etc/mnttab`.

`-g`
Globally mount the file system. On a clustered system, this globally mounts the file system on all nodes of the cluster. On a non-clustered system this has no effect.

`-o`
Specify *FSType*-specific options in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the *FSType*-specific module of the command. (See `mount_ufs(1M)`.) When you use `-o` with a file system that has an entry in `/etc/vfstab`, any `mount` options entered for that file system in `/etc/vfstab` are ignored.

The following options are supported:

`devices | nodevices`
Allow or disallow the opening of device-special files. The default is `devices`.

If you use `nosuid` in conjunction with `devices`, the behavior is equivalent to that of `nosuid`.

`exec | noexec`
Allow or disallow executing programs in the file system. Allow or disallow `mmap(2)` with `PROT_EXEC` for files within the file system. The default is `exec`.

`nbmand | nonbmand`
Allow or disallow non-blocking mandatory locking semantics on this file system. Non-blocking mandatory locking is disallowed by default.

If the file system is mounted with the `nbmand` option, then applications can use the `fcntl(2)` interface to place non-blocking mandatory locks on files and the system enforces those semantics. If you enable this option, it can cause standards conformant applications to see unexpected errors.

Do not use the `nbmand` option with `/`, `/var` and `/usr`.

You should not use the `remount` option to change the `nbmand` disposition of the file system. The `nbmand` option is mutually exclusive of the global option. See `-g`.

`ro | rw`
Specify read-only or read-write. The default is `rw`.

`setuid | nosetuid`
Allow or disallow `setuid` or `setgid` execution. The default is `setuid`.

If you specify `setuid` in conjunction with `nosuid`, the behavior is the same as `nosuid`.

`nosuid` is equivalent to `nosetuid` and `nodevices`. When `suid` or `nosuid` is combined with `setuid` or `nosetuid` and `devices` or `nodevices`, the most restrictive options take effect.

This option is highly recommended whenever the file system is shared by way of NFS with the `root=` option. Without it, NFS clients could add `setuid` programs to the server or create devices that could open security holes.

`suid | nosuid`
Allow or disallow `setuid` or `setgid` execution. The default is `suid`. This option also allows or disallows opening any device-special entries that appear within the filesystem.

`nosuid` is equivalent to `nosetuid` and `nodevices`. When `suid` or `nosuid` is combined with `setuid` or `nosetuid` and `devices` or `nodevices`, the most restrictive options take effect.

This option is highly recommended whenever the file system is shared using NFS with the `root=`*option*, because, without it, NFS clients could add `setuid` programs to the server, or create devices that could open security holes.

`-O`
Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error "device busy".

`-r`
Mount the file system read-only.

Usage See `largefile(5)` for the description of the behavior of `mount` and `umount` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files `/etc/mnttab`
Table of mounted file systems.

`/etc/default/fs`
Default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`

`LOCAL:`
The default partition for a command if no *FSType* is specified.

`/etc/vfstab`
List of default parameters for each file system.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `mount_cachefs(1M)`, `mount_hsfs(1M)`, `mount_nfs(1M)`, `mount_pcfs(1M)`, `mount_tmpfs(1M)`, `mount_ufs(1M)`, `mountall(1M)`, `umountall(1M)`, `fcntl(2)`, `mmap(2)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`, `lofs(7FS)`, `pcfs(7FS)`

Notes If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Bugs The `mount -p` output is incorrect for `cachefs`.

Name mountall, umountall – mount, unmount multiple file systems

Synopsis

```
mountall [-F FSType] [-l | -r] [file_system_table]

umountall [-k] [-s] [-F FSType] [-l | -r] [-n]  [-Z]

umountall [-k] [-s] [-h host] [-n] [-Z]
```

Description `mountall` is used to mount file systems specified in a file system table. The file system table must be in `vfstab(4)` format. If no *file_system_table* is specified, `/etc/vfstab` is used. If – is specified as *file_system_table*, `mountall` reads the file system table from the standard input. `mountall` mounts only those file systems with the `mount at boot` field set to `yes` in the *file_system_table*.

For each file system in the file system table, the following logic is executed: if there exists a file`/usr/lib/fs/`*FSType*`/fsckall`, where *FSType* is the type of the file system, save that file system in a list to be passed later, and all at once, as arguments to the `/usr/lib/fs/`*FSType*`/fsckall` script. The `/usr/lib/fs/`*FSType*`/fsckall` script checks all of the file systems in its argument list to determine whether they can be safely mounted. If no `/usr/lib/fs/`*FSType*`/fsckall` script exists for the *FSType* of the file system, the file system is individually checked using `fsck(1M)`. If the file system does not appear mountable, it is fixed using `fsck` before the mount is attempted. File systems with a – entry in the `fsckdev` field are mounted without first being checked.

`umountall` causes all mounted file systems in the current zone except `root`, `/usr`, `/var`, `/var/adm`, `/var/run`, `/proc`, and `/dev/fd` to be unmounted. If the *FSType* is specified, `mountall` and `umountall` limit their actions to the *FSType* specified. There is no guarantee that `umountall` unmounts *busy* file systems, even if the `-k` option is specified.

Options The following options are supported:

`-F` Specify the *FSType* of the file system to be mounted or unmounted.

`-h` *host* Unmount all file systems listed in `/etc/mnttab` that are remote-mounted from host.

`-k` Use the `fuser -k` *mount-point* command. See the `fuser(1M)` for details. The `-k` option sends the `SIGKILL` signal to each process using the file. As this option spawns kills for each process, the kill messages might not show up immediately. There is no guarantee that `umountall` unmounts *busy* file systems, even if the `-k` option is specified.

`-l` Limit the action to local file systems.

`-n` List the actions that would be performed for the specified options, but do not actually execute these actions. Repeating the command without the `-n` option executes the listed actions, assuming that the `/etc/mnttab` file has not changed in the interval prior to repeating the command.

`-r` Limit the action to remote file system types.

`-s` Do not perform the `umount` operation in parallel.

`-Z` Apply the action(s) only to the file systems mounted in non-global zones. By default, `umountall` unmounts only file systems mounted in the current zone. Option `-Z` is ignored if used in a non-global zone.

Files `/etc/mnttab`
Mounted file system table

`/etc/vfstab`
Table of file system defaults

`/usr/lib/fs/`*FSType*`/fsckall`
Script called by `mountall` to perform the file system check of all file systems of type *FSType*

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed
Output Stability	Uncommitted

See Also `fsck(1M)`, `fuser(1M)`, `mount(1M)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`

Diagnostics No messages are printed if the file systems are mountable and clean.

Error and warning messages come from `fsck(1M)` and `mount(1M)`.

Notes At this time, NFS is the only remote file system supported by the `-l`, `-r`, and `-h` options.

Name mount_cachefs – mount CacheFS file systems

Synopsis

```
mount -F cachefs [generic_options] -o backfstype=file_system_type
      [specific_options]
      [-O] special mount_point
```

Description The CacheFS-specific version of the `mount` command mounts a cached file system; if necessary, it NFS-mounts its back file system. It also provides a number of CacheFS-specific options for controlling the caching process. For more information regarding back file systems, refer to the *System Administration Guide: Basic Administration*.

`mount_cachefs` cannot be used with replicated NFS mounts. `mount_cachefs` creates a pass through when used with an NFS version 4 mount. No caching is performed.

Options To mount a CacheFS file system, use the generic `mount` command with the `-F` option followed by the argument `cachefs`.

See `mount(1M)` for a list of supported *generic_options*.

`-o` *specific_options* Specify CacheFS file system specific options in a comma-separated list with no intervening spaces.

`acdirmax=`*n* Specifies that cached attributes are held for no more than *n* seconds after directory update. After *n* seconds, all directory information is purged from the cache. The default value is `30` seconds.

`acdirmin=`*n* Specifies that cached attributes are held for at least *n* seconds after directory update. After *n* seconds, CacheFS checks to see if the directory modification time on the back file system has changed. If it has, all information about the directory is purged from the cache and new data is retrieved from the back file system. The default value is `30` seconds.

`acregmax=`*n* Specifies that cached attributes are held for no more than *n* seconds after file modification. After *n* seconds, all file information is purged from the cache. The default value is `30` seconds.

`acregmin=`*n*	Specifies that cached attributes are held for at least *n* seconds after file modification. After *n* seconds, CacheFS checks to see if the file modification time on the back file system has changed. If it has, all information about the file is purged from the cache and new data is retrieved from the back file system. The default value is `30` seconds.
`actimeo=`*n*	Sets `acregmin`, `acregmax`, `acdirmin`, and `acdirmax` to *n*.
`backfstype=`*file_system_type*	The file system type of the back file system (can be `nfs` or `hsfs`).
`backpath=`*path*	Specifies where the back file system is already mounted. If this argument is not supplied, CacheFS determines a mount point for the back file system. The back file system must be read-only.
`cachedir=`*directory*	The name of the cache directory.
`cacheid=`*ID*	*ID* is a string specifying a particular instance of a cache. If you do not specify a cache ID, CacheFS will construct one.
`demandconst`	Verifies cache consistency only when explicitly requested, rather than the periodic checking that is done by default. A consistency check is requested by using the `-s` option of the `cfsadmin(1M)` command. This option is useful for back file systems that change infrequently, for example, `/usr/openwin`. `demandconst` and `noconst` are mutually exclusive.
`local-access`	Causes the front file system to interpret the mode bits used for access checking instead of having the back file system verify access

permissions. Do not use this argument with secure NFS.

`noconst` — Disables cache consistency checking. By default, periodic consistency checking is enabled. Specify `noconst` only when you know that the back file system will not be modified. Trying to perform cache consistency check using `cfsadmin -s` will result in error. `demandconst` and `noconst` are mutually exclusive.

`write-around` | `non-shared` — Write modes for CacheFS. The `write-around` mode (the default) handles writes the same as NFS does; that is, writes are made to the back file system, and the affected file is purged from the cache. You can use the `non-shared` mode when you are sure that no one else will be writing to the cached file system. In this mode, all writes are made to both the front and the back file system, and the file remains in the cache.

`-O` — Overlay mount. Allows the filesystem to be mounted over an existing mount point, making the underlying filesystem inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, mount will fail with the error: `mount -F cachefs: mount failed Device busy`.

Examples

EXAMPLE 1 CacheFS-mounting a File System

The following example CacheFS-mounts the file system `server1:/user2`, which is already NFS-mounted on `/usr/abc` as `/xyz`.

```
example# mount -F cachefs -o backfstype=nfs,backpath=/usr/abc,
     cachedir=/cache1 server1:/user2 /xyz
```

The lines similar to the following appear in the `/etc/mnttab` file after the `mount` command is executed:

```
server1:/user2   /usr/abc        nfs
/usr/abc         /cache1/xyz    cachefs     backfstype=nfs
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `cfsadmin`(1M), `fsck_cachefs`(1M), `mount`(1M), `attributes`(5) *System Administration Guide: Basic Administration*

Bugs The output for the *generic_option* `-p` output is incorrect for `cachefs`.

Name mountd – server for NFS mount requests and NFS access checks

Synopsis

```
/usr/lib/nfs/mountd [-v] [-r]
```

Description `mountd` is an RPC server that answers requests for NFS access information and file system mount requests. It reads the file `/etc/dfs/sharetab` to determine which file systems are available for mounting by which remote machines. See `sharetab(4)`. `nfsd` running on the local server will contact `mountd` the first time an NFS client tries to access the file system to determine whether the client should get read-write, read-only, or no access. This access can be dependent on the security mode used in the remoted procedure call from the client. See `share_nfs(1M)`.

The command also provides information as to what file systems are mounted by which clients. This information can be printed using the `showmount(1M)` command.

The `mountd` daemon is automatically invoked by `share(1M)`.

Only super user can run the `mountd` daemon.

Options The options shown below are supported for NVSv2/v3 clients. They are not supported for Solaris NFSv4 clients.

`-r` Reject mount requests from clients. Clients that have file systems mounted will not be affected.

`-v` Run the command in verbose mode. Each time `mountd` determines what access a client should get, it will log the result to the console, as well as how it got that result.

Files `/etc/dfs/sharetab` shared file system table

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfssu

See Also `nfsd(1M)`, `share(1M)`, `share_nfs(1M)`, `showmount(1M)`, `nfs(4)`, `sharetab(4)`, `attributes(5)`

Notes Since `mountd` must be running for `nfsd` to function properly, `mountd` is automatically started by the `svc:/network/nfs/server` service. See `nfs(4)`.

Some routines that compare hostnames use case-sensitive string comparisons; some do not. If an incoming request fails, verify that the case of the hostname in the file to be parsed matches the case of the hostname called for, and attempt the request again.

Name mount_hsfs – mount hsfs file systems

Synopsis

```
mount -F hsfs [generic_options]
     [-o FSType-specific_options] [-O ] special | mount_point

mount -F hsfs [generic_options]
     [-o FSType-specific_options] [-O] special  mount_point
```

Description `mount` attaches a High Sierra file system (hsfs) to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the `mount` operation, these are hidden until the file system is unmounted.

If `mount` is invoked with *special* or *mount_point* as the only arguments, `mount` will search `/etc/vfstab` to fill in the missing arguments, including the *FSType-specific_options*; see `mount(1M)` for more details.

If the file system being mounted contains `Rock Ridge` extensions, by default they will be used, enabling support of features not normally available under High Sierra file systems such as symbolic links, and special files.

Options

generic_options — See `mount(1M)` for the list of supported options.

`-o` — Specify `hsfs` file system specific options. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

`global` | `noglobal` — If `global` is specified and supported on the file system, and the system in question is part of a cluster, the file system will be globally visible on all nodes of the cluster. If `noglobal` is specified, the mount will not be globally visible. The default behavior is `noglobal`.

`ro` — Mount the file system read-only. This option is required.

`nrr` — `no Rock Ridge`: if `Rock Ridge` extensions are present in the file system, ignore them; interpret it as a regular High Sierra file system.

`notraildot` — File names on High Sierra file systems consist of a proper name and an extension separated by a '.' (dot) character. By default, the separating dot is always considered part of the file's name for all file access operations, even if there is no extension present. Specifying `notraildot` makes it optional to specify the trailing dot to access a file whose name lacks an extension.

Exceptions: This option is effective only on file systems for which `Rock Ridge` extensions are not active, either because they are not present on the CD-ROM, or they are explicitly ignored via the `nrr` option. If `Rock Ridge` extensions are active, `hsfs` quietly ignores this option.

`nomaplcase` — File names on High Sierra cdroms with no `Rock Ridge` extensions present should be uppercase characters only. By default, `hsfs` maps file names read from a `non-Rock Ridge` disk to all lowercase characters. `nomaplcase` turns off this mapping. The exceptions for `notraildot` discused above apply to `nomaplcase`.

`-O` — Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error `device busy`.

Files

`/etc/mnttab` table of mounted file systems

`/etc/vfstab` list of default parameters for each file system

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`mount(1M)`, `mountall(1M)`, `mount(2)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`

Notes

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Name mount_nfs – mount remote NFS resources

Synopsis

```
mount [-F nfs] [generic_options] [-o specific_options] [-O] resource

mount [-F nfs] [generic_options] [-o specific_options] [-O] mount_point

mount [-F nfs] [generic_options] [-o specific_options]
     [-O] resource mount_point
```

Description The `mount` utility attaches a named *resource* to the file system hierarchy at the pathname location *mount_point*, which must already exist. If *mount_point* has any contents prior to the `mount` operation, the contents remain hidden until the *resource* is once again unmounted.

`mount_nfs` starts the `lockd(1M)` and `statd(1M)` daemons if they are not already running.

If the resource is listed in the `/etc/vfstab` file, the command line can specify either *resource* or *mount_point*, and `mount` consults `/etc/vfstab` for more information. If the `-F` option is omitted, `mount` takes the file system type from `/etc/vfstab`.

If the resource is not listed in the `/etc/vfstab` file, then the command line must specify both the *resource* and the *mount_point*.

host can be an IPv4 or IPv6 address string. As IPv6 addresses already contain colons, enclose *host* in a pair of square brackets when specifying an IPv6 address string. Otherwise the first occurrence of a colon can be interpreted as the separator between the host name and path, for example, `[1080::8:800:200C:417A]:tmp/file`. See `inet(7P)` and `inet6(7P)`.

host:*pathname*
: Where *host* is the name of the NFS server host, and *pathname* is the path name of the directory on the server being mounted. The path name is interpreted according to the server's path name parsing rules and is not necessarily slash-separated, though on most servers, this is the case.

nfs://*host*[:*port*]/*pathname*
: This is an NFS URL and follows the standard convention for NFS URLs as described in *NFS URL Scheme*, RFC 2224. See the discussion of URL's and the public option under `NFS FILE SYSTEMS` for a more detailed discussion.

host:*pathname* *nfs*://*host*[:*port*]/*pathname*
: *host*:*pathname* is a comma-separated list of *host*:*pathname*.

  See the discussion of replicated file systems and failover under `NFS FILE SYSTEMS` for a more detailed discussion.

hostlist pathname
: *hostlist* is a comma-separated list of hosts.

  See the discussion of replicated file systems and failover under `NFS FILE SYSTEMS` for a more detailed discussion.

The mount command maintains a table of mounted file systems in /etc/mnttab, described in mnttab(4).

mount_nfs supports both NFSv3 and NFSv4 mounts. The default NFS version is NFSv4.

Options See mount(1M) for the list of supported *generic_options*. See share_nfs(1M) for a description of server options.

-o *specific_options*
Set file system specific options according to a comma-separated list with no intervening spaces.

acdirmax=*n*
Hold cached attributes for no more than *n* seconds after directory update. The default value is 60.

acdirmin=*n*
Hold cached attributes for at least *n* seconds after directory update. The default value is 30.

acregmax=*n*
Hold cached attributes for no more than *n* seconds after file modification. The default value is 60.

acregmin=*n*
Hold cached attributes for at least *n* seconds after file modification. The default value is 3.

actimeo=*n*
Set *min* and *max* times for regular files and directories to *n* seconds. See "File Attributes," below, for a description of the effect of setting this option to 0.

See "Specifying Values for Attribute Cache Duration Options," below, for a description of how acdirmax, acdirmin, acregmax, acregmin, and actimeo are parsed on a mount command line.

bg | fg
If the first attempt fails, retry in the background, or, in the foreground. The default is fg.

forcedirectio | noforcedirectio
If forcedirectio is specified, then for the duration of the mount, forced direct I/O is used. If the filesystem is mounted using forcedirectio, data is transferred directly between client and server, with no buffering on the client. If the filesystem is mounted using noforcedirectio, data is buffered on the client. forcedirectio is a performance option that is of benefit only in large sequential data transfers. The default behavior is noforcedirectio.

grpid
By default, the GID associated with a newly created file obeys the System V semantics; that is, the GID is set to the effective GID of the calling process. This behavior can be

overridden on a per-directory basis by setting the set-GID bit of the parent directory; in this case, the GID of a newly created file is set to the GID of the parent directory (see `open(2)` and `mkdir(2)`). Files created on file systems that are mounted with the `grpid` option obeys BSD semantics independent of whether the set-GID bit of the parent directory is set; that is, the GID is unconditionally inherited from that of the parent directory.

`hard` | `soft`
Continue to retry requests until the server responds (`hard`) or give up and return an error (`soft`). The default value is `hard`. Note that NFSv4 clients do not support soft mounts.

`intr` | `nointr`
Allow (do not allow) keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted file system. The default is `intr`, which makes it possible for clients to interrupt applications that can be waiting for a remote mount.

`noac`
Suppress data and attribute caching. The data caching that is suppressed is the write-behind. The local page cache is still maintained, but data copied into it is immediately written to the server.

`nocto`
Do not perform the normal close-to-open consistency. When a file is closed, all modified data associated with the file is flushed to the server and not held on the client. When a file is opened the client sends a request to the server to validate the client's local caches. This behavior ensures a file's consistency across multiple NFS clients. When `-nocto` is in effect, the client does not perform the flush on close and the request for validation, allowing the possiblity of differences among copies of the same file as stored on multiple clients.

This option can be used where it can be guaranteed that accesses to a specified file system are made from only one client and only that client. Under such a condition, the effect of `-nocto` can be a slight performance gain.

`port=`*n*
The server IP port number. The default is `NFS_PORT`. If the `port` option is specified, and if the resource includes one or more NFS URLs, and if any of the URLs include a `port` number, then the `port` number in the option and in the URL must be the same.

`posix`
Request POSIX.1 semantics for the file system. Requires a mount Version 2 `mountd(1M)` on the server. See `standards(5)` for information regarding POSIX.

`proto=`*netid* | `rdma`
By default, the transport protocol that the NFS mount uses is the first available TCP transport supported both by the client and the server. If no TCP transport is found, then it attempts to use a UDP transport, as ordered in the `/etc/netconfig` file.

Use this option to override the default behavior.

`proto` is set to the value of *netid* or `rdma`. *netid* is the value of the `network_id` field entry in the `/etc/netconfig` file.

The UDP protocol is not supported for NFS Version 4. If you specify a UDP protocol with the `proto` option, NFS version 4 is not used.

`public`
The `public` option forces the use of the public file handle when connecting to the NFS server. The resource specified might not have an NFS URL. See the discussion of URLs and the public option under `NFS FILE SYSTEMS` for a more detailed discussion.

`quota` | `noquota`
Enable or prevent `quota(1M)` to check whether the user is over quota on this file system; if the file system has quotas enabled on the server, quotas are still checked for operations on this file system.

`remount`
Remounts a read-only file system as read-write (using the `rw` option). This option cannot be used with other `-o` options, and this option works only on currently mounted read-only file systems.

`retrans=`*n*
Set the number of NFS retransmissions to *n*. The default value is `5`. For connection-oriented transports, this option has no effect because it is assumed that the transport performs retransmissions on behalf of NFS.

`retry=`*n*
The number of times to retry the `mount` operation. The default for the `mount` command is `10000`.

The default for the automounter is `0`, in other words, do not retry. You might find it useful to increase this value on heavily loaded servers, where automounter traffic is dropped, causing unnecessary server not responding errors.

`rsize=`*n*
Set the read buffer size to a maximum of *n* bytes. The default value is `1048576` when using connection-orientated transports with Version 3 or Version 4 of the NFS protocol, and `32768` when using connection-less transports. The default can be negotiated down if the server prefers a smaller transfer size. "`Read`" operations may not necessarily use the maximum buffer size. When using Version 2, the default value is `32768` for all transports.

`sec=`*mode*
Set the security *mode* for NFS transactions. If `sec=` is not specified, then the default action is to use `AUTH_SYS` over NFS Version 2 mounts, or to negotiate a *mode* over NFS Version 3 or Version 4 mounts.

NFS Version 3 mounts negotiate a security mode when the server returns an array of security modes. The client picks the first mode in the array that is supported on the client. In negotiations, an NFS Version 3 client is limited to the security flavors listed in `/etc/nfssec.conf`.

NFS Version 4 mounts negotiate a security mode when the server returns an array of security modes. The client attempts the mount with each security mode, in order, until one is successful.

Only one mode can be specified with the `sec=` option. See `nfssec(5)` for the available *mode* options.

`secure`
This option has been deprecated in favor of the `sec=`*dh* option.

`timeo=`*n*
Set the NFS timeout to *n* tenths of a second. The default value is `11` tenths of a second for connectionless transports, and `600` tenths of a second for connection-oriented transports. This value is ignored for connectionless transports. Such transports might implement their own timeouts, which are outside the control of NFS.

`vers=`*NFS version number*
By default, the version of NFS protocol used between the client and the server is the highest one available on both systems. The default maximum for the client is Version 4. This can be changed by setting the `NFS_CLIENT_VERSMAX` parameter in `/etc/default/nfs` to a valid version (2, 3, or 4). If the NFS server does not support the client's default maximum, the next lowest version attempted until a matching version is found.

`wsize=`*n*
Set the write buffer size to a maximum of *n* bytes. The default value is `1048576` when using connection-orientated transports with Version 3 or Version 4 of the NFS protocol, and `32768` when using connection-less transports. The default can be negotiated down if the server prefers a smaller transfer size. "`Write`" operations may not necessarily use the maximum buffer size. When using Version 2, the default value is `32768` for all transports.

`xattr` | `noxattr`
Allow or disallow the creation and manipulation of extended attributes. The default is `xattr`. See `fsattr(5)` for a description of extended attributes.

`-O`
Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount fails, producing the error "device busy."

Nfs File Systems

Background versus Foreground

File systems mounted with the `bg` option indicate that `mount` is to retry in the background if the server's mount daemon (`mountd(1M)`) does not respond. `mount` retries the request up to the count specified in the `retry=`*n* option. (Note that the default value for `retry` differs between `mount` and `automount`. See the description of `retry`, above.) Once the file system is mounted, each NFS request made in the kernel waits `timeo=`*n* tenths of a second for a response. If no response arrives, the time-out is multiplied by `2` and the request is retransmitted. When the number of retransmissions has reached the number specified in the `retrans=`*n* option, a file system mounted with the `soft` option returns an error on the request; one mounted with the `hard` option prints a warning message and continues to retry the request.

Hard versus Soft

File systems that are mounted read-write or that contain executable files should always be mounted with the `hard` option. Applications using `soft` mounted file systems can incur unexpected I/O errors, file corruption, and unexpected program core dumps. The soft option is not recommended.

Authenticated requests

The server can require authenticated NFS requests from the client. `sec=`*dh* authentication might be required. See `nfssec(5)`.

URLs and the public option

If the `public` option is specified, or if the *resource* includes and NFS URL, `mount` attempts to connect to the server using the public file handle lookup protocol. See *WebNFS Client Specification*, RFC 2054. If the server supports the public file handle, the attempt is successful; `mount` does not need to contact the server's `rpcbind(1M)` and the `mountd(1M)` daemons to get the port number of the `mount` server and the initial file handle of *pathname*, respectively. If the NFS client and server are separated by a firewall that allows all outbound connections through specific ports, such as `NFS_PORT`, then this enables NFS operations through the firewall. The public option and the NFS URL can be specified independently or together. They interact as specified in the following matrix:

	Resource Style	
	host:*pathname*	NFS URL
public option	Force public file handle and fail mount if not supported.	Force public file handle and fail mount if not supported.
	Use Native paths.	Use Canonical paths.
default	Use MOUNT protocol.	Try public file handle with Canonical paths. Fall back to MOUNT protocol if not supported.

A Native path is a path name that is interpreted according to conventions used on the native operating system of the NFS server. A Canonical path is a path name that is interpreted according to the URL rules. See *Uniform Resource Locators (URL)*, RFC 1738. See Examples for uses of Native and Canonical paths.

Replicated file systems and failover

resource can list multiple read–only file systems to be used to provide data. These file systems should contain equivalent directory structures and identical files. It is also recommended that they be created by a utility such as `rdist(1)`. The file systems can be specified either with a comma–separated list of *host:/pathname* entries and/or NFS URL entries, or with a comma –separated list of hosts, if all file system names are the same. If multiple file systems are named and the first server in the list is down, failover uses the next alternate server to access files. If the read–only option is not chosen, replication is disabled. File access, for NFS Versions 2 and 3, is blocked on the original if NFS locks are active for that file.

File Attributes

To improve NFS read performance, files and file attributes are cached. File modification times get updated whenever a write occurs. However, file access times can be temporarily out-of-date until the cache gets refreshed.

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting `actimeo=`*n* sets flush time to *n* seconds for both regular files and directories.

Setting `actimeo=0` disables attribute caching on the client. This means that every reference to attributes is satisfied directly from the server though file data is still cached. While this guarantees that the client always has the latest file attributes from the server, it has an adverse effect on performance through additional latency, network load, and server load.

Setting the `noac` option also disables attribute caching, but has the further effect of disabling client write caching. While this guarantees that data written by an application is written directly to a server, where it can be viewed immediately by other clients, it has a significant adverse effect on client write performance. Data written into memory-mapped file pages (`mmap(2)`) are not written directly to this server.

Specifying Values for Attribute Cache Duration Options

The attribute cache duration options are `acdirmax`, `acdirmin`, `acregmax`, `acregmin`, and `actimeo`, as described under OPTIONS. A value specified for `actimeo` sets the values of all attribute cache duration options except for any of these options specified following `actimeo` on a `mount` command line. For example, consider the following command:

```
example# mount -o acdirmax=10,actimeo=1000 server:/path /localpath
```

Because `actimeo` is the last duration option in the command line, its value (`1000`) becomes the setting for all of the duration options, including `acdirmax`. Now consider:

```
example# mount -o actimeo=1000,acdirmax=10 server:/path /localpath
```

Because the `acdirmax` option follows `actimeo` on the command line, it is assigned the value specified (`10`). The remaining duration options are set to the value of `actimeo` (`1000`).

Examples

EXAMPLE 1 Mounting an NFS File System

To mount an NFS file system:

```
example# mount serv:/usr/src /usr/src
```

EXAMPLE 2 Mounting An NFS File System Read-Only With No suid Privileges

To mount an NFS file system read-only with no suid privileges:

```
example# mount -r -o nosuid serv:/usr/src /usr/src
```

EXAMPLE 3 Mounting An NFS File System Over Version 2, with the UDP Transport

To mount an NFS file system over Version 2, with the UDP transport:

```
example# mount -o vers=2,proto=udp serv:/usr/src /usr/src
```

EXAMPLE 4 Mounting an NFS File System Using An NFS URL

To mount an NFS file system using an NFS URL (a canonical path):

```
example# mount nfs://serv/usr/man /usr/man
```

EXAMPLE 5 Mounting An NFS File System Forcing Use Of The Public File Handle

To mount an NFS file system and force the use of the public file handle and an NFS URL (a canonical path) that has a non 7–bit ASCII escape sequence:

```
example# mount -o public nfs://serv/usr/%A0abc /mnt/test
```

EXAMPLE 6 Mounting an NFS File System Using a Native Path

To mount an NFS file system using a native path (where the server uses colons (":") as the component separator) and the public file handle:

```
example# mount -o public serv:C:doc:new /usr/doc
```

EXAMPLE 7 Mounting a Replicated Set of NFS File Systems with the Same Pathnames

To mount a replicated set of NFS file systems with the same pathnames:

```
example# mount serv—a,serv—b,serv—c:/usr/man /usr/man
```

EXAMPLE 8 Mounting a Replicated Set of NFS File Systems with Different Pathnames

To mount a replicated set of NFS file systems with different pathnames:

```
example# mount serv—x:/usr/man,serv—y:/var/man,nfs://serv-z/man /usr/man
```

Files

`/etc/mnttab`
table of mounted file systems

`/etc/dfs/fstypes`
default distributed file system type

`/etc/vfstab`
table of automatically mounted resources

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfscu

See Also

`rdist(1)`, `lockd(1M)`, `mountall(1M)`, `mountd(1M)`, `mountd(1M)`, `quota(1M)`, `statd(1M)`, `mkdir(2)`, `mmap(2)`, `mount(2)`, `open(2)`, `umount(2)`, `mnttab(4)`, `attributes(5)`, `fsattr(5)`, `nfssec(5)`, `standards(5)`, `inet(7P)`, `inet6(7P)`, `lofs(7FS)`

Callaghan, Brent, *WebNFS Client Specification*, RFC 2054, October 1996.

Callaghan, Brent, *NFS URL Scheme*, RFC 2224, October 1997.

Berners-Lee, Masinter & McCahill , *Uniform Resource Locators (URL)*, RFC 1738, December 1994.

Notes

An NFS server should not attempt to mount its own file systems. See `lofs(7FS)`.

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on *the directory to which the symbolic link refers,* rather than being mounted on top of the symbolic link itself.

SunOS 4.x used the `biod` maintenance procedure to perform parallel read-ahead and write-behind on NFS clients. SunOS 5.x made `biod` obsolete with multi-threaded processing, which transparently performs parallel read-ahead and write-behind.

Since the root (`/`) file system is mounted read-only by the kernel during the boot process, only the `remount` option (and options that can be used in conjunction with `remount`) affect the root (`/`) entry in the `/etc/vfstab` file.

`mount_cachefs` cannot be used with replicated NFS mounts or any NFS Version 4 mount.

Name mount_pcfs – mount pcfs file systems

Synopsis

```
mount -F pcfs [generic_options]
     [-o FSType-specific_options] special | mount_point

mount -F pcfs [generic_options]
     [-o FSType-specific_options] special mount_point
```

Description `mount` attaches an MS-DOS file system (`pcfs`) to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the `mount` operation, these are hidden until the file system is unmounted.

If `mount` is invoked with *special* or *mount_point* as the only arguments, `mount` will search `/etc/vfstab` to fill in the missing arguments, including the *FSType-specific_options*; see `mount(1M)` for more details.

The *special* argument can be one of two special device file types:

- A floppy disk, such as `/dev/diskette0` or `/dev/diskette1`.
- A DOS logical drive on a hard disk expressed as *device-name*:*logical-drive* , where *device-name* specifies the special block device-file for the whole disk and *logical-drive* is either a drive letter (c through z) or a drive number (1 through 24). Examples are `/dev/dsk/c0t0d0p0:`*c* and `/dev/dsk/c0t0d0p0:`*1*.

The *special* device file type must have a formatted MS-DOS file system with either a 12-bit, 16-bit, or 32-bit File Allocation Table.

Options *generic_options*

See `mount(1M)` for the list of supported options.

`-o`

Specify `pcfs` file system-specific options. The following options are supported:

`foldcase` | `nofoldcase`

Force uppercase characters in filenames to lowercase when reading them from the filesystem. This is for compatibility with the previous behavior of `pcfs`. The default is `nofoldcase`.

`hidden` | `nohidden`

Allow or disallow listing of files with hidden or system bits set. Option `hidden` is the default. When `nohidden` is effect, hidden and system files are neither visible nor accessible. Note that PCFS in previous releases of the Solaris operating system used the `nohidden` option as the default.

`atime` | `noatime`

Enable or disable write access timestamps on DOS-formatted media. Default for fixed disks is `atime`, while for removable media `noatime` is used. The latter default is so that writes to flash-based media (“memory sticks”) can be minimized, to prolong lifetime.

`timezone=`*timezone*
Timestamps on DOS-formatted media are recorded in the local time of the recording system. This can cause confusion when accessing removable media in which the recording and receiving system use different time zones. Use this option to force media timestamps to be interpreted for a specific time zone. The `mount_pcfs` command converts the given time zone name into a numerical offset that is passed to the `pcfs` kernel module, using the same rules as described in `environ(5)` for the `TZ` environment variable. By default, the `timezone` value is taken from the `TZ` environment variable.

Files

`/etc/mnttab`
table of mounted file systems

`/etc/vfstab`
list of default parameters for each file system

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu

See Also

`mount(1M)`, `mountall(1M)`, `mount(2)`, `stat(2)`, `time(2)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`, `environ(5)`, `pcfs(7FS)`

Notes

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Name mount_tmpfs – mount tmpfs file systems

Synopsis `mount [-F tmpfs] [-o` *specific_options*`] [-O]` *special mount_point*

Description `tmpfs` is a memory based file system which uses kernel resources relating to the VM system and page cache as a file system.

`mount` attaches a `tmpfs` file system to the file system hierarchy at the pathname location *mount_point*, which must already exist. If *mount_point* has any contents prior to the `mount` operation, these remain hidden until the file system is once again unmounted. The attributes (mode, owner, and group) of the root of the `tmpfs` filesystem are inherited from the underlying *mount_point*, provided that those attributes are determinable. If not, the root's attributes are set to their default values.

The *special* argument is usually specified as `swap` but is in fact disregarded and assumed to be the virtual memory resources within the system.

Options

`-o` *specific_options* Specify `tmpfs` file system specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

size=*sz* The *sz* argument controls the size of this particular `tmpfs` file system. If the argument is has a 'k' suffix, the number will be interpreted as a number of kilobytes. An 'm' suffix will be interpreted as a number of megabytes. No suffix is interpreted as bytes. In all cases, the actual size of the file system is the number of bytes specified, rounded up to the physical pagesize of the system.

`xattr` | `noxattr` Allow or disallow the creation and manipulation of extended attributes. The default is `xattr`. See `fsattr(5)` for a description of extended attributes.

`-O` Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error`device busy`.

Files `/etc/mnttab` Table of mounted file systems

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also mount(1M), mkdir(2), mount(2), open(2), umount(2), mnttab(4), attributes(5), fsattr(5), tmpfs(7FS)

Notes If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Name mount_udfs – mount a udfs file system

Synopsis

```
mount -F udfs [generic_options] [-o specific_options] [-O] special mount_point

mount -F udfs [generic_options] [-o specific_options] [-O] special | mount_point
```

Description The `mount` utility attaches a `udfs` file system to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the mount operation, these are hidden until the file system is unmounted.

If mount is invoked with either *special* or *mount_point* as the only arguments, `mount` searches `/etc/vfstab` to fill in the missing arguments, including the *specific_options*. See `mount(1M)`.

If *special* and *mount_point* are specified without any *specific_options*, the default is `rw`.

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Options See `mount`(1M) for the list of supported *generic_options*.

The following options are supported:

`-o` *specific_options* Specify `udfs` file system specific options in a comma-separated list with no intervening spaces. The following *specific_options* are available:

- m — Mount the file system without making an entry in `/etc/mnttab`.
- remount — Remount the file system as read-write. The option is used in conjunction with the `rw` option.

  A file system mounted read-only can be remounted as read-write. This option fails if the file system is not currently mounted.

`-O` Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount fails, producing the error `device busy`.

Files

`/etc/mnttab` Table of mounted file systems

`/etc/vfstab` List of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWudf

See Also fsck(1M), fsck_udfs(1M), mount(1M), mountall(1M), mount(2), mnttab(4), vfstab(4), attributes(5)

Diagnostics

not super user	The command is run by a non-root user. Run as root.
no such device	The device name specified does not exist.
not a directory	The specified mount point is not a directory.
is not an udfs file system	The device specified does not contain a udf 1.50 file system or the udfs file system module is not available.
is already mounted	The specified device is already in use.
not a block device	The device specified is not a block device. Use block device to mount.
write-protected	The device is read-only.
is corrupted. needs checking	The file system is in an inconsistent state. Run `fsck`.

Notes Copy-protected files can be stored on DVD-ROM media using Universal Disk Format (UDF). Reading these copy-protected files is not possible as this involves an authentication process. Unless an authentication process between the host and the drive is completed, reading these copy-protected files after mounting and before the authentication process, returns an error.

Name mount_ufs – mount ufs file systems

Synopsis

```
mount -F ufs [generic_options] [-o specific_options]
     [-O] special | mount_point

mount -F ufs [generic_options] [-o specific_options]
     [-O] special mount_point
```

Description The `mount` utility attaches a `ufs` file system to the file system hierarchy at the *mount_point*, which is the pathname of a directory. If *mount_point* has any contents prior to the `mount` operation, these are hidden until the file system is unmounted.

The `ufs` file system supports direct mounting of files containing the file system as well as block devices. See `mount(1M)` and `lofiadm(1M)`.

If `mount` is invoked with *special* or *mount_point* as the only arguments, `mount` will search `/etc/vfstab` to fill in the missing arguments, including the *specific_options*. See `mount(1M)`.

If *special* and *mount_point* are specified without any *specific_options*, the default is `rw`.

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

Options See `mount(1M)` for the list of supported *generic_options*.

The following options are supported:

`-o` *specific_options*
Specify `ufs` file system specific options in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is printed and the invalid options are ignored. The following options are available:

`dfratime` | `nodfratime`
By default, writing access time updates to the disk may be deferred (`dfratime`) for the file system until the disk is accessed for a reason other than updating access times. `nodfratime` disables this behavior.

If power management is enabled on the system, do not set `nodfratime` unless `noatime` is also set. If you set `nodfratime` without setting `noatime`, the disk is spun up every time a file within a file system on the disk is accessed - even if the file is not modified.

`forcedirectio` | `noforcedirectio`
If `forcedirectio` is specified and supported by the file system, then for the duration of the mount, forced direct I/O will be used. If the filesystem is mounted using `forcedirectio`, data is transferred directly between user address space and the disk. If the filesystem is mounted using `noforcedirectio`, data is buffered in kernel address

space when data is transferred between user address space and the disk. `forcedirectio` is a performance option that is of benefit only in large sequential data transfers. The default behavior is `noforcedirectio`.

`global` | `noglobal`
If `global` is specified and supported on the file system, and the system in question is part of a cluster, the file system will be globally visible on all nodes of the cluster. If `noglobal` is specified, the mount will not be globally visible. The default behavior is `noglobal`.

`intr` | `nointr`
Allow (do not allow) keyboard interrupts to kill a process that is waiting for an operation on a locked file system. The default is `intr`.

`largefiles` | `nolargefiles`
If `nolargefiles` is specified and supported by the file system, then for the duration of the mount it is guaranteed that all regular files in the file system have a size that will fit in the smallest object of type `off_t` supported by the system performing the mount. The mount will fail if there are any files in the file system not meeting this criterion. If `largefiles` is specified, there is no such guarantee. The default behavior is `largefiles`.

If `nolargefiles` is specified, `mount` will fail for `ufs` if the file system to be mounted has contained a large file (a file whose size is greater than or equal to 2 Gbyte) since the last invocation of `fsck` on the file system. The large file need not be present in the file system at the time of the mount for the mount to fail; it could have been created previously and destroyed. Invoking `fsck` (see `fsck_ufs(1M)`) on the file system will reset the file system state if no large files are present. After invoking `fsck`, a successful mount of the file system with `nolargefiles` specified indicates the absence of large files in the file system; an unsuccessful mount attempt indicates the presence of at least one large file.

`logging` | `nologging`
If `logging` is specified, then logging is enabled for the duration of the mounted file system. Logging is the process of storing transactions (changes that make up a complete UFS operation) in a log before the transactions are applied to the file system. Once a transaction is stored, the transaction can be applied to the file system later. This prevents file systems from becoming inconsistent, therefore reducing the possibility that `fsck` might run. And, if `fsck` is bypassed, logging generally reduces the time required to reboot a system.

The default behavior is `logging` for all UFS file systems.

The log is allocated from free blocks in the file system, and is sized approximately 1 Mbyte per 1 Gbyte of file system, up to a maximum of 256 Mbytes. The log size may be larger (up to a maximum of 512 Mbytes) dependent upon the number of cylinder groups present in the file system.

Logging is enabled on any UFS file system, including root (`/`), except under the following conditions:

- When logging is specifically disabled.
- If there is insufficient file system space for the log. In this case, the following message is displayed and file system is still mounted:

```
# mount /dev/dsk/c0t4d0s0 /mnt
  /mnt: No space left on device
  Could not enable logging for /mnt on  /dev/dsk/c0t4d0s0.
```

The log created by UFS logging is continually flushed as it fills up. The log is totally flushed when the file system is unmounted or as a result of the `lockfs -f` command.

`m`
Mount the file system without making an entry in `/etc/mnttab`.

`noatime`
By default, the file system is mounted with normal access time (`atime`) recording. If `noatime` is specified, the file system will ignore access time updates on files, except when they coincide with updates to the `ctime` or `mtime`. See `stat(2)`. This option reduces disk activity on file systems where access times are unimportant (for example, a Usenet news spool).

`noatime` turns off access time recording regardless of `dfratime` or `nodfratime`.

The POSIX standard requires that access times be marked on files. `-noatime` ignores them unless the file is also modified.

`nosec`
By default, Access Control Lists (ACLs) are supported on a mounted UFS file system. Use this option to disallow the setting or any modification of an ACL on a file within a mounted UFS file system. See `getfacl(1)` for background on ACLs.

`onerror` = *action*
This option specifies the action that UFS should take to recover from an internal inconsistency on a file system. Specify *action* as `panic`, `lock`, or `umount`. These values cause a forced system shutdown, a file system lock to be applied to the file system, or the file system to be forcibly unmounted, respectively. The default is `panic`.

`quota`
Quotas are turned on for the file system.

`remount`
Remounts a file system with a new set of options. All options not explicitly set with `remount` revert to their default values.

`rq`
Read-write with quotas turned on. Equivalent to `rw, quota`.

`xattr` | `noxattr`
Allow or disallow the creation and manipulation of extended attributes. The default is `xattr`. See `fsattr(5)` for a description of extended attributes.

`-O`
Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error "device busy".

Examples

EXAMPLE 1 Turning Off (and On) Logging

The following command turns off logging on an already mounted file system. The subsequent command restores logging.

```
# mount -F ufs -o remount,nologging /export
# (absence of message indicates success)
# mount -F ufs -o remount,logging /export
```

In the preceding commands, the `-F ufs` option is not necessary.

Files

`/etc/mnttab`
table of mounted file systems

`/etc/vfstab`
list of default parameters for each file system

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs

See Also

`getfacl(1)`, `fsck(1M)`, `fsck_ufs(1M)`, `lofiadm(1M)`, `mount(1M)`, `mountall(1M)`, `fcntl(2)`, `mount(2)`, `stat(2)`, `mnttab(4)`, `vfstab(4)`, `attributes(5)`, `fsattr(5)`, `largefile(5)`

Notes

Since the root (`/`) file system is mounted read-only by the kernel during the boot process, only the `remount` option (and options that can be used in conjunction with `remount`) affect the root (`/`) entry in the `/etc/vfstab` file.

Name mount_xmemfs – mount xmemfs file systems

Synopsis

```
mount -F xmemfs [generic_options] -o[largebsize,]size=sz
     [-O] special mount_point
```

Interface Level `xmemfs` is obsolete. Users requiring large physical memory should migrate to 64-bit platform support.

Description `xmemfs` is an extended memory file system which provides file system semantics to manage and access large amounts of physical memory which can exceed 4 GB in size.

`mount` attaches a `xmemfs` file system to the file system hierarchy at the pathname location `mount_point`, which must already exist. If `mount_point` has any contents prior to the mount operation, these remain hidden until the file system is once again unmounted. The attributes (`mode`, `owner`, and `group`) of the root of the `xmemfs` filesystem are inherited from the underlying `mount_point`, provided that those attributes are determinable. If not, the root's attributes are set to their default values.

The special argument is not currently used by `xmemfs` but a placeholder, (such as `xmem`), needs to be specified nevertheless.

Options See `mount(1M)` for the list of supported *generic_options*.

`-o`*specific_options* Specify `xmemfs` file system specific options in a comma-separated list with *no intervening spaces*. If invalid options are specified, a warning message is printed and the invalid options are ignored.

The `size=`*sz specific option* is required.

The following options are available:

size=*sz* The *sz* argument specifies the desired size of this particular `xmemfs` file system. If the *sz* argument has a `k` suffix, the number is interpreted as kilobytes. An `m` suffix is interpreted as megabytes and `g` is interpreted as gigabytes. A *sz* specified with no suffix is interpreted as bytes.

In all cases, the actual size of the file system is the number of bytes specified, rounded up to the physical `pagesize` of the system or to the large page size if `largebsize` is specified.

This *specific_option* is required.

largebsize If `largebsize` is specified, `xmemfs` uses the large memory page size as the file system block size. On `IA32`, the large memory page size with `mmu36` which supports PAE (Physical Address Extension) is 2 MB. The large memory

page size without `mmu36/PAE` is 4 MB. If there is no large page support, the file system block size is `PAGESIZE`.

`-O` Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount fails, producing the error `device busy`.

Files `/etc/mnttab` table of mounted file systems

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Architecture	i386
Interface Stability	Obsolete

See Also `mount(1M)`, `mount(2)`, `mkdir(2)`, `open(2)`, `umount(2)`, `mnttab(4)`, `attributes(5)`,`xmemfs(7FS)`

Notes If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on the directory to which the symbolic link refers, rather than on top of the symbolic link itself.

The only file types allowed on xmemfs are directories and regular files. The execution of object files resident in `xmemfs` is not supported. Execution is prevented by not allowing users to set execute permissions on regular files.

Support for `xmemfs` may be removed in a future release of Solaris.

Name mpathadm – multipath discovery and administration

Synopsis `mpathadm` *subcommand direct-object* `[`*options*`] [`*operand*`]`

Description The `mpathadm` command enables multipathing discovery and management. The `mpathadm` command is implemented as a set of subcommands, many with their own options, that are described in the section for that subcommand. Options not associated with a particular subcommand are described under OPTIONS. The `mpathadm` subcommands operate on a *direct-object*. These are described in this section for each subcommand. The *direct-objects*, *initiator-port*, *target-port*, and*logical-unit* in the subcommands are consistent with SCSI standard definitions.

The `mpathadm` command supports the following subcommands, which are described in detail in subsections that follow.

`list`	Display a list of discovered instances for a given object.
`show`	Display information about a given object instance.
`modify`	Modify properties of an object.
`enable`	Enable an object.
`disable`	Disable an object.
`failover`	Cause target port group failover for a logical-unit.
`override`	Set a path to be used over other paths on a logical-unit.

The `mpathadm` subcommands operate on a *direct-object*. These are described in this section for each subcommand.

list Subcommand The syntax for the `list` subcommand is:

```
# mpathadm list direct-object [operands...]
```

The `list` subcommand displays data for following direct-objects:

`mpath-support` [*mpath-support-name*, ...]
List the multipathing support that can be administered by this CLI. This presents itself in the form of a library name registered through the MPAPI framework. If no mpath-support name *mpath-support-name* is specified, all registered multipathing support libraries will be displayed.

`initiator-port` [*initiator-port-name*, ...]
List the initiator ports that are discovered on this system. If no *initiator-port-name* is specified, all discovered initiator ports are displayed.

{`logical-unit` | `lu`} [*options*] [*logical-unit-name*, ...]
List the information on multipath logical units. If no *logical-unit-name* is specified, all discovered logical-units will be displayed.

Options for `list logical-unit` are as follows:

`-n`, `--name` *name*	Return the logical unit name that is associated with the given name string. This name can be extracted from the output of the `mpathadm show lu` command.
`-t`, `--target-port` *target-port-name*	Return the list of logical units names that are associated with the given *target-port-name*.

`show` Subcommand The syntax for the `show` subcommand is:

```
# mpathadm show direct-object [operands...]
```

The `show` subcommand displays detailed information for following the direct-objects:

`mpath-support` [*mpath-support-name*, ...]
Show the detailed information on the given *mpath-support-name* if the name exists. If the given *mpath-support-name* supports only a limited set of device products, the list of device products will be listed as part of the output.

`initiator-port` *initiator-port-name*[,*initiator-port-name*, ...]
Show the detailed information for the given *initiator-port-name*.

{`logical-unit` | `lu`} [*logical-unit-name*, ...]
Display the detailed information on multipath logical unit(s), including path and target port group information. Note that the name property in the logical unit information represents the identifier for this LUN, derived from the hardware, and used by this system. If the name is derived from SCSI Inquiry Vital Product Data (VPD) page 83h, the name type property represents an associated identifier type defined by the SCSI standards.

`modify` Subcommand The syntax for the `modify` subcommand is:

```
# mpathadm modify direct-object [options] [operands...]
```

The `modify` subcommand modifies characteristics of the following direct-objects:

`mpath-support` [*options*] *mpath-support-name*, ...
Configuration management of an *mpath-support*. Options to modify mpath-support are as follows:

Set auto failback. Applicable only when mpath-support provides auto failback support.

Set auto probing. Applicable only when mpath-support provides auto probing support.

Change the default load balance type. The `loadbalance` type is one of the

supported types listed in the `show mpath-support` output.

{`logical-unit` | `lu`} [*options*] *logical-unit-name*, ... — Configuration management of a logical unit. Options to modify `logical-unit` are as follows:

Set auto failback. Applicable only when `mpath-support` provides auto failback support

Set auto probing. Applicable only when `mpath-support` provides auto probing support.

Set load balance type for the logical unit. Applicable only when load balance configuration is supported at the logical unit level.

`enable` Subcommand

The syntax for the `enable` subcommand is:

```
# mpathadm disable [options]
```

The `enable` subcommand supports the following direct-objects to be enabled:

`path -i` *initiator-port-name* `-t` *target-port-name* `-l` *logical-unit-name* — The path that consists of the specified initiator port, target port, and logical unit will be enabled.

`disable` Subcommand

The syntax for the `disable` subcommand is:

```
# mpathadm disable [options]
```

The `disable` subcommand supports the following direct-objects to be disabled:

`path -i` *initiator-port-name* `-t` *target-port-name* `-l` *logical-unit-name* — The path that consists of the specified initiator port, target port, and logical unit will be disabled.

`failover` Subcommand

The syntax for the `failover` subcommand is:

```
# mpathadm failover direct-object [operand]
```

The `failover` subcommand supports failover for the following direct-objects:

{`logical-unit` | `lu`} *logical-unit-name* — The target port group will failover when the given logical-unit is asymmetric and supports explicit

state change. The currently active target port group will be changed to the standby state and the standby target port group will be active.

`override` Subcommand

The syntax for the `override` subcommand is:

```
# mpathadm override [options]
```

The `override` subcommand controls whether or not the following direct-objects override another:

path {`-i` *initiator-port-name* `-t` *target-port-name* | `-c`} `-l` *logical-unit-name*

Cause a path that consists of the specified initiator port, target port, and logical unit to override other paths on the logical unit. Once a path overrides other paths, the mpath-support uses only that path regardless of any other path selection configuration. The `-c` option cancels the setting. The path that consists of the specified initiator port, target port, and logical unit will be disabled.

Options for override path are as follows:

`-i`, `--initiator-port` *initiator-port-name*	Represent the initiator port element of a path. Options `-t` and `-l` must also be included.
`-t`, `--target-port` *target-port-name*	Represent the target port element of a path. Options `-i` and `-l` must also be included.
`-l`, `--logical-unit` *logical-unit*	Represent the logical unit element of a path. Options `-i` and `-t` must also be included.
`-c`, `--cancel`	Cancels overriding setting for the given logical unit. Option `-l` must also be included.

Options The following options are supported:

`-V`, `--version`	Display the version information.
`-?`, `--help`	Display context help. Can be used following an `mpathadm` command with no arguments, following a subcommand, or following a subcommand direct-object combination. Responds with help information appropriate for your entry. For example, if you enter: **`# mpathadm add mpath-support-help`** ... `mpathadm` responds with a display of the options available for that combination of subcommand and direct-object.

Examples

EXAMPLE 1 Obtaining a List of Multipathing Support

The following command uses the `list` subcommand to list all currently registered `mpath-support` libraries.

```
# mpathadm list mpath-support
mpath-support: libmpscsi_vhci.so
```

EXAMPLE 2 Displaying the Properties of a Particular Multipathing Support

The following command uses the `show` subcommand to display the properties for a currently registered `mpath-support` library.

```
# mpathadm show mpath-support libmpscsi_vhci.so
mpath-support:    libmpscsi_vhci.so
       Vendor:  Sun Microsystems
       Driver Name:  scsi_vhci
       Default Load Balance:  round-robin
       Supported Load Balance Types:
              round-robin
          logical-block
Allows To Activate Target Port Group Access:  yes
Allows Path Override:  no
Supported Auto Failback Config:     1
Auto Failback:    on
Failback Polling Rate (current/max):  0/0
Supported Auto Probing Config:    0
Auto Probing:  NA
Probing Polling Rate (current/max):  NA/NA
Supported Devices:
       Vendor:     SUN
       Product:  T300
       Revision:
       Supported Load Balance Types:
              round-robin
       Vendor:     SUN
       Product:  T4
       Revision:
       Supported Load Balance Types:
              round-robin
```

EXAMPLE 3 Obtaining a List of Initiator Ports Discovered Through the `mpath-support` Libraries

The following command uses the `list initiator-port` subcommand to display a list of initiator ports discovered by the currently registered `mpath-support` libraries.

```
# mpathadm list initiator-port
Initiator-Port: iqn.1986-03.com.sun:01:080020b7ac2b.437a3b3e,4000002a0000
Initiator-Port: 2000000173018713
Initiator-Port: 2000000173818713
```

EXAMPLE 4 Displaying the Properties of a Particular Initiator Port

The following command uses the `show initiator-port` subcommand to display the properties of a particular initiator port discovered using the `list initiator-port` subcommand in an example above.

```
# mpathadm show initiator-port 2000000173018713
initiator-port:     2000000173018713
        Transport Type:     Fibre Channel
        OS device File:     devices/pci@1f,4000/pci@2/SUNW,qlca@5/fp@0,0:fc
```

EXAMPLE 5 Displaying the Properties of a Particular Logical Unit

The following command uses the `show logical-unit` subcommand to display the properties of the logical unit with the specified name.

```
# mpathadm show lu /dev/rdsk/c4t60003BA27D2120004204AC2B000DAB00d0s2
Logical Unit: /dev/rdsk/c4t60003BA27D2120004204AC2B000DAB00d0s2
           mpath-support libmpscsi_vhci.so
           Vendor: SUN
           Product: T4
           Revision: 0301
    Name Type: SCSI Inquiry VPD Page 83 type 3
    Name: 60003ba27d2120004204ac2b000dab00
    Asymmetric: yes
    Current Load Balance: round-robin
    Logical Unit Group ID: NA
    Aauto Failback: on
    Auto Probing: NA

Paths:
    Initiator Port Name: 2000000173818713
    Target Port Name: 20030003ba27d212
    Override Path: NA
    Path State: OK
    Disabled: no

    Initiator Port Name: 2000000173018713
    Target Port Name: 20030003ba27d095
    Override Path: NA
    Path State: OK
    Disabled: no

Target Port Group:
    ID: 2
    Explicit Failover: yes
    Access State: standby
    Target Ports:
            Name: 20030003ba27d212
```

EXAMPLE 5 Displaying the Properties of a Particular Logical Unit *(Continued)*

```
          Relative ID: 0

    ID: 5
    Explicit Failover: yes
    Access State: active
    Target Ports
           Name: 20030003ba27d095
          Relative ID: 0
```

EXAMPLE 6 Enabling a Path

The following command uses the `enable path` subcommand to enable the path with the specified initiator port, target port, and logical unit.

```
# mpathadm enable path -i 2000000173018713 -t 20030003ba27d095     \
-l /dev/rdsk/c4t60003BA27D2120004204AC2B000DAB00d0s2
```

EXAMPLE 7 Modifying `mpath-support` To Turn On `autofailback`

```
# mpathadm modify mpath-support -a on libmpscsi_vhci.so
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmpapi
	SUNWmpapir (`exec_attr(4)` entry)
Interface Stability	Evolving

See Also

`stmsboot(1M)`, `libMPAPI(3LIB)`, `exec_attr(4)`, `attributes(5)`

Name mpstat – report per-processor or per-processor-set statistics

Synopsis

```
/usr/bin/mpstat [-aq] [-p | -P set] [interval [count]]
```

Description The `mpstat` command reports processor statistics in tabular form. Each row of the table represents the activity of one processor. The first table summarizes all activity since boot. Each subsequent table summarizes activity for the preceding interval. All values are rates listed as events per second unless otherwise noted.

During execution of the kernel status command, the `state` of the kernel can change. If relevant, a state change message is included in the `mpstat` output, in one of the following forms:

```
<<processor 3 moved from pset: -1 to: 1>>
<<pset destroyed: 1>>
<<pset created: 1>>
<<processors added: 1, 3>>
<<processors removed: 1, 3>>
```

The `mpstat` command reports the following information:

`CPU` or `SET`	Without the `-a` option, `mpstat` reports `CPU` statistics for a processor ID. With the `-a` option, `mpstat` reports `SET` statistics for a processor set ID.
`minf`	minor faults
`mjf`	major faults
`xcal`	inter-processor cross-calls
`intr`	interrupts
`ithr`	interrupts as threads (not counting clock interrupt)
`csw`	context switches
`icsw`	involuntary context switches
`migr`	thread migrations (to another processor)
`smtx`	spins on mutexes (lock not acquired on first try)
`srw`	spins on readers/writer locks (lock not acquired on first try)
`syscl`	system calls
`usr`	percent user time
`sys`	percent system time
`wt`	the I/O wait time is no longer calculated as a percentage of CPU time, and this statistic will always return zero.
`idl`	percent idle time

`sze`	number of processors in the requested processor set
`set`	processor set membership of each CPU

Options The following options are supported:

`-a`	Aggregate output by processor set. Sort the output by set. The default output is sorted by CPU number.
`-p`	Report processor set membership of each CPU. Sort the output by set. The default output is sorted by CPU number.
`-P` *set*	Display only those processors in the specified *set*.
`-q`	Suppress messages related to state changes.
interval	Report once each *interval* seconds.
count	Only print *count* reports.

Examples **EXAMPLE 1** Using `mpstat` to Generate User and System Operation Statistics

The following command generates processor statistics over a five–second interval in two reports. The command shows the processor set membership of each CPU. The default output is sorted by CPU number, aggregated by *processor set*, for user (`usr`) and system (`sys`) operations.

```
example% mpstat -ap 5 2


SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
  0    6   0  355  291  190  22    0    0    0   0    43   0   2  0  43   1
  1   24  17  534  207  200  70    1    0    2   0   600   4   1  0  84   2
  2   19   7  353  325  318  44    0    0    5   0   345   1   1  0  94   3
  3   36   2  149  237  236  14    0    0    4   0    97   0   0  0  98   2
SET minf mjf xcal intr ithr csw icsw migr smtx srw syscl usr sys wt idl sze
  0    1   0  720  405  304  55    0    0   18   0    12   0  15  0  81   1
  1    0  69 1955  230  200 313   33    4   41   9  7086  34  10  0  19   2
  2    0  46  685  314  300 203   11    0   54   1  5287  36   6  0  28   3
  3    0   0   14  386  384   0    0    0    0   0     0   0   0  0 100   2
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	See below.

Invocation is evolving. Human readable output is unstable.

See Also sar(1), iostat(1M), sar(1M), vmstat(1M), attributes(5)

Notes The sum of CPU utilization might vary slightly from 100 due to rounding errors in the production of a percentage figure.

The total time used for CPU processing is the sum of `usr` and `sys` output values, reported for user and system operations. The `idl` value reports the time that the CPU is idle for any reason other than pending disk I/O operations.

Run the `iostat` command with the `-x` option to report I/O service times in `svc_t` output. The `iostat` utility also reports the same `wt`, user (`us`), and system (`sy`) statistics. See iostat(1M) for more information.

When executing in a `zone` and if the pools facility is active, `mpstat`(1M) will only provide information for those processors which are a member of the processor set of the pool to which the `zone` is bound.

Name msgid – generate message IDs

Synopsis `/usr/sbin/msgid`

Description The `msgid` utility generates message IDs.

A message ID is a numeric identifier that uniquely identifies a message. Although the probability of two distinct messages having the same ID is high, this can be greatly reduced with the appropriate priority or facility.level designator (see `syslogd(1M)`). Specifically, the message ID is a hash signature on the message's unexpanded format string, generated by `STRLOG_MAKE_MSGID()` as defined in `<sys/strlog.h>`.

`syslogd(1M)` is a simple filter that takes strings as input and produces those same strings, preceded by their message IDs, as output. Every message logged by `syslogd`(1M) includes the message ID. The message ID is intended to serve as a small, language-independent identifier.

Examples **EXAMPLE 1** Using the msgid command to generate a message ID

The following example uses the `msgid` command to generate a message ID for the `echo` command.

```
example# echo hello | msgid205790 hello
```

EXAMPLE 2 Using the msgid command to generate a message catalog

The following example uses the `msgid` command to enumerate all of the messages in the binary `ufs`, to generate a message catalog.

```
example# strings /kernel/fs/ufs | msgid

137713 free:
    freeing free frag, dev:0x%lx, blk:%ld, cg:%d, ino:%lu, fs:%s
567420 ialloccg: block not in mapfs = %s
845546 alloc: %s: file system full
...
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `syslogd(1M)`, `attributes(5)`, `log(7D)`

Name mvdir – move a directory

Synopsis `/usr/sbin/mvdir` *dirname name*

Description `mvdir` moves directories within a file system. `dirname` must be a directory. If *name* does not exist, it will be created as a directory. If *name* does exist, and is a directory, `dirname` will be created as *name/dirname*. `dirname` and *name* may not be on the same path; that is, one may not be subordinate to the other. For example:

```
example% mvdir x/y x/z
```

is legal, but

```
example% mvdir x/y x/y/z
```

is not.

Operands

`dirname` The name of the directory that is to be moved to another directory in the filesystem.

name The name of the directory into which `dirname` is to be moved. If *name* does not exist, it will be created. It may not be on the same path as `dirname`.

Usage See `largefile(5)` for the description of the behavior of `mvdir` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status

`0` Successful operation.

`>0` Operation failed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `mkdir(1)`, `mv(1)`, `attributes(5)`, `largefile(5)`

Name named, in.named – Internet domain name server

Synopsis

```
named [-fgsVv] [-c config-file] [-d debug-level] [-m flag]
      [-n #cpus] [-p port] [-S #max-socks] [-t directory]
      [-u user] [-x cache-file] [-4 | -6]
```

Description The `named` utility is a Domain Name System (DNS) server, part of the BIND 9 distribution from ISC. For more information on the DNS, see RFCs 1033, 1034, and 1035.

When invoked without arguments, `named` reads the default configuration file `/etc/named.conf`, reads any initial data, and listens for queries.

`in.named` is a link to `named`.

Options The following options are supported:

`-4`
Use only IPv4 transport. By default, both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-6`
Use only IPv6 transport. By default, both IPv4 and IPv6 transports can be used. Options `-4` and `-6` are mutually exclusive.

`-c` *config-file*
Use *config-file* as the configuration file instead of the default `/etc/named.conf`. To ensure that reloading the configuration file continues to work after the server has changed its working directory due to to a possible *directory* option in the configuration file, *config-file* should be an absolute pathname.

`-d` *debug-level*
Set the daemon's debug level to *debug-level*. Debugging traces from `named` become more verbose as the debug level increases.

`-f`
Run the server in the foreground (that is, do not run as a daemon).

`-g`
Run the server in the foreground and force all logging to `stderr`.

`-m` *flag*
Turn on memory usage debugging flags. Possible flags are `usage`, `trace`, and `record`, `size`, and `mctx`. These correspond to the `ISC_MEM_DEBUG`*XXXX* flags described in `<isc/mem.h>`.

`-n` *#cpus*
Create *#cpus* worker threads to take advantage of multiple CPUs. If not specified, `named` will try to determine the number of CPUs present and create one thread per CPU. If it is unable to determine the number of CPUs, a single worker thread will be created.

`-p` *port*
Listen for queries on port *port*. If not specified, the default is port 53.

`-S` *#max-socks*
Allow `named` to use up to *#max-socks* sockets.

This option should be unnecessary for the vast majority of users. The use of this option could even be harmful, because the specified value might exceed the limitation of the underlying system API. It therefore should be set only when the default configuration causes exhaustion of file descriptors and the operational environment is known to support the specified number of sockets. Note also that the actual maximum number is normally a little smaller than the specified value because `named` reserves some file descriptors for its internal use.

`-s`
Write memory usage statistics to *stdout* on exit.

This option is mainly of interest to BIND 9 developers and might be removed or changed in a future release.

`-t` *directory*
Change the root directory using `chroot(2)` to *directory* after processing the command line arguments, but before reading the configuration file.

This option should be used in conjunction with the `-u` option, as chrooting a process running as root does not enhance security on most systems; the way `chroot()` is defined allows a process with root privileges to escape a `chroot` jail.

`-u` *user*
Set the real user ID using `setuid(2)` to *user* after completing privileged operations, such as creating sockets that listen on privileged ports.

`-V`
Report the version number and build options, and exit.

`-v`
Report the version number and exit.

`-x` *cache-file*
Load data from *cache-file* into the cache of the default view.

Do not use this option. It is of interest only to BIND 9 developers and might be removed or changed in a future release.

Extended Description

This section describes additional attributes of `named`.

SMF Properties

When starting named from the service management facility, `smf(5)`, `named` configuration is read from the service configuration repository. Use `svcprop(1)` to list the properties and `svccfg(1M)` to make changes.

The following application configuration properties are available to administrators:

`options/`*server*
Specifies the server executable to be used instead of the default server, `/usr/sbin/named`.

`options/`*configuration_file*
Specifies the configuration file to be used instead of the default, `/etc/named.conf`. A directory option might be specified in the configuration file. To ensure that reloading the configuration file continues to work in such a situation, *configuration_file* should be specified as an absolute pathname. This pathname should not include the *chroot_dir* pathname. This property is the equivalent of the `-c` option.

`options/`*ip_interfaces*
Specifies over which IP transport, IPv4 or IPv6, BIND will transmit. Possible values are `IPv4` or `IPv6`. Any other setting assumes `all`, the default. This property is the equivalent of command line option `-4` or `-6`

`options/`*listen_on_port*
Specifies the default UDP and TCP port to be used for listening to DNS requests. This property is the equivalent of the command line option `-p` *port*.

`options/`*debug_level*
Specifies the default debug level. The default is 0, which means no debugging. The higher the number the more verbose debug information becomes. Equivalent of the command line option `-d` *debug_level*.

`options/`*threads*
Specifies the number of CPU worker threads to create. The default of 0 causes `named` to try and determine the number of CPUs present and create one thread per CPU. Equivalent of command line option `-n` *#cpus*.

`options/`*chroot_dir*
Specifies the directory to be used as the root directory after processing SMF properties and the command line arguments but before reading the configuration file. Use this property when using a `chroot(2)` environment. Synonymous to command line option `-t` *pathname*.

When using `chroot(2)`, `named` is unable to disable itself when receiving `rndc(1M)` `stop` or `halt` commands. Instead, you must use the `svcadm(1M)` `disable` command.

In the event of a configuration error originating in one of the above SMF application options, `named` displays a message providing information about the error and the parameters that need correcting. The process then exits with exit code `SMF_EXIT_ERR_CONFIG`.

At startup, in the event of an error other than a configuration error, `named` exits with exit code `SMF_EXIT_ERR_FATAL`. Both of this code and `SMF_EXIT_ERR_CONFIG` cause the start method, `smf_method(5)`, to place the service in the maintenance state, which can be observed with the `svcs(1)` command `svcs -x`.

In addition to the properties listed above, the following property can be used to invoke `named` as a user other than root:

`start/`*user*
Specifies the identity of the user that is invoking `named`. See `smf_method(5)` and `chroot(2)`. Note that the user must have `solaris.smf.manage.bind` authorization. Without this role the `named` will be unable to manage its SMF FMRI and `named` will automatically be restarted by the SMF after an `rndc(1M)` `stop` or `halt` command. See `EXAMPLES` for a sequence of commands that establishes the correct authorization.

SIGNALS In routine operation, signals should not be used to control the nameserver; `rndc(1M)` should be used instead.

`SIGHUP`
Force a reload of the server.

`SIGINT`, `SIGTERM`
Shut down the server.

The result of sending any other signals to the server is undefined.

Configuration The `named` configuration file is too complex to describe in detail here. A list of configuration options is provided in the `named.conf` man page shipped with the BIND 9 distribution. A complete description is provided in the *BIND 9 Administrator Reference Manual*.

Examples

EXAMPLE 1 Configuring `named` to Transmit Only over IPv4 Networks

The following command sequence configures `named` such that it will transmit only over IPv4 networks.

```
# svccfg -s svc:network/dns/server:default setprop \
> options/ip_interfaces=IPv4
# svcadm refresh svc:network/dns/server:default
#
```

EXAMPLE 2 Listing Current Configuration File and Setting an Alternative File

The following sequence of commands lists the current `named` configuration file and sets an alternative file.

```
# svcprop -p options/configuration_file dns/server:default
/etc/named.conf
# svccfg -s dns/server:default setprop \
> options/configuration_file=/var/named/named.conf
# svcadm refresh dns/server:default
# svcprop -p options/configuration_file dns/server:default
/var/named/named.conf
```

EXAMPLE 3 Establishing Appropriate Authorization for `named`

To have `named` start with the `solaris.smf.manage.bind` authorization, perform the steps shown below.

Add the user `dnsadmin` to the `solaris.smf.manage.bind` role:

EXAMPLE 3 Establishing Appropriate Authorization for named *(Continued)*

```
# usermod -A solaris.smf.manage.bind dnsadmin
Observe effect of command:
# tail -1 /etc/user_attr
dnsadmin::::type=normal;auths=solaris.smf.manage.bind
```

Modify the service properties:

```
# svccfg
svc:> select svc:/network/dns/server:default
svc:/network/dns/server:default> setprop start/user = dnsadmin
svc:/network/dns/server:default> setprop start/group = dnsadmin
svc:/network/dns/server:default> exit
# svcadm refresh svc:/network/dns/server:default
# svcadm restart svc:/network/dns/server:default
```

Because only root has write access to create the default process-ID file, `/var/run/named/named.pid`, named must be configured to use an alternative path for the user `dnsadmin`. Here is an example of how to accomplish this:

```
# mkdir /var/named/tmp
# chown dnsadmin /var/named/tmp
```

Shown below is what you must add to `named.conf` to make use of the directory created above.

```
# head /etc/named.conf
options {
directory "/var/named";
pid-file "/var/named/tmp/named.pid";
};
```

Files

`/etc/named.conf`
default configuration file

`/var/run/named/named.pid`
default process-ID file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also

`svcs(1)`, `named-checkconf(1M)`, `named-checkzone(1M)`, `rndc(1M)`, `rndc-confgen(1M)`, `svcadm(1M)`, `svccfg(1M)`, `svcprop(1)`, `chroot(2)`, `setuid(2)`, `bind(3SOCKET)`, `attributes(5)`, `smf(5)`, `smf_method(5)`

RFC 1033, *RFC 1034*, *RFC 1035*

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

The `named.conf` man page shipped with the BIND 9 distribution

Name named-checkconf – named configuration file syntax checking tool

Synopsis

```
named-checkconf [-hjvz] [-t directory] filename
```

Description The `named-checkconf` utility checks the syntax, but not the semantics, of a specified configuration file.

Options The following options are supported:

`-h`
Display the usage summary and exit.

`-j`
When loading a zonefile, read the journal if it exists.

`-t` *directory*
Change the root directory to *directory* so that include directives in the configuration file are processed as if run by a named configuration whose root directory has been similarly changed.

`-v`
Print the version of the `named-checkconf` program and exit.

`-z`
Perform a test load of the master zones found in `named.conf`.

Operands The following operands are supported:

filename
The name of the configuration file to be checked. If not specified, it defaults to `/etc/named.conf`.

Exit Status

`0`
No errors were detected.

`1`
An error was detected.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also `named(1M)`, `named-checkzone(1M)`, `attributes(5)`

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name named-checkzone, named-compilezone – zone file validity checking or converting tool

Synopsis

```
named-checkzone [-Ddhjqv] [-c class] [-F format] [-f format]
     [-i mode] [-k mode] [-M mode] [-m mode] [-n mode]
     [-o filename] [-S mode] [-s style] [-t directory]
     [-W mode] [-w directory] zonename filename

named-compilezone [-Ddjqv] [-C mode] [-c class] [-F format]
     [-f format] [-i mode] [-k mode] [-m mode] [-n mode]
     [-o filename] [-s style] [-t directory]
     [-W mode] [-w directory] zonename filename
```

Description The `named-checkzone` utility checks the syntax and integrity of a zone file. It performs the same checks as `named(1M)` does when loading a zone. The `named-checkzone` utility is useful for checking zone files before configuring them into a name server.

`named-compilezone` is similar to `named-checkzone`, differing in that it always dumps the zone contents to a specified file in a specified format. Additionally, it applies stricter check levels by default, since the dump output will be used as an actual zone file loaded by `named(1M)`. Unless manually specified otherwise, the check levels must be at least as strict as those specified in the `named` configuration file.

Options For either or both utilities, the following options are supported:

`-c` *class*
Specify the class of the zone. If not specified, "IN" is assumed.

`-D`
Dump zone file in canonical format.

`-d`
Enable debugging.

`-F` *format*
Specify the format of the output file specified. Possible formats are `text` (default) and `raw`. For `named-checkzone`, this does not cause any effects unless it dumps the zone contents.

`-f` *format*
Specify the format of the zone file. Possible formats are `text` (default) and `raw`.

`-h`
Display usage message for `named-checkzone`.

`-i` *mode*
Perform post-load zone integrity checks. Possible modes are `full` (default), `full-sibling`, `local`, `local-sibling`, and `none`.

Mode `full` checks that MX records refer to the A or AAAA record (both in-zone and out-of-zone hostnames). Mode `local` checks only MX records that refer to in-zone hostnames.

Mode `full` checks that SRV records refer to the A or AAAA record (both in-zone and out-of-zone hostnames). Mode `local` checks only SRV records that refer to in-zone hostnames.

Mode `full` checks that delegation NS records refer to A or AAAA record (both in-zone and out-of-zone hostnames). It also checks that glue address records in the zone match those advertised by the child. Mode `local` checks only NS records that refer to in-zone hostnames or check that some required glue exists, that is, when the nameserver is in a child zone.

Mode `full-sibling` and `local-sibling` disable sibling glue checks, but are otherwise the same as `full` and `local`, respectively.

Mode `none` disables the checks.

`-k` *mode*
Perform "check-name" checks with the specified failure mode. Possible modes are `fail` (default for `named-compilezone`), `warn` (default for `named-checkzone`) and `ignore`.

`-j`
Read the journal, if it exists, when loading the zone file.

`-M` *mode*
Check if an MX record refers to a `CNAME`. Possible modes are `fail`, `warn` (default) and `ignore`.

`-m` *mode*
Specify whether MX records should be checked to see if they are addresses. Possible modes are `fail`, `warn` (default) and `ignore`.

`-n` *mode*
Specify whether NS records should be checked to see if they are addresses. Possible modes are `fail` (default for `named-compilezone`), `warn` (default for `named-checkzone`) and `ignore`.

`-o` *filename*
Write zone output to *filename*. If filename is `-` (a hyphen), then write to standard out. The hyphen mandatory for `named-compilezone`

`-q`
Run in quiet mode, reporting only the exit status.

`-S` *mode*
Check if a SRV record refers to a `CNAME`. Possible modes are `fail`, `warn` (default) and `ignore`.

`-s` *style*
Specify the style of the dumped zone file. Possible styles are `full` (default) and `relative`. The `full` format is most suitable for processing automatically by a separate script. The `relative` format is more human-readable and is thus suitable for editing by hand. For

`named-checkzone` this option does not cause any effects unless it dumps the zone contents. It also has no effect if the output format is not text.

`-t` *directory*
`chroot` to directory so that include directives in the configuration file are processed as if run by a similarly `chrooted` `named`.

`-v`
Print the version of the `named-checkzone` program and exit.

`-W` *mode*
Specify whether to check for non-terminal wildcards. Non-terminal wildcards are almost always the result of a failure to understand the wildcard matching algorithm (RFC 1034). Possible modes are `warn` (default) and `ignore`.

`-w` *directory*
`chdir` to *directory* so that relative filenames in master file `$INCLUDE` directives work. This is similar to the directory clause in `named.conf`.

Operands The following operands are supported:

filename
The name of the zone file.

zonename
The domain name of the zone being checked.

Exit Status

`0`
No errors were detected.

`1`
An error was detected.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dns/bind
Interface Stability	Volatile

See Also `named(1M)`, `named-checkconf(1M)`, `attributes(5)`

RFC 1035

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name ncaconfd – Solaris Network Cache and Accelerator (NCA) configuration daemon

Synopsis `/usr/lib/inet/ncaconfd [-al ]` *interface1* `[`*interface2* `...]`

Description Use the `ncaconfd` utility to set up NCA on a system. At boot time, the `ncakmod` initialization script reads in `nca.if(4)` to determine on which interface(s) NCA should run. `ncaconfd` then sets up the interface.

`ncaconfd` also operates as a daemon if the `nca_active` key is set to enabled in `ncakmod.conf(4)` file. In this case, `ncaconfd` will continue as a daemon after all the NCA interfaces have been set up, listening for routing changes. The changes are then passed to NCA to control which interface NCA should use to make active outgoing TCP connnections.

Options The following options are supported:

`-a` Enable active connections.

`-l` Enable logging.

Files `/etc/nca/ncakmod.conf`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWncau
Interface Stability	Evolving

See Also `nca(1)`, `ncakmod(1)`, `nca.if(4)`, `ncakmod.conf(4)`, `attributes(5)`

Name ncheck – generate a list of path names versus i-numbers

Synopsis

```
ncheck [-F FSType] [-V] [generic_options]
     [-o FSType-specific_options] [special]...
```

Description `ncheck` with no options generates a path-name versus i-number list of all files on *special*. If *special* is not specified on the command line the list is generated for all *special*s in `/etc/vfstab` which have a numeric `fsckpass`. *special* is the raw device on which the file system exists.

Options

`-F` Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by finding an entry in the table that has a numeric `fsckpass` field and an `fsckdev` that matches *special*.

`-V` Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option may be used to verify and validate the command line.

generic_options Options that are commonly supported by most FSType-specific command modules. The following options are available:

- `-i` *i-list* Limit the report to the files on the *i-list* that follows. The *i-list* must be separated by commas with no intervening spaces.
- `-a` Print the names "." and ". ." which are ordinarily suppressed.
- `-s` Report only special files and files with set-user-ID mode. This option may be used to detect violations of security policy.

`-o` Specify *FSType-specific_options* in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the FSType-specific module of the command.

Usage See `largefile(5)` for the description of the behavior of `ncheck` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files `/etc/vfstab` list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `vfstab(4)`, `attributes(5)`, `largefile(5)` Manual pages for the FSType-specific modules of `ncheck`

Notes This command may not be supported for all *FSTypes*.

Name ncheck_ufs – generate pathnames versus i-numbers for ufs file systems

Synopsis

```
ncheck -F ufs [generic_options] [-o m] [special]...
```

Description `ncheck -F ufs` generates a pathname versus i-number list of files for the `ufs` file system residing on *special*. Names of directory files are followed by '`/.`'.

Options See `ncheck(1M)` for the list of *generic_options* supported.

`-o` Specify `ufs` file system specific options. The available option is:

`m` Print mode information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `ff(1M)`, `ncheck(1M)`, `attributes(5)`

Diagnostics When the file system structure is improper, '`??`' denotes the "parent" of a parentless file and a pathname beginning with '`. . .`' denotes a loop.

Name ndd – get and set driver configuration parameters

Synopsis `ndd [-set] driver parameter [value]`

Description `ndd` gets and sets selected configuration parameters in some kernel drivers. Currently, `ndd` only supports the drivers that implement the TCP/IP Internet protocol family. Each driver chooses which parameters to make visible using `ndd`. Since these parameters are usually tightly coupled to the implementation, they are likely to change from release to release. Some parameters may be read-only.

If the `-set` option is omitted, `ndd` queries the named *driver*, retrieves the value associated with the specified *parameter*, and prints it. If the `-set` option is given, `ndd` passes *value*, which must be specified, down to the named *driver* which assigns it to the named *parameter*.

By convention, drivers that support `ndd` also support a special read-only *parameter* named "?" which can be used to list the parameters supported by the driver.

Examples **EXAMPLE 1** Getting Parameters Supported By The TCP Driver

To see which parameters are supported by the TCP driver, use the following command:

```
example% ndd /dev/tcp \?
```

The parameter name "?" may need to be escaped with a backslash to prevent its being interpreted as a shell meta character.

The following command sets the value of the parameter *ip_forwarding* in the dual stack IP driver to zero. This disables IPv4 packet forwarding.

```
example% ndd -set /dev/ip ip_forwarding 0
```

Similarly, in order to disable IPv6 packet forwarding, the value of parameter *ip6_forwarding*

```
example% ndd -set /dev/ip ip6_forwarding 0
```

To view the current IPv4 forwarding table, use the following command:

```
example% ndd /dev/ip ipv4_ire_status
```

To view the current IPv6 forwarding table, use the following command:

```
example% ndd /dev/ip ipv6_ire_status
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also nca(1), ioctl(2), attributes(5), arp(7P), ip(7P), ip6(7P), tcp(7P), udp(7P)

Notes The parameters supported by each driver may change from release to release. Like programs that read `/dev/kmem`, user programs or shell scripts that execute `ndd` should be prepared for parameter names to change.

The `ioctl()` command that `ndd` uses to communicate with drivers is likely to change in a future release. User programs should avoid making dependencies on it.

The meanings of many `ndd` parameters make sense only if you understand how the driver is implemented.

Name netservices – enable or disable network services

Synopsis

```
netservices open

netservices limited
```

Description The `netservices` command uses the Solaris service management facility, `smf(5)`, to control services that accept over the network from remote clients.

When `netservices` is invoked with the `limited` command-line argument, all network services except the secure shell daemon, sshd(1M), are either disabled or constrained to respond to local requests only.

Invoking netservices with the `open` command-line argument enables a large set of network services, as in previous releases of Solaris.

To customize the configuration set enabled by `netservices`, use `svcadm(1M)` to enable or disable individual services. Use `svccfg(1M)` to set properties that determine whether a service accepts input from remote clients. See the man pages for individual services for the names of service instances and their properties.

Note that the `netservices` command has an interface stability of Obsolete.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Obsolete

See Also `svcadm(1M)`, `svccfg(1M)`, `sshd(1M)`, `attributes(5)`, `smf(5)`

Name netstat – show network status

Synopsis

```
netstat [-anvR] [-f address_family] [-P protocol]

netstat -g [-nv] [-f address_family]

netstat -p [-n] [-f address_family]

netstat -s [-f address_family] [-P protocol]
     [interval [count]]

netstat -m [-v] [interval [count]]

netstat -i [-I interface] [-an] [-f address_family]
     [interval [count]]

netstat -r [-anvR] [-f address_family | filter]

netstat -M [-ns] [-f address_family]

netstat -D [-I interface] [-f address_family]
```

Description The `netstat` command displays the contents of certain network-related data structures in various formats, depending on the options you select.

The `netstat` command has the several forms shown in the SYNOPSIS section, above, listed as follows:

- The first form of the command (with no required arguments) displays a list of active sockets for each protocol.
- The second, third, and fourth forms (`-g`, `-p`, and `-s` options) display information from various network data structures.
- The fifth form (`-m` option) displays STREAMS memory statistics.
- The sixth form (`-i` option) shows the state of the interfaces.
- The seventh form (`-r` option) displays the routing table.
- The eighth form (`-M` option) displays the multicast routing table.
- The ninth form (`-D` option) displays the state of DHCP on one or all interfaces.

These forms are described in greater detail below.

With no arguments (the first form), `netstat` displays connected sockets for `PF_INET`, `PF_INET6`, and `PF_UNIX`, unless modified otherwise by the `-f` option.

Options `-a`

Show the state of all sockets, all routing table entries, or all interfaces, both physical and logical. Normally, listener sockets used by server processes are not shown. Under most conditions, only interface, host, network, and default routes are shown and only the status of physical interfaces is shown.

`-f` *address_family*
Limit all displays to those of the specified *address_family*. The value of *address_family* can be one of the following:

`inet` For the `AF_INET` address family showing IPv4 information.

`inet6` For the `AF_INET6` address family showing IPv6 information.

`unix` For the `AF_UNIX` address family.

`-f` *filter*
With `-r` only, limit the display of routes to those matching the specified filter. A filter rule consists of a *keyword*:*value* pair. The known keywords and the value syntax are:

`af:{inet|inet6|unix|`*number*`}`	Selects an address family. This is identical to `-f` *address_family* and both syntaxes are supported.
`{inif|outif}:{`*name*`|`*ifIndex*`|any|none}`	Selects an input or output interface. You can specify the interface by name (such as `hme0`) or by `ifIndex` number (for example, `2`). If `any` is used, the filter matches all routes having a specified interface (anything other than null). If `none` is used, the filter matches all routes having a null interface. Note that you can view the index number (*ifIndex*) for an interface with the `-a` option of `ifconfig(1M)`.
`{src|dst}:{`*ip-address*`[/`*mask*`]|any|none}`	Selects a source or destination IP address. If specified with a mask length, then any routes with matching or longer (more specific) masks are selected. If `any` is used, then all but addresses but 0 are selected. If `none` is used, then address 0 is selected.
`flags:[+ -]?[ABDGHLMSU]+`	Selects routes tagged with the specified flags. By default, the flags as specified must be set in order to match. With a leading `+`, the flags specified must be set but others are ignored. With a leading `-`, the flags specified must not be set and others are permitted.

You can specify multiple instances of `-f` to specify multiple filters. For example:

```
% netstat -nr -f outif:hme0 -f outif:hme1 -f dst:10.0.0.0/8
```

The preceding command displays routes within network 10.0.0.0/8, with mask length 8 or greater, and an output interface of either `hme0` or `hme1`, and excludes all other routes.

`-g`
Show the multicast group memberships for all interfaces. If the `-v` option is included, source-specific membership information is also displayed. See DISPLAYS, below.

`-i`
Show the state of the interfaces that are used for IP traffic. Normally this shows statistics for the physical interfaces. When combined with the `-a` option, this will also report information for the logical interfaces. See `ifconfig(1M)`.

`-m`
Show the STREAMS memory statistics.

`-n`
Show network addresses as numbers. `netstat` normally displays addresses as symbols. This option may be used with any of the display formats.

`-p`
Show the net to media tables. See DISPLAYS, below.

`-r`
Show the routing tables. Normally, only interface, host, network, and default routes are shown, but when this option is combined with the `-a` option, all routes will be displayed, including cache.

`-s`
Show per-protocol statistics. When used with the `-M` option, show multicast routing statistics instead. When used with the `-a` option, per-interface statistics will be displayed, when available, in addition to statistics global to the system. See DISPLAYS, below.

`-v`
Verbose. Show additional information for the sockets, STREAMS memory statistics, routing table, and multicast group memberships.

`-I` *interface*
Show the state of a particular interface. *interface* can be any valid interface such as `hme0` or `eri0`. Normally, the status and statistics for physical interfaces are displayed. When this option is combined with the `-a` option, information for the logical interfaces is also reported.

`-M`
Show the multicast routing tables. When used with the `-s` option, show multicast routing statistics instead.

`-P` *protocol*
Limit display of statistics or state of all sockets to those applicable to *protocol*. The protocol can be one of `ip`, `ipv6`, `icmp`, `icmpv6`, `icmp`, `icmpv6`, `igmp`, `udp`, `tcp`, `rawip`. `rawip` can also be specified as `raw`. The command accepts protocol options only as all lowercase.

`-D`
Show the status of DHCP configured interfaces.

`-R`

This modifier displays extended security attributes for sockets and routing table entries. The `-R` modifier is available only if the system is configured with the Solaris Trusted Extensions feature.

With `-r` only, this option displays the routing entries' gateway security attributes. See `route(1M)` for more information on security attributes.

When displaying socket information using the first form of the commmand, this option displays additional information for Multi-Level Port(MLP) sockets. This includes:

- The label for the peer if the the socket is connected.
- The following flags can be appended to the socket's "State" output:

  `P` The socket is a MLP on zone-private IP addresses.

  `S` The socket is a MLP on IP addresses shared between zones.

Operands

interval Display statistics accumulated since last display every *interval* seconds, repeating forever, unless *count* is specified. When invoked with *interval*, the first row of netstat output shows statistics accumulated since last reboot.

The following options support *interval*: `-i`, `-m`, `-s` and `-Ms`. Some values are configuration parameters and are just redisplayed at each interval.

count Display interface statistics the number of times specified by *count*, at the interval specified by *interval*.

Displays

Active Sockets (First Form)

The display for each active socket shows the local and remote address, the send and receive queue sizes (in bytes), the send and receive windows (in bytes), and the internal state of the protocol.

The symbolic format normally used to display socket addresses is either:

`hostname.`*port*

when the name of the host is specified, or

network`.`*port*

if a socket address specifies a network but no specific host.

The numeric host address or network number associated with the socket is used to look up the corresponding symbolic hostname or network name in the *hosts* or *networks* database.

If the network or hostname for an address is not known, or if the `-n` option is specified, the numerical network address is shown. Unspecified, or "wildcard", addresses and ports appear as an asterisk (`*`). For more information regarding the Internet naming conventions, refer to `inet(7P)` and `inet6(7P)`.

For SCTP sockets, because an endpoint can be represented by multiple addresses, the verbose option (`-v`) displays the list of all the local and remote addresses.

TCP Sockets The possible state values for TCP sockets are as follows:

`BOUND`	Bound, ready to connect or listen.
`CLOSED`	Closed. The socket is not being used.
`CLOSING`	Closed, then remote shutdown; awaiting acknowledgment.
`CLOSE_WAIT`	Remote shutdown; waiting for the socket to close.
`ESTABLISHED`	Connection has been established.
`FIN_WAIT_1`	Socket closed; shutting down connection.
`FIN_WAIT_2`	Socket closed; waiting for shutdown from remote.
`IDLE`	Idle, opened but not bound.
`LAST_ACK`	Remote shutdown, then closed; awaiting acknowledgment.
`LISTEN`	Listening for incoming connections.
`SYN_RECEIVED`	Initial synchronization of the connection under way.
`SYN_SENT`	Actively trying to establish connection.
`TIME_WAIT`	Wait after close for remote shutdown retransmission.

SCTP Sockets The possible state values for SCTP sockets are as follows:

`CLOSED`	Closed. The socket is not being used.
`LISTEN`	Listening for incoming associations.
`ESTABLISHED`	Association has been established.
`COOKIE_WAIT`	`INIT` has been sent to the peer, awaiting acknowledgment.
`COOKIE_ECHOED`	State cookie from the INIT-ACK has been sent to the peer, awaiting acknowledgement.
`SHUTDOWN_PENDING`	`SHUTDOWN` has been received from the upper layer, awaiting acknowledgement of all outstanding `DATA` from the peer.

SHUTDOWN_SENT	All outstanding data has been acknowledged in the SHUTDOWN_SENT state. SHUTDOWN has been sent to the peer, awaiting acknowledgement.
SHUTDOWN_RECEIVED	SHUTDOWN has been received from the peer, awaiting acknowledgement of all outstanding DATA.
SHUTDOWN_ACK_SENT	All outstanding data has been acknowledged in the SHUTDOWN_RECEIVED state. SHUTDOWN_ACK has been sent to the peer.

Network Data Structures (Second Through Fifth Forms)

The form of the display depends upon which of the `-g`, `-m`, `-p`, or `-s` options you select.

`-g` Displays the list of multicast group membership.

`-m` Displays the memory usage, for example, STREAMS mblks.

`-p` Displays the net to media mapping table. For IPv4, the address resolution table is displayed. See `arp(1M)`. For IPv6, the neighbor cache is displayed.

`-s` Displays the statistics for the various protocol layers.

The statistics use the MIB specified variables. The defined values for `ipForwarding` are:

forwarding(1)	Acting as a gateway.
not-forwarding(2)	Not acting as a gateway.

The IPv6 and ICMPv6 protocol layers maintain per-interface statistics. If the `-a` option is specified with the `-s` option, then the per-interface statistics as well as the total sums are displayed. Otherwise, just the sum of the statistics are shown.

For the second, third, and fourth forms of the command, you must specify at least `-g`, `-p`, or `-s`. You can specify any combination of these options. You can also specify `-m` (the fifth form) with any set of the `-g`, `-p`, and `-s` options. If you specify more than one of these options, `netstat` displays the information for each one of them.

Interface Status (Sixth Form)

The interface status display lists information for all current interfaces, one interface per line. If an interface is specified using the `-I` option, it displays information for only the specified interface.

The list consists of the interface name, `mtu` (maximum transmission unit, or maximum packet size)(see `ifconfig(1M)`), the network to which the interface is attached, addresses for each interface, and counter associated with the interface. The counters show the number of input packets, input errors, output packets, output errors, and collisions, respectively. For Point-to-Point interfaces, the Net/Dest field is the name or address on the other side of the link.

If the `-a` option is specified with either the `-i` option or the `-I` option, then the output includes names of the physical interface(s), counts for input packets and output packets for each logical interface, plus additional information.

If the `-n` option is specified, the list displays the IP address instead of the interface name.

If an optional *interval* is specified, the output will be continually displayed in *interval* seconds until interrupted by the user or until *count* is reached. See OPERANDS.

The physical interface is specified using the `-I` option. When used with the *interval* operand, output for the `-I` option has the following format:

```
input    eri0          output        input          (Total)   output
packets  errs  packets errs  colls   packets  errs  packets  errs   colls
227681   0     659471  1     502     261331   0     99597    1      502
10       0     0       0     0       10       0     0        0      0
8        0     0       0     0       8        0     0        0      0
10       0     2       0     0       10       0     2        0      0
```

If the input interface is not specified, the first interface of address family `inet` or `inet6` will be displayed.

Routing Table (Seventh Form)

The routing table display lists the available routes and the status of each. Each route consists of a destination host or network, and a gateway to use in forwarding packets. The *flags* column shows the status of the route. These flags are as follows:

`U` Indicates route is up.

`G` Route is to a gateway.

`H` Route is to a host and not a network.

`M` Redundant route established with the `-multirt` option.

`S` Route was established using the `-setsrc` option.

`D` Route was created dynamically by a redirect.

If the `-a` option is specified, there will be routing entries with the following flags:

`A` Combined routing and address resolution entries.

`B` Broadcast addresses.

`L` Local addresses for the host.

Interface routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface.

The `use` column displays the number of packets sent using a combined routing and address resolution (`A`) or a broadcast (`B`) route. For a local (`L`) route, this count is the number of packets

received, and for all other routes it is the number of times the routing entry has been used to create a new combined route and address resolution entry.

The *interface* entry indicates the network interface utilized for the route.

Multicast Routing Tables (Eighth Form)

The multicast routing table consists of the virtual interface table and the actual routing table.

DHCP Interface Information (Ninth Form)

The DHCP interface information consists of the interface name, its current state, lease information, packet counts, and a list of flags.

The states correlate with the specifications set forth in *RFC 2131*.

Lease information includes:

- when the lease began;
- when lease renewal will begin; and
- when the lease will expire.

The flags currently defined include:

`BOOTP` The interface has a lease obtained through `BOOTP` (IPv4 only).

`BUSY` The interface is busy with a DHCP transaction.

`PRIMARY` The interface is the primary interface. See `dhcpinfo(1)` and `ifconfig(1M)`.

`FAILED` The interface is in failure state and must be manually restarted.

Packet counts are maintained for the number of packets sent, the number of packets received, and the number of lease offers declined by the DHCP client. All three counters are initialized to zero and then incremented while obtaining a lease. The counters are reset when the period of lease renewal begins for the interface. Thus, the counters represent either the number of packets sent, received, and declined while obtaining the current lease, or the number of packets sent, received, and declined while attempting to obtain a future lease.

Files

`/etc/default/inet_type` `DEFAULT_IP` setting

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`arp(1M)`, `dhcpinfo(1)`, `dhcpagent(1M)`, `ifconfig(1M)`, `iostat(1M)`, `kstat(1M)`, `mibiisa(1M)`, `savecore(1M)`, `vmstat(1M)`, `hosts(4)`, `inet_type(4)`, `networks(4)`, `protocols(4)`, `services(4)`, `attributes(5)`, `dhcp(5)`, `kstat(7D)`, `inet(7P)`, `inet6(7P)`

Droms, R., *RFC 2131, Dynamic Host Configuration Protocol*, Network Working Group, March 1997.

Droms, R. *RFC 3315, Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. Cisco Systems. July 2003.

Notes When displaying interface information, `netstat` honors the `DEFAULT_IP` setting in `/etc/default/inet_type`. If it is set to `IP_VERSION4`, then `netstat` will omit information relating to IPv6 interfaces, statistics, connections, routes and the like.

However, you can override the `DEFAULT_IP` setting in `/etc/default/inet_type` on the command-line. For example, if you have used the command-line to explicitly request IPv6 information by using the `inet6` address family or one of the IPv6 protocols, it will override the `DEFAULT_IP` setting.

If you need to examine network status information following a kernel crash, use the `mdb(1)` utility on the `savecore(1M)` output.

The `netstat` utility obtains TCP statistics from the system by opening `/dev/tcp` and issuing queries. Because of this, `netstat` might display an extra, unused connection in `IDLE` state when reporting connection status.

Previous versions of `netstat` had undocumented methods for reporting kernel statistics published using the `kstat(7D)` facility. This functionality has been removed. Use `kstat(1M)` instead.

`netstat` restricts its output to information that is relevant to the zone in which `netstat` runs. (This is true for both shared-IP and exclusive-IP zones.)

Name netstrategy – return network configuration information

Synopsis

```
/sbin/netstrategy
```

Description The `netstrategy` command determines the network configuration strategy in use on a system and returns information in a form that is easily consumable by a script. The command returns three tokens:

<root filesystem type> <primary interface> <network config strategy>

These tokens are described as follows:

<root filesystem type>
Type of filesystem that contains the bootable kernel, as would be specified in the *fstype* column of the `mnttab(4)`.

<primary interface>
Name of the primary network interface. For a diskless machine, this is the interface used to load the kernel.

<network config strategy>
The means by which a system obtains its IP address for booting. This can be one of `rarp`, `dhcp`, or `none`.

The `netstrategy` command is not intended for use on a command line.

Options The `netstrategy` command has no options.

Exit Status

`0` Success.

`!=0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr
Interface Stability	Evolving

See Also `ifconfig(1M)`, `mnttab(4)`, `attributes(5)`

Name newaliases – rebuild the data base for the mail aliases file

Synopsis

```
newaliases
```

Description `newaliases` rebuilds the random access data base for the mail aliases file `/etc/mail/aliases`.

`newaliases` accepts all the flags that `sendmail(1M)` accepts. However, most of these flags have no effect, except for the `-C` option and three of the `Processing Options` that can be set from a configuration file with the `-o` option:

`-C` */path/to/alt/config/file*	Use alternate configuration file.
`-oA`*file*	Specify possible alias files.
`-oL`*n*	Set the default log level to *n*. Defaults to `9`.
`-on`	Validate the RHS of aliases when rebuilding the `aliases(4)` database.

`newaliases` runs in verbose mode (`-v` option) automatically.

Examples **EXAMPLE 1** Running the `newaliases` Command

The following command runs `newaliases` on an alias file different from the `/etc/mail/aliases` default in `sendmail(1M)`:

```
example% newaliases -oA/path/to/alternate/alias/file
```

Exit Status `newaliases` returns an exit status describing what it did. The codes are defined in `/usr/include/sysexits.h`.

`EX_OK`	Successful completion on all addresses.
`EX_NOUSER`	User name not recognized.
`EX_UNAVAILABLE`	Catchall. Necessary resources were not available.
`EX_SYNTAX`	Syntax error in address.
`EX_SOFTWARE`	Internal software error, including bad arguments.
`EX_OSERR`	Temporary operating system error, such as “cannot fork”.
`EX_NOHOST`	Host name not recognized.
`EX_TEMPFAIL`	Message could not be sent immediately, but was queued.

Files

`/etc/aliases`	Symbolic link to `/etc/mail/aliases`
`/etc/mail/aliases.pag`	
`/etc/mail/aliases.dir`	`ndbm` files maintained by `newaliases`
`/etc/mail/aliases.db`	Berkeley DataBase file maintained by newaliases

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsndmu

See Also sendmail(1M), aliases(4), attributes(5)

Name newfs – construct a UFS file system

Synopsis `newfs [-NSBTv] [`*mkfs-options*`] `*raw-device*

Description `newfs` is a "friendly" front-end to the `mkfs(1M)` program for making UFS file systems on disk partitions. `newfs` calculates the appropriate parameters to use and calls `mkfs`.

If run interactively (that is, standard input is a tty), `newfs` prompts for confirmation before making the file system.

If the `-N` option is not specified and the inodes of the device are not randomized, `newfs` calls `fsirand(1M)`.

You must be super-user or have appropriate write privileges to use this command, except when creating a UFS file system on a *diskette*. See `EXAMPLES`.

Creating a Multiterabyte UFS File System

Keep the following limitations in mind when creating a multiterabyte UFS file system:

- *nbpi* is set to 1 Mbyte unless you specifically set it higher. You cannot set *nbpi* lower than 1 Mbyte on a multiterabyte UFS file system.
- *fragsize* is set equal to *bsize*.

Options The following options are supported:

`-N` Print out the file system parameters that would be used to create the file system without actually creating the file system. `fsirand(1M)` is not called here.

`-S` Sends to stdout a human-readable version of the superblock that would be used to create a filesystem with the specified configuration parameters.

`-B` Sends to stdout a binary (machine-readable) version of the superblock that would be used to create a filesystem with the specified configuration parameters.

`-T` Set the parameters of the file system to allow eventual growth to over a terabyte in total file system size. This option sets *fragsize* to be the same as *bsize*, and sets *nbpi* to 1 Mbyte, unless the `-i` option is used to make it even larger. If you use the `-f` or `-i` options to specify a *fragsize* or *nbpi* that is incompatible with this option, the user-supplied value of *fragsize* or *nbpi* is ignored.

`-v` Verbose. `newfs` prints out its actions, including the parameters passed to `mkfs`.

mkfs-options Options that override the default parameters are:

`-a` *apc* The number of alternate sectors per cylinder to reserve for bad block replacement for SCSI devices only. The default is `0`.

This option is not applicable for disks with EFI labels and is ignored.

`-b` *bsize*
: The logical block size of the file system in bytes, either 4096 or 8192. The default is `8192`. The sun4u architecture does not support the `4096` block size.

`-c` *cgsize*
: The number of cylinders per cylinder group, ranging from `16` to `256`. The default is calculated by dividing the number of sectors in the file system by the number of sectors in a gigabyte. Then, the result is multiplied by `32`. The default value is always between `16` and `256`.

  `mkfs` can override this value. See `mkfs_ufs(1M)` for details.

  This option is not applicable for disks with EFI labels and is ignored.

`-C` *maxcontig*
: The maximum number of logical blocks, belonging to one file, that are allocated contiguously. The default is calculated as follows:

  `maxcontig =` *disk drive maximum transfer size / disk block size*

  If the disk drive's maximum transfer size cannot be determined, the default value for `maxcontig` is calculated from kernel parameters as follows:

  If `maxphys` is less than `ufs_maxmaxphys`, which is typically 1 Mbyte, then `maxcontig` is set to `maxphys`. Otherwise, `maxcontig` is set to `ufs_maxmaxphys`.

  You can set `maxcontig` to any positive integer value.

  The actual value will be the lesser of what has been specified and what the hardware supports.

  You can subsequently change this parameter by using `tunefs(1M)`.

`-d` *gap*
: Rotational delay. This option is obsolete in the Solaris 10 release. The value is always set to `0`, regardless of the input value.

`-f` *fragsize*
: The smallest amount of disk space in bytes that can be allocated to a file. *fragsize* must be a power of 2 divisor of *bsize*, where:

bsize / *fragsize* is 1, 2, 4, or 8.

This means that if the logical block size is `4096`, legal values for *fragsize* are `512`, `1024`, `2048`, and `4096`. When the logical block size is `8192`, legal values are `1024`, `2048`, `4096`, and `8192`. The default value is `1024`.

For file systems greater than 1 terabyte or for file systems created with the `-T` option, *fragsize* is forced to match block size (*bsize*).

`-i` *nbpi*

The number of bytes per inode, which specifies the density of inodes in the file system. The number is divided into the total size of the file system to determine the number of inodes to create.

This value should reflect the expected average size of files in the file system. If fewer inodes are desired, a larger number should be used. To create more inodes, a smaller number should be given. The default for *nbpi* is as follows:

```
Disk size                Density

Less than 1GB            2048
Less than 2GB            4096
Less than 3GB            6144
3GB to 1 Tbyte           8192
Greater than 1 Tbyte
   or created with -T    1048576
```

The number of inodes can increase if the file system is expanded with the `growfs` command.

`-m` *free*

The minimum percentage of free space to maintain in the file system, between 0% and 99%, inclusively. This space is off-limits to users. Once the file system is filled to this threshold, only the super-user can continue writing to the file system.

The default is ((64 Mbytes/partition size) * 100), rounded down to the nearest integer and limited between 1% and 10%, inclusively.

This parameter can be subsequently changed using the `tunefs(1M)` command.

`-n` *nrpos*	The number of different rotational positions in which to divide a cylinder group. The default is `8`. This option is not applicable for disks with EFI labels and is ignored.
`-o` *space* | *time*	The file system can either be instructed to try to minimize the *time* spent allocating blocks, or to try to minimize the *space* fragmentation on the disk. The default is *time*. This parameter can subsequently be changed with the `tunefs(1M)` command.
`-r` *rpm*	The rotational speed of the disk in revolutions per minute. The default is driver- or device-specific. Note that you specify *rpm* for `newfs` and *rps* for `mkfs`. This option is not applicable for disks with EFI labels and is ignored.
`-s` *size*	The size of the file system in sectors. The default is to use the entire partition.
`-t` *ntrack*	The number of tracks per cylinder on the disk. The default is taken from the disk label. This option is not applicable for disks with EFI labels and is ignored.

Operands The following operands are supported:

raw-device	The name of a raw special device residing in the `/dev` directory (for example, `/dev/rdsk/c0t0d0s6`) on which to create the file system.

Usage See `largefile(5)` for the description of the behavior of `newfs` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Displaying the Parameters for the Raw Special Device

The following example verbosely displays the parameters for the raw special device, `c0t0d0s6`. It does not actually create a new file system:

```
example# newfs -Nv /dev/rdsk/c0t0d0s6
mkfs -F ufs -o N /dev/rdsk/c0t0d0s6 1112940 54 15 8192 1024 16 10 60
2048 t 0 –1 8 /dev/rdsk/c0t0d0s6: 1112940 sectors in
1374 cylinders of 15 tracks, 54 sectors 569.8MB in 86 cyl
groups (16 c/g, 6.64MB/g, 3072 i/g) super-block backups
(for fsck -b #) at:
```

EXAMPLE 1 Displaying the Parameters for the Raw Special Device *(Continued)*

```
32, 13056, 26080, 39104, 52128, 65152, 78176, 91200, 104224, . . .
```

EXAMPLE 2 Creating a UFS File System

The following example creates a UFS file system on a diskette that is managed by Volume Manager.

```
example% newfs /vol/dev/aliases/floppy0
newfs: construct a new file system /vol/dev/aliases/floppy0: (y/n)? y
/vol/dev/aliases/floppy0: 2880 sectors in 80 cylinders of 2 tracks,
18 sectors 1.4MB in 5 cyl groups (16 c/g, 0.28MB/g, 128 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
32, 640, 1184, 1792, 2336, . . .
```

EXAMPLE 3 Creating a UFS File System That Will Eventually Be Grown to a Multiterabyte UFS File System

The following example creates a UFS file system that will eventually be grown to a multiterabyte UFS file system.

This command creates a 800-Gbyte file system on the volume, `/dev/md/rdsk/d99`.

```
# newfs -T /dev/md/rdsk/d99
newfs: construct a new file system /dev/md/rdsk/d99: (y/n)? y
   /dev/md/rdsk/d99: 1677754368 sectors in 45512 cylinders of
   144 tracks, 256 sectors
   819216.0MB in 1821 cyl groups (25 c/g, 450.00MB/g, 448 i/g) . . .
```

Then, if you increase the volume size for this file system, you can use the `growfs` command to expand the file system. The file system is grown to 1.2 terabytes in this example:

```
# growfs -v /dev/md/rdsk/d99
/usr/lib/fs/ufs/mkfs -G /dev/md/rdsk/d99 2516631552 /dev/md/rdsk/d99:
   2516631552 sectors in 68268 cylinders of 144 tracks, 256  sectors
   1228824.0MB in 2731 cyl groups (25 c/g, 450.00MB/g, 448 i/g). . .
```

Exit Status

The following exit values are returned:

`0` The operation was successful.

`1`, `10` Usage error or internal error. A message is output to `STDERR` explaining the error.

Other exit values may be returned by `mkfs(1M)`, which is called by `newfs`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also fsck(1M), fsck_ufs(1M), fsirand(1M), mkfs(1M), mkfs_ufs(1M), tunefs(1M), attributes(5), largefile(5), ufs(7FS)

Diagnostics

`newfs: No such file or directory`	The device specified does not exist, or a disk partition was not specified.
special`: cannot open`	You must write access to the device to use this command.

Name newkey – create a new Diffie-Hellman key pair in the publickey database

Synopsis

```
newkey -h hostname [-s nisplus | nis | files | ldap]

newkey -u username [-s nisplus | nis | files | ldap]
```

Description `newkey` establishes new public keys for users and machines on the network. These keys are needed when using secure RPC or secure NFS service.

`newkey` prompts for a password for the given *username* or `hostname` and then creates a new public/secret Diffie-Hellman 192 bit key pair for the user or host. The secret key is encrypted with the given password. The key pair can be stored in the `/etc/publickey` file, the NIS `publickey` map, or the NIS+ `cred.org_dir` table.

`newkey` consults the `publickey` entry in the name service switch configuration file (see `nsswitch.conf(4)`) to determine which naming service is used to store the secure RPC keys. If the `publickey` entry specifies a unique name service, `newkey` will add the key in the specified name service. However, if there are multiple name services listed, `newkey` cannot decide which source to update and will display an error message. The user is required to specify the source explicitly with the `-s` option.

In the case of NIS, `newkey` should be run by the superuser on the master NIS server for that domain. In the case of NIS+, `newkey` should be run by the superuser on a machine which has permission to update the `cred.org_dir` table of the new user/host domain.

In the case of NIS+, `nisaddcred(1M)` should be used to add new keys. `newkey` cannot be used to create keys other than 192-bit Diffie-Hellman.

In the case of LDAP, `newkey` should be run by the superuser on a machine that also recognizes the directory manager's bind distinguished name (DN) and password to perform an LDAP update for the host.

Options

`-h` *hostname* — Create a new public/secret key pair for the privileged user at the given `hostname`. Prompts for a password for the given `hostname`.

`-u` *username* — Create a new public/secret key pair for the given *username*. Prompts for a password for the given *username*.

`-s nisplus`
`-s nis`
`-s files`
`-s ldap` — Update the database in the specified source: `nisplus` (for NIS+), `nis` (for NIS), `files`, or `ldap` (LDAP). Other sources may be available in the future.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also chkey(1), keylogin(1), nisaddcred(1M), nisclient(1M), nsswitch.conf(4), publickey(4), attributes(5)

Notes NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nfs4cbd – NFS Version 4 callback daemon

Synopsis

```
/usr/lib/nfs/nfs4cbd
```

Description The `nfs4cbd` daemon manages communication endpoints for the NFS Version 4 protocol callback program. `nfs4cbd` runs on the NFS Version 4 client and creates a listener port for each transport over which callbacks can be sent.

The `nfs4cbd` daemon is provided for the exclusive use of the NFS version 4 client.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfscu

See Also `svcs(1)`, `mount_nfs(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

Notes The `nfs4cbd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nfs/cbd
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

If it is disabled, it will be enabled by `mount_nfs(1M)` and `automountd(1M)` on the first NFSv4 mount, unless its `application/auto_enable` property is set to `false`.

This daemon might not exist in a future release of Solaris.

Name nfsd – NFS daemon

Synopsis

```
/usr/lib/nfs/nfsd [-a] [-c #_conn] [-l listen_backlog]
     [-p protocol] [-t device] [nservers]
```

Description `nfsd` is the daemon that handles client file system requests. Only users with `{PRIV_SYS_NFS}` and sufficient privileges to write to `/var/run` can run this daemon.

The `nfsd` daemon is automatically invoked using `share(1M)` with the `-a` option.

By default, `nfsd` starts over the TCP and UDP transports for versions 2 and 3. By default, it starts over the TCP for version 4. You can change this with the `-p` option.

A previously invoked `nfsd` daemon started with or without options must be stopped before invoking another `nfsd` command.

Administrators wanting to change startup parameters for nfsd should, as root, make changes in the `/etc/default/nfs` file. See `nfs(4)`.

Options The following options are supported:

`-a` Start a NFS daemon over all available connectionless and connection-oriented transports, including UDP and TCP. Equivalent of setting the `NFSD_PROTOCOL` parameter to `ALL` in the `nfs` file.

`-c` *#_conn* This sets the maximum number of connections allowed to the NFS server over connection-oriented transports. By default, the number of connections is unlimited. Equivalent of the `NFSD_MAX_CONNECTIONS` parameter in the `nfs` file.

`-l` Set connection queue length for the NFS TCP over a connection-oriented transport. The default value is 32 entries. Equivalent of the `NFSD_LISTEN_BACKLOG` parameter in the `nfs` file.

`-p` *protocol* Start a NFS daemon over the specified protocol. Equivalent of the `NFSD_PROTOCOL` parameter in the `nfs` file.

`-t` *device* Start a NFS daemon for the transport specified by the given device. Equivalent of the `NFSD_DEVICE` parameter in the `nfs` file.

Operands The following operands are supported:

nservers This sets the maximum number of concurrent NFS requests that the server can handle. This concurrency is achieved by up to *nservers* threads created as needed in the kernel. *nservers* should be based on the load expected on this server. 16 is the usual number of *nservers*. If *nservers* is not specified, the maximum number of concurrent NFS requests will default to 1. Equivalent of the `NFSD_SERVERS` parameter in the `nfs` file.

Usage If the NFS_PORTMON variable is set in /etc/system, then clients are required to use privileged ports (ports < IPPORT_RESERVED) to get NFS services. This variable is equal to zero by default. This variable has been moved from the "nfs" module to the "nfssrv" module. To set the variable, edit the /etc/system file and add this entry:

```
set nfssrv:nfs_portmon = 1
```

Exit Status

0	Daemon started successfully.
1	Daemon failed to start.

Files

.nfs*XXX*	Client machine pointer to an open-but-unlinked file.
/etc/default/nfs	Contains startup parameters for nfsd.
/etc/system	System configuration information file.
/var/nfs/v4_state /var/nfs/v4_oldstate	Directories used by the server to manage client state information. These directories should not be removed.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfssu

See Also ps(1), svcs(1), mountd(1M), share(1M), svcadm(1M), nfs(4), sharetab(4), system(4), attributes(5), smf(5)

System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)

Notes Manually starting and restarting nfsd is not recommended. If it is necessary to do so, use svcadm to enable or disable the nfs service (svc:/network/nfs/server). If it is disabled, it will be enabled by share_nfs(1M), unless its application/auto_enable property is set to false. See the *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*, and svcadm(1M) for more information.

The nfsd service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/nfs/server
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

If nfsd is killed with SIGTERM, it will not be restarted by the service management facility. Instead, nfsd can be restarted by other signals, such as SIGINT.

Name nfslogd – nfs logging daemon

Synopsis `/usr/lib/nfs/nfslogd`

Description The `nfslogd` daemon provides operational logging to the Solaris NFS server. It is the `nfslogd` daemon's job to generate the activity log by analyzing the RPC operations processed by the NFS server. The log will only be generated for file systems exported with logging enabled. This is specified at file system export time by means of the `share_nfs(1M)` command.

NFS server logging is not supported on Solaris machines that are using NFS Version 4.

Each record in the log file includes a time stamp, the IP address (or hostname if it can be resolved) of the client system, the file or directory name the operation was performed on, and the type of operation. In the basic format, the operation can either be an input (i) or output (o) operation. The basic format of the NFS server log is compatible with the log format generated by the Washington University FTPd daemon. The log format can be extended to include directory modification operations, such as `mkdir`, `rmdir`, and `remove`. The extended format is not compatible with the Washington University FTPd daemon format. See `nfslog.conf(4)` for details.

The NFS server logging mechanism is divided in two phases. The first phase is performed by the NFS kernel module, which records raw RPC requests and their results in work buffers backed by permanent storage. The location of the work buffers is specified in the `/etc/nfs/nfslog.conf` file. Refer to `nfslog.conf(4)` for more information. The second phase involves the `nfslogd` user-level daemon, which periodically reads the work buffers, interprets the raw RPC information, groups related RPC operations into single transaction records, and generates the output log. The `nfslogd` daemon then sleeps waiting for more information to be logged to the work buffers. The amount of time that the daemon sleeps can be configured by modifying the `IDLE_TIME` parameter in `/etc/default/nfslogd`. The work buffers are intended for internal consumption of the `nfslogd` daemon.

NFS operations use file handles as arguments instead of path names. For this reason the `nfslogd` daemon needs to maintain a database of file handle to path mappings in order to log the path name associated with an operation instead of the corresponding file handle. A file handle entry is added to the database when a client performs a lookup or other NFS operation that returns a file handle to the client.

Once an NFS client obtains a file handle from a server, it can hold on to it for an indefinite time, and later use it as an argument for an NFS operation on the file or directory. The NFS client can use the file handle even after the server reboots. Because the database needs to survive server reboots, it is backed by permanent storage. The location of the database is specified by the *fhtable* parameter in the `/etc/nfs/nfslog.conf` file. This database is intended for the internal use of the `nfslogd` daemon.

In order to keep the size of the file handle mapping database manageable, `nfslogd` prunes the database periodically. It removes file handle entries that have not been accessed in more than a specified amount of time. The `PRUNE_TIMEOUT` configurable parameter in

/etc/default/nfslogd specifies the interval length between successive runs of the pruning process. A file handle record will be removed if it has not been used since the last time the pruning process was executed. Pruning of the database can effectively be disabled by setting the PRUNE_TIMEOUT as high as INT_MAX.

When pruning is enabled, there is always a risk that a client may have held on to a file handle longer than the PRUNE_TIMEOUT and perform an NFS operation on the file handle after the matching record in the mapping database had been removed. In such case, the pathname for the file handle will not be resolved, and the log will include the file handle instead of the pathname.

There are various configurable parameters that affect the behavior of the nfslogd daemon. These parameters are found in /etc/default/nfslogd and are described below:

UMASK	Sets the file mode for the log files, work buffer files and file handle mapping database.
MIN_PROCESSING_SIZE	Specifies the minimum size, in bytes, that the buffer file must reach before processing the work information and writing to the log file. The value of MIN_PROCESSING_SIZE must be between 1 and ulimit.
IDLE_TIME	Specifies the amount of time, in seconds, the daemon should sleep while waiting for more information to be placed in the buffer file. IDLE_TIME also determines how often the configuration file will be reread. The value of IDLE_TIME must be between 1 and INT_MAX.
MAX_LOGS_PRESERVE	The nfslogd periodically cycles its logs. MAX_LOGS_PRESERVE specifies the maximum number of log files to save. When MAX_LOGS_PRESERVE is reached, the oldest files will be overwritten as new log files are created. These files will be saved with a numbered extension, beginning with filename.0. The oldest file will have the highest numbered extension up to the value configured for MAX_LOGS_PRESERVE. The value of MAX_LOGS_PRESERVE must be between 1 and INT_MAX.
CYCLE_FREQUENCY	Specifies how often, in hours, the log files are cycled. CYCLE_FREQUENCY is used to insure that the log files do not get too large. The value of CYCLE_FREQUENCY must be between 1 and INT_MAX.
MAPPING_UPDATE_INTERVAL	Specifies the time interval, in seconds, between updates of the records in the file handle to path mapping tables. Instead of updating the atime of a record each time that record is accessed, it is only updated if it has aged based on this

parameter. The record access time is used by the pruning routine to determine whether the record should be removed from the database. The value of this parameter must be between 1 and `INT_MAX`.

`PRUNE_TIMEOUT` Specifies when a database record times out, in hours. If the time that elapsed since the record was last accessed is greater than `PRUNE_TIMEOUT` then the record can be pruned from the database. The default value for `PRUNE_TIMEOUT` is 168 hours (7 days). The value of `PRUNE_TIMEOUT` must be between 1 and `INT_MAX`.

Exit Status The following exit values are returned:

`0` Daemon started successfully.

`1` Daemon failed to start.

Files `/etc/nfs/nfslogtab`

`/etc/nfs/nfslog.conf`

`/etc/default/nfslogd`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfssu

See Also `share_nfs(1M)`, `nfslog.conf(4)`, `attributes(5)`

Name nfsmapid – NFS user and group id mapping daemon

Synopsis

```
/usr/lib/nfs/nfsmapid
```

Description The `nfsmapid` daemon maps to and from NFS version 4 `owner` and `owner_group` identification attributes and local UID and GID numbers used by both the NFS version 4 client and server.

`nfsmapid` uses the `passwd` and `group` entries in the `/etc/nsswitch.conf` file to direct how it performs the mappings.

The `nfsmapid` daemon has no external, customer-accessible interfaces. You can, however, administratively configure `nfsmapid` in one of the following ways:

- Specify the `NFSMAPID_DOMAIN` parameter in `nfs(4)`
- Specify the `_nfsv4idmapdomain` DNS resource record.

Please refer to the *System Administration Guide: Network Services* for further details.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfscu

See Also `svcs(1)`, `automountd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `mount_nfs(1M)`, `passmgmt(1M)`, `svcadm(1M)`, `share_nfs(1M)`, `userdel(1M)`, `usermod(1M)`, `nfs(4)`, `attributes(5)`, `smf(5)`

System Administration Guide: Network Services

Notes The `nfsmapid` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nfs/mapid
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

If it is disabled, it will be enabled by `mount_nfs(1M)`, `share_nfs(1M)`, and `automountd(1M)`, unless its `application/auto_enable` property is set to `false`.

`nfsmapid` caches a user's UID and GID. If a user subsequently changes a UID or GID, using one of the utilities listed below, the `nfsmapid` cache becomes stale. At this point, any NFS operation that gets or set attributes will result in the exchange of this stale information. To resolve this situation, restart `nfsmapid`, as follows:

```
# svcadm restart svc:/network/nfs/mapid:default
```

The utilities that allow you to change UID and GID are:

- usermod(1M)
- userdel(1M)
- groupmod(1M)
- groupdel(1M)
- passmgmt(1M)

The nfsmapid daemon might not exist in a future release of Solaris.

Name nfsstat – NFS statistics

Synopsis

```
nfsstat [-cnrsza] [-v version] [interval [count]]

nfsstat -m [pathname]...
```

Description `nfsstat` displays statistical information about the NFS and RPC (Remote Procedure Call), interfaces to the kernel. It can also be used to reinitialize this information. If no options are given the default is as follows:

```
nfsstat -csnra
```

The default displays everything, but reinitializes nothing.

Options

`-a` Display `NFS_ACL` information.

`-c` Display client information. Only the client side NFS, RPC, and NFS_ACL information is printed. Can be combined with the `-n`, `-r`, and `-a` options to print client side NFS, RPC, and NFS_ACL information only.

`-m` [*pathname...*] Display statistics for each NFS mounted file system. If *pathname* is not specified, displays statistics for all NFS mounted file systems. If *pathname* is specified, displays statistics for the NFS mounted file systems indicated by *pathname*.

This includes the server name and address, mount flags, current read and write sizes, the retransmission count, the attribute cache timeout values, failover information, and the timers used for dynamic retransmission. The dynamic retransmission timers are displayed only where dynamic retransmission is in use. By default, NFS mounts over the TCP protocols and NFS Version 3 mounts over either TCP or UDP do not use dynamic retransmission.

If you specify the `-m` option, this is the only option that `nfsstat` uses. If you specify other options with `-m`, you receive an error message alerting that the `-m` flag cannot be combined with other options.

`-n` Display NFS information. NFS information for both the client and server side are printed. Can be combined with the `-c` and `-s` options to print client or server NFS information only.

`-r` Display RPC information.

`-s` Display server information.

`-v` *version* Specify which NFS version for which to print statistics. When followed by the optional *version* argument, (`2`|`3`|`4`), specifies statistics for that version. By default, prints statistics for all versions.

`-z` Zero (reinitialize) statistics. This option is for use by the super user only, and can be combined with any of the above options to zero particular sets of statistics after printing them.

Operands The following operands are supported:

count Display only count reports

interval Report once each interval seconds.

pathname Specify the pathname of a file in an NFS mounted file system for which statistics are to be displayed.

Displays The server RPC display includes the following fields:

`calls` The total number of RPC calls received.

`badcalls` The total number of calls rejected by the RPC layer (the sum of `badlen` and `xdrcall` as defined below).

`nullrecv` The number of times an RPC call was not available when it was thought to be received.

`badlen` The number of RPC calls with a length shorter than a minimum-sized RPC call.

`xdrcall` The number of RPC calls whose header could not be XDR decoded.

`dupchecks` The number of RPC calls that looked up in the duplicate request cache.

`dupreqs` The number of RPC calls that were found to be duplicates.

The server NFS display shows the number of NFS calls received (`calls`) and rejected (`badcalls`), and the counts and percentages for the various calls that were made.

The server NFS_ACL display shows the counts and percentages for the various calls that were made.

The client RPC display includes the following fields:

`calls` The total number of RPC calls made.

`badcalls` The total number of calls rejected by the RPC layer.

`badxids` The number of times a reply from a server was received which did not correspond to any outstanding call.

`timeouts` The number of times a call timed out while waiting for a reply from the server.

`newcreds` The number of times authentication information had to be refreshed.

`badverfs` The number of times the call failed due to a bad verifier in the response.

`timers`	The number of times the calculated time-out value was greater than or equal to the minimum specified time-out value for a call.
`cantconn`	The number of times the call failed due to a failure to make a connection to the server.
`nomem`	The number of times the call failed due to a failure to allocate memory.
`interrupts`	The number of times the call was interrupted by a signal before completing.
`retrans`	The number of times a call had to be retransmitted due to a timeout while waiting for a reply from the server. Applicable only to RPC over connection-less transports.
`cantsend`	The number of times a client was unable to send an RPC request over a connectionless transport when it tried to do so.

The client NFS display shows the number of calls sent and rejected, as well as the number of times a `CLIENT` handle was received (`clgets`), the number of times the `CLIENT` handle cache had no unused entries (`cltoomany`), as well as a count of the various calls and their respective percentages.

The client NFS_ACL display shows the counts and percentages for the various calls that were made.

The `-m` option includes information about mount flags set by mount options, mount flags internal to the system, and other mount information. See `mount_nfs(1M)`.

The following mount flags are set by mount options:

`sec`	`sec` has one of the following values:	
	`none`	No authentication.
	`sys`	UNIX-style authentication (UID, GID).
	`short`	Short hand UNIX–style authentication.
	`dh`	`des`–style authentication (encrypted timestamps).
	`krb5`	`kerberos v5`–style authentication.
	`krb5i`	`kerberos v5`–style authentication with integrity.
	`krb5p`	`kerberos v5`–style authentication with privacy.
`hard`	Hard mount.	
`soft`	Soft mount.	
`intr`	Interrupts allowed on hard mount.	
`nointr`	No interrupts allowed on hard mount.	

`noac`	Client is not caching attributes.
`rsize`	Read buffer size in bytes.
`wsize`	Write buffer size in bytes.
`retrans`	NFS retransmissions.
`timeo`	Initial NFS timeout, in tenths of a second.
`nocto`	No close-to-open consistency.
`llock`	Local locking being used (no lock manager).
`grpid`	System V group id inheritance.
`rpctimesync`	RPC time sync.

The following mount flags are internal to the system:

`printed`	"Not responding" message printed.
`down`	Server is down.
`dynamic`	Dynamic transfer size adjustment.
`link`	Server supports links.
`symlink`	Server supports symbolic links.
`readdir`	Use `readdir` instead of `readdirplus`.
`acl`	Server supports NFS_ACL.

The following flags relate to additional mount information:

`vers`	NFS version.
`proto`	Protocol.

The `-m` option also provides attribute cache timeout values. The following fields in `-m` ouput provide timeout values for attribute cache:

`acregmin`	Minimum seconds to hold cached file attributes.
`acregmax`	Maximum seconds to hold cached file attributes.
`acdirmin`	Minimum seconds to hold cached directory attributes.
`acdirmax`	Maximum seconds to hold cached directory attributes.

The following fields in `-m` output provide failover information:

`noresponse`	How many times servers have failed to respond.
`failover`	How many times a new server has been selected.

`remap` How many times files have been re-evaluated to the new server.

`currserver` Which server is currently providing NFS service. See the *System Administration Guide: IP Services* for additional details.

The fields in `-m` output shown below provide information on dynamic retransmissions. These items are displayed only where dynamic retransmission is in use.

`srtt` The value for the smoothed round-trip time, in milliseconds.

`dev` Estimated deviation, in milliseconds.

`cur` Current backed-off retransmission value, in milliseconds.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfscu

See Also `mount_nfs(1M)`, `attributes(5)`

Solaris 10 Installation Guide: Basic Installations

System Administration Guide: IP Services

Name nisaddcred – create NIS+ credentials

Synopsis

```
nisaddcred [-p principal] [-P nis_principal]
     [-l login_password] auth_type [domain_name]

nisaddcred -r [nis_principal] [domain_name]
```

Description The `nisaddcred` command is used to create security credentials for NIS+ principals. NIS+ credentials serve two purposes. The first is to provide authentication information to various services; the second is to map the authentication service name into a NIS+ principal name.

When the `nisaddcred` command is run, these credentials get created and stored in a table named `cred.org_dir` in the default NIS+ domain. If *domain_name* is specified, the entries are stored in the `cred.org_dir` of the specified domain. The specified domain must either be the one to which you belong, or one in which you are authenticated and authorized to create credentials, that is, a subdomain. Note that the credentials of normal users must be stored in the same domain as their passwords.

It is simpler to add credentials using `nisclient(1M)`, because it obtains the required information itself. `nispopulate(1M)` is used for "bulk" updates and can also be used to add credentials for entries in the `hosts` and the `passwd` NIS+ tables.

NIS+ principal names are used in specifying clients that have access rights to NIS+ objects. For more details, refer to the "Principal Names" subsection of the `NIS+(1)` manual page. See `nischmod(1)`, `nischown(1)`, `nis_objects(3NSL)`, and `nis_groups(3NSL)`. Various other services can also implement access control based on these principal names.

The `cred.org_dir` table is organized as follows:

cname	auth_type	auth_name	public_data	private_data
user1.foo.com.	LOCAL	2990	10,102,44	
user1.foo.com.	DES	unix.2990@foo.com	098...819	3b8...ab2
user1.foo.com.	DHmmm-n	unix.2990@foo.com	248...428	a42...f32

The `cname` column contains a canonical representation of the NIS+ principal name. By convention, this name is the login name of a user, or the host name of a machine, followed by a dot ('.') followed by the fully qualified "home" domain of that principal. For users, the home domain is defined to be the domain where their DES credentials are kept. For hosts, their home domain is defined to be the domain name returned by the `domainname(1M)` command executed on that host.

There are two basic types of *auth_type* entries in the `cred.org_dir` table, those with authentication type LOCAL, and those with authentication type DES, *auth_type*, specified on the command line in upper or lower case, should be either *local* or `des`.

However, the `cred.org_dir` table may also be used to hold data for other values of *auth_type*. Currently, this is limited to the mechanisms listed on the `nisauthconf(1M)` man page, for which the `nisaddcred` *auth_type* argument is the same as the name of the mechanism. These mechanisms use a modified form of Secure RPC, and they are similar to the DES authentication type.

If the *auth_type* is `des`, and other authentication mechanisms are configured with `nisauthconf(1M)`, then credential entries are added or updated for each mechanism configured. To only add or update 1992-bit Diffie Hellman credentials, that is, those with the *auth_type* of DES, use dh192-0 on the command line. If there are no authentication mechanisms configured, using `des` on the command line will only add or update 192-bit Diffie Hellman credentials.

Entries of type LOCAL are used by the NIS+ service to determine the correspondence between fully qualified NIS+ principal names and users identified by UIDs in the domain containing the `cred.org_dir` table. This correspondence is required when associating requests made using the `AUTH_SYS` RPC authentication flavor (see `rpc_clnt_auth(3NSL)`) to a NIS+ principal name. It is also required for mapping a UID in one domain to its fully qualified NIS+ principal name whose home domain may be elsewhere. The principal's credentials for any authentication flavor may then be sought for within the `cred.org_dir` table in the principal's home domain (extracted from the principal name). The same NIS+ principal may have LOCAL credential entries in more than one domain. Only users, and not machines, have LOCAL credentials. In their home domain, users of NIS+ should have both types of credentials.

The *auth_name* associated with the LOCAL type entry is a UID that is valid for the principal in the domain containing the `cred.org_dir` table. This may differ from that in the principal's home domain. The public information stored in *public_data* for this type contains a list of GIDs for groups in which the user is a member. The GIDs also apply to the domain in which the table resides. There is no private data associated with this type. Neither a UID nor a principal name should appear more than once among the LOCAL entries in any one `cred.org_dir` table.

The DES *auth_type* is used for Secure RPC authentication (see `secure_rpc(3NSL)`).

The authentication name associated with the DES *auth_type* is a Secure RPC *netname*. A Secure RPC netname has the form `unix.id@`*domain.com,* where *domain* must be the same as the domain of the principal. For principals that are users the `id` must be the UID of the principal in the principal's home domain. For principals that are hosts, the `id` is the host's name. In Secure RPC, processes running under effective UID 0 (root) are identified with the host principal. Unlike LOCAL, there cannot be more than one DES credential entry for one NIS+ principal in the NIS+ namespace.

The public information in an entry of authentication type DES is the public key for the principal. The private information in this entry is the private key of the principal encrypted by the principal's network password.

User clients of NIS+ should have credentials of both types in their home domain. In addition, a principal must have a LOCAL entry in the `cred.org_dir` table of each domain from which the principal wishes to make authenticated requests. A client of NIS+ that makes a request from a domain in which it does not have a LOCAL entry will be unable to acquire DES credentials. A NIS+ service running at security level 2 or higher will consider such users unauthenticated and assign them the name *nobody* for determining access rights.

This command can only be run by those NIS+ principals who are authorized to add or delete the entries in the `cred` table.

If credentials are being added for the caller itself, `nisaddcred` automatically performs a keylogin for the caller.

You can list the `cred` entries for a particular principal with `nismatch(1)`.

The `cred.org_dir` NIS+ table replaces the maps *publickey.byname* and *netid.byname* used in NIS (YP).

Options The following options are supported:

`-p` *principal* — The name *principal* specifies the name of the principal as defined by the naming rules for that specific mechanism. For example, LOCAL credential names are supplied with this option by including a string specifying a UID. For DES credentials, the name should be a Secure RPC netname of the form `unix.id@`*domain.com,* as described earlier. If the `-p` option is not specified, the *auth_name* field is constructed from the effective UID of the current process and the name of the local domain.

`-P` *nis_principal* — Use the NIS+ principal name *nis_principal*. This option should be used when creating LOCAL or DES credentials for users whose home domain is different than the local machine's default domain.

Whenever the `-P` option is not specified, `nisaddcred` constructs a principal name for the entry as follows. When it is not creating an entry of type LOCAL, `nisaddcred` calls `nis_local_principal`, which looks for an existing LOCAL entry for the effective UID of the current process in the `cred.org_dir` table and uses the associated principal name for the new entry. When creating an entry of authentication type LOCAL, `nisaddcred` constructs a default NIS+ principal name by taking the login name of the effective UID for its own process, and appending to it a dot ('.') followed by the local machine's default domain. If the caller is a superuser, the machine name is used instead of the login name.

`-l` *login_password* — Use the *login_password* specified as the password to encrypt the secret key for the credential entry. This overrides the prompting for a password from the shell. This option is intended for administration

scripts only. Prompting guarantees not only that no one can see your password on the command line using ps(1) but it also checks to make sure you have not made any mistakes. *login_password* does not really have to be the user's password but if it is, it simplifies logging in.

`-r` [*nis_principal*] Remove all credentials associated with the principal *nis_principal* from the `cred.org_dir` table. This option can be used when removing a client or user from the system. If *nis_principal* is not specified the default is to remove credentials for the current *user*. If *domain_name* is not specified, the operation is executed in the default NIS+ domain.

Examples

EXAMPLE 1 Adding the LOCAL and DES Credentials

The following examples illustrate how to add the LOCAL and DES credentials for some user, `user1`, with a UID of `2990`, who is an NIS+ user principal in the `some.domain.com.` NIS+ domain:

```
example% nisaddcred -p 2990 -P user1.some.domain.com. local
```

Note that credentials are always added in the `cred.org_dir` table in the domain where `nisaddcred` is run, unless *domain_name* is specified as the last parameter on the command line. If credentials are being added from the domain server for its clients, then *domain_name* should be specified. The caller should have adequate permissions to create entries in the `cred.org_dir` table.

The system administrator can add a DES credential for the same user, using the following example:

```
example% nisaddcred -p unix.2990@some.domain.com -P user1.some.domain.com. des
```

Please note that DES credentials can be added only after the LOCAL credentials have been added. Also, if the system is configured to use more than one authentication mechanism, credentials will be made for each mechanism configured. See nisauthconf(1M).

Note that the secure RPC netname does not end with a dot ('.') while the NIS+ principal name, specified with the `-P` option, does. This command should be executed from a machine in the same domain as is the user.

The following example shows how to add a machine's DES credentials in the same domain:

```
example% nisaddcred -p unix.foo@some.domain.com -P foo.some.domain.com. des
```

Please note that no LOCAL credentials are needed in this case.

The following example illustrates how to add a NIS+ workstation's principal DES credential:

```
example% nisaddcred -p unix.host1@sub.some.domain.com \
     -P newhost.sub.some.domain.com. des sub.some.domain.com.
```

EXAMPLE 1 Adding the LOCAL and DES Credentials *(Continued)*

This format is particularly useful if you are running this command from a server which is in a higher domain than sub.some.domain.com. Without the last option for domain name, `nisaddcred` would fail because it would attempt to use the default domain of some.domain.com.

The following example illustrates adding DES credentials without being prompted for the root login password:

```
example% nisaddcred -p unix.2990@some.domain.com \
     -P user1.some.domain.com. -l login_password des
```

The following example shows how to add a credential for a user using a specific authentication mechanism that was previously configured with `nisauthconf(1M)`. See `nisauthconf(1M)` for a list of the valid values of *auth_type*:

```
example% nisaddcred -p unix.2990@some.domain.com \
     -P user.1.some.domain.com dh640-0
```

The password should be the same for all the credentials that belong to the user. Otherwise, only the credentials encrypted with the user's password will be used at login, and the user will have to run `chkey(1)` using the `-p` option.

The following example shows how to add a DES credential when other authentication mechanisms are configured on the system:

```
example% nisaddcred -p unix.2990@some.domain.com \
     -P user1.some.domain.com dh192-0
```

Exit Status

The following exit values are returned:

`0` Successful operation.

`1` Operation failed.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also

`chkey(1)`, `keylogin(1)`, `NIS+(1)`, `nischmod(1)`, `nischown(1)`, `nismatch(1)`, `nistbladm(1)`, `ps(1)`, `domainname(1M)`, `nisclient(1M)`, `nispopulate(1M)`, `nis_groups(3NSL)`, `nis_local_names(3NSL)`, `nis_objects(3NSL)`, `rpc_clnt_auth(3NSL)`, `secure_rpc(3NSL)`, `attributes(5)`

Notes NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisaddent – create NIS+ tables from corresponding /etc files or NIS maps

Synopsis

```
/usr/lib/nis/nisaddent [-D defaults] [-Paorv] [-t table] type
     [nisdomain]

/usr/lib/nis/nisaddent [-D defaults] [-Paprmov] -f file
     [-t table] type [nisdomain]

/usr/lib/nis/nisaddent [-D defaults] [-Parmv] [-t table] -y ypdomain
     [-Y map] type [nisdomain]

/usr/lib/nis/nisaddent -d [-AMoq] [-t table] type
     [nisdomain]
```

Description `nisaddent` creates entries in NIS+ tables from their corresponding `/etc` files and NIS maps. This operation is customized for each of the standard tables that are used in the administration of Solaris systems. The `type` argument specifies the type of the data being processed. Legal values for this type are one of `aliases`, `bootparams`, `ethers`, `group`, `hosts`, `ipnodes`, `netid`, `netmasks`, `networks`, `passwd`, `protocols`, `publickey`, `rpc`, `services`, `shadow`, or `timezone` for the standard tables, or `key-value` for a generic two-column (key, value) table. For a site specific table, which is not of `key-value` type, one can use `nistbladm(1)` to administer it.

The NIS+ tables should have already been created by `nistbladm(1)`, `nissetup(1M)`, or `nisserver(1M)`.

It is easier to use `nispopulate(1M)` instead of `nisaddent` to populate the system tables.

By default, `nisaddent` reads from the standard input and adds this data to the NIS+ table associated with the `type` specified on the command line. An alternate NIS+ table may be specified with the `-t` option. For type `key-value`, a table specification is required.

Note that the *data* type can be different than the table name (`-t`). For example, the automounter tables have `key-value` as the table type.

Although, there is a *shadow* data type, there is no corresponding *shadow* table. Both the shadow and the passwd data is stored in the `passwd` table itself.

Files may be processed using the `-f` option, and NIS version 2 (YP) maps may be processed using the `-y` option. The merge option is not available when reading data from standard input.

When a *ypdomain* is specified, the `nisaddent` command takes its input from the `dbm` files for the appropriate NIS map (`mail.aliases`, `bootparams`, `ethers.byaddr`, `group.byname`, `hosts.byaddr`, `hosts.byname`, `ipnodes.byaddr`,`ipnodes.byname`, `netid.byname`, `netmasks.byaddr`, `networks.byname`, `passwd.byname`, `protocols.byname`, `publickey.byname`, `rpc.bynumber`, `services.byname`, or `timezone.byname`). An alternate NIS map may be specified with the `-Y` option. For type `key-value`, a map specification is required. The map must be in the `/var/yp/`*ypdomain* directory on the local machine. Note that *ypdomain* is case sensitive. `ypxfr(1M)` can be used to get the NIS maps.

If a *nisdomain* is specified, `nisaddent` operates on the NIS+ table in that NIS+ domain, otherwise the default domain is used.

In terms of performance, loading up the tables is fastest when done through the dbm files (`-y`).

To accommodate other credential entries used by other authentication mechanisms stored in the `cred.org_dir` table, the `publickey` dump output has been modified to include a special algorithm `type` field. This format is incompatible with older versions of `nisaddent`. To produce dumps that can be read by older versions of `nisaddent`, or to load dumps created by such older versions, use the `-o` option.

Options The following options are supported:

`-a` Add the file or map to the NIS+ table without deleting any existing entries. This option is the default. Note that this mode only propagates additions and modifications, not deletions.

`-A` All data. This option specifies that the data within the table and all of the data in tables in the initial table's concatenation path be returned.

`-d` Dump the NIS+ table to the standard output in the appropriate format for the given `type`. For tables of type `key-value`, use `niscat(1)` instead. To dump the `cred` table, dump the `publickey` and the `netid` types.

`-D` *defaults* This option specifies a different set of defaults to be used during this operation. The *defaults* string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

`ttl=time` This token sets the default time to live for objects that are created by this command. The value `time` is specified in the format as defined by the `nischttl(1)` command. The default is 12 hours.

`owner=`*ownername* This token specifies that the NIS+ principal *ownername* should own the created object. The default for this value is the principal who is executing the command.

`group=`*groupname* This token specifies that the group *groupname* should be the group owner for the object that is created. The default is `NULL`.

`access=`*rights* This token specifies the set of access rights that are to be granted for the given object. The value *rights* is specified in the format as defined by the `nischmod(1)` command. The default is

`– – – –rmcdr – – –r – – –`

`-f` *file*	Specify that `file` should be used as the source of input (instead of the standard input).
`-m`	Merge the file or map with the NIS+ table. This is the most efficient way to bring an NIS+ table up to date with a file or NIS map when there are only a small number of changes. This option adds entries that are not already in the database, modifies entries that already exist (if changed), and deletes any entries that are not in the source. Use the `-m` option whenever the database is large and replicated, and the map being loaded differs only in a few entries. This option reduces the number of update messages that have to be sent to the replicas. Also see the `-r` option.
`-M`	Master server only. This option specifies that lookups should be sent to the master server. This guarantees that the most up-to-date information is seen at the possible expense that the master server may be busy, or that it may be made busy by this operation.
`-o`	Use strictly conforming `publickey` files. Dumps will not add the algorithm `type` field used by additional authentication mechanisms that might be configured using `nisauthconf(1M)`. 192-bit keys that are dumped using this option can be read by previous versions of `nisaddent`. However, the algorithm field will be lost and assumed to be "0" when read. Use the `-o` option when reading `publickey` files from previous versions of `nisaddent` to avoid warnings about the missing algorithm field.
`-p`	Process the password field when loading password information from a file. By default, the password field is ignored because it is usually not valid (the actual password appears in a shadow file).
`-P`	Follow concatenation path. This option specifies that lookups should follow the concatenation path of a table if the initial search is unsuccessful.
`-q`	Dump tables in "quick" mode. The default method for dumping tables processes each entry individually. For some tables, for example, hosts, multiple entries must be combined into a single line, so extra requests to the server must be made. In "quick" mode, all of the entries for a table are retrieved in one call to the server, so the table can be dumped more quickly. However, for large tables, there is a chance that the process will run out of virtual memory and the table will not be dumped.
`-r`	Replace the file or map in the existing NIS+ table by first deleting any existing entries, and then add the entries from the source (`/etc` files, or NIS+ maps). This option has the same effect as the `-m` option. The use of this option is *strongly* discouraged due to its adverse impact on performance, unless there are a large number of changes.

`-t` *table* Specify that *table* should be the NIS+ table for this operation. This should be a relative name as compared to your default domain or the `domainname` if it has been specified.

`-v` Verbose.

`-y` *ypdomain* Use the `dbm` files for the appropriate NIS map, from the NIS domain *ypdomain*, as the source of input. The files are expected to be on the local machine in the `/var/yp/`*ypdomain* directory. If the machine is not an NIS server, use `ypxfr(1M)` to get a copy of the `dbm` files for the appropriate map.

`-Y` *map* Use the `dbm` files for *map* as the source of input.

Examples

EXAMPLE 1 Using `nisaddent`

This example adds the contents of `/etc/passwd` to the `passwd.org_dir` table:

```
example% cat /etc/passwd | nisaddent passwd
```

The next example adds the shadow information. Note that the table type here is "shadow", not "passwd", even though the actual information is stored in the `passwd` table:

```
example% cat /etc/shadow | nisaddent shadow
```

This example replaces the `hosts.org_dir` table with the contents of `/etc/hosts` (in verbose mode):

```
example% nisaddent -rv -f /etc/hosts hosts
```

This example merges the `passwd` map from `yypdomain` with the `passwd.org_dir.nisdomain` table (in verbose mode). The example assumes that the `/var/yp/myypdomain` directory contains the `yppasswd` map.

```
example% nisaddent -mv -y myypdomain passwd nisdomain
```

This example merges the `auto.master` map from `myypdomain` with the `auto_master.org_dir` table:

```
example% nisaddent -m -y myypdomain -Y auto.master \
     -t auto_master.org_dir key-value
```

This example dumps the `hosts.org_dir` table:

```
example% nisaddent -d hosts
```

This example dumps the `ipnodes.org_dir` table:

```
example% nisaddent -d ipnodes
```

Environment Variables

`NIS_DEFAULTS` This variable contains a default string that will override the NIS+ standard defaults. If the `-D` switch is used, those values will then override both the `NIS_DEFAULTS` variable and the standard defaults. To avoid security

accidents, the access rights in the `NIS_DEFAULTS` variable are ignored for the `passwd` table (but access rights specified with `-D` are used).

`NIS_PATH` If this variable is set, and neither the *nisdomain* nor the *table* are fully qualified, each directory specified in `NIS_PATH` will be searched until the table is found (see `nisdefaults(1)`).

Exit Status The following exit values are returned:

`0` Successful operation.

`1` Failure caused by an error other than parsing.

`2` A parsing error occurred on an entry. A parsing error does not cause termination; the invalid entries are simply skipped.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `niscat(1)`, `nischmod(1)`, `nischttl(1)`, `nisdefaults(1)`, `nistbladm(1)`, `nisauthconf(1M)`, `nispopulate(1M)`, `nisserver(1M)`, `nissetup(1M)`, `ypxfr(1M)`, `hosts(4)`, `passwd(4)`, `shadow(4)`, `attributes(5)`

Notes NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisauthconf – configure NIS+ security

Synopsis

```
nisauthconf [-v] [mechanism,]...
```

Description `nisauthconf` controls which authentication flavors NIS+ should use when communicating with other NIS+ clients and servers. If the command is not executed, then NIS+ will default to the `AUTH_DES` authentication flavor when running security level 2. See `rpc.nisd(1M)`.

`nisauthconf` takes a list of authentication *mechanism*'s in order of preference. An authentication *mechanism* may use one or more authentication flavors listed below. If `des` is the only specified mechanism, then NIS+ only use `AUTH_DES` with other NIS+ clients and servers. If `des` is the first mechanism, then other authentication *mechanism*'s after `des` will be ignored by NIS+, except for `nisaddcred(1M)`. After changing the mechanism configuration, the `keyserv(1M)` daemon must be restarted. Note that doing so will remove encryption keys stored by the running `keyserv` process. This means that a `reboot` usually is the safest option when the mechanism configuration has been changed.

The following mechanisms are available:

Authentication *mechanism*	Authentication Flavor
des	`AUTH_DES`
dh640–0	`RPCSEC_GSS` using 640-bit Diffie-Hellman keys
dh1024–0	`RPCSEC_GSS` using 1024-bit Diffie-Hellman keys

If no mechanisms are specified, then a list of currently configured mechanisms is printed.

Options `-v` Displays a verbose table listing the currently configured authentication mechanisms.

Examples

EXAMPLE 1 Configuring a System with only `RPCSEC_GSS` Authentication Flavor

To configure a system to use only the `RPCSEC_GSS` authentication flavor with 640-bit Diffie-Hellman keys, execute the following as root:

```
example# /usr/lib/nis/nisauthconf dh640-0
```

EXAMPLE 2 Configuring a System with both `RPCSEC_GSS` and `AUTH_DES` Authentication Flavors

To configure a system to use both `RPCSEC_GSS` (with 640-bit Diffie-Hellman keys) and `AUTH_DES` authentication flavors:

```
example# /usr/lib/nis/nisauthconf dh640-0 des
```

EXAMPLE 3 Transitioning to Other Authentication Flavors

The following example can be used while adding credentials for a new mechanism before NIS+ is authenticating with the new mechanism:

EXAMPLE 3 Transitioning to Other Authentication Flavors *(Continued)*

```
example# /usr/lib/nis/nisauthconf des dh640-0
```

Note that except for nisaddcred(1M), NIS+ will not use mechanisms that follow 'des.'

Exit Status The following exit values are returned:

0 Successful completion.

1 An error occurred.

Files

/etc/rpcsec/nisplussec.conf	NIS+ authentication configuration file. This file may change or be removed in future versions of Solaris.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also NIS+(1), keyserv(1M), nisaddcred(1M), rpc.nisd(1M), attributes(5)

Notes A NIS+ client of a server that is configured for either dh640–0 or dh1024–0 must run Solaris 7 or later, even if the server is also configured with des.

NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisbackup – backup NIS+ directories

Synopsis

```
nisbackup [-v] backup-dir directory...

nisbackup [-v] -a backup-dir
```

Description `nisbackup` backs up a NIS+ directory object on a NIS+ master server. Updates to the NIS+ database will be temporarily disabled while `nisbackup` is running. The *backup-dir* is a UNIX directory that must exist prior to running `nisbackup`. The `nisbackup` command can be used to backup an individual NIS+ directory object or all (`-a`) of the NIS+ directory objects served by a master server. The NIS+ directory objects being backed up will be placed into subdirectories under the *backup-dir* directory. These subdirectories are named according to the NIS+ directory object they contain. `nisbackup` operates on individual NIS+ directory objects (for example, `org_dir.wiz.com`). This allows an administrator to selectively backup specific directories.

The `rpc.nisd(1M)` process must be running on the master server with a stable NIS+ database for `nisbackup` to complete. `nisbackup` will not attempt to correct any corruption in the NIS+ database, so it is important that backups be done regularly as part of the NIS+ administration.

The first synopsis is used to backup a single NIS+ directory object or a list of NIS+ directory objects. The objects can be partially qualified or fully qualified. The machine on which the command is executing must be the master for the NIS+ directory objects specified.

The second synopsis will backup all of the NIS+ directory objects that are served by this master. The `-a` option is the recommended method of backing up a master server, since it will backup all NIS+ directory objects that are served by this master. If this server is a master server for more than one domain, the backup will include NIS+ directories that belong to all of the domains served. Individual NIS+ directory objects can be selected for restoring from a *backup-dir* created with the `-a` option. See `nisrestore(1M)`.

The `-a` option only includes directory objects for which this server is the master. It is possible, but not recommended, to configure a master server as a replica for other domains. The objects belonging to those replicated domains will not be backed up with the `-a` option. The backup of replicated objects must be run on the master server for those objects.

Do not use the same *backup-dir* to backup different master servers. Each master server must have its own *backup-dir*.

`nisbackup` will set the `rpc.nisd(1M)` to read only mode, which will disable updates to the NIS+ database. This is neccessary to ensure the consistency of the backup. For this reason, `nisbackup` should not be run while large numbers of updates are being applied to the NIS+ database. Update utilities such as `nisaddent(1M)` should not be run simultaneously with `nisbackup`.

Options

`-a` Creates a backup of all NIS+ directory objects for which this server is a master.

`-v` Verbose option. Additional output will be produced and sent to `syslog(3C)` upon execution of the command (see `syslog.conf(4)`).

Operands

backup-dir The directory into which the subdirectories containing the backed up objects are placed. This must be created prior to running `nisbackup`.

directory The NIS+ directory object(s) being backed up.

Examples

EXAMPLE 1 Backup of the `org_dir` NIS+ directory object of the domain `foo.com` on a master server to a directory named `/backup`

To backup the `org_dir` NIS+ directory object of the domain `foo.com` on a master server to a directory named `/backup`:

```
master_server# nisbackup /backup org_dir.foo.com.
```

EXAMPLE 2 Backup of the entire NIS+ domain `foo.com` to a directory named `/backup`

To backup the entire NIS+ domain `foo.com` to a directory named `/backup`:

```
master_server# nisbackup /backup foo.com. \
              org_dir.foo.com. groups_dir.foo.com. \
              ctx_dir.foo.com.
```

EXAMPLE 3 Backup of an entire NIS+ database to a backup directory named `/backup`

To backup an entire NIS+ database to a backup directory named `/backup`:

```
master_server# nisbackup -a /backup
```

Exit Status

`0` Successful completion.

`1` An error occurred.

Files

/backup-dir/backup_list	This ascii file contains a list of all the objects contained in this *backup-dir* directory.
/backup-dir/directory-object	A subdirectory that is created in the *backup-dir* that contains the NIS+ directory-object backup.
/backup-dir/directory-object`/data`	A subdirectory that contains the data files that are part of the NIS+ directory-object backup.
/backup-dir/directory-object`/last.upd`	This data file contains timestamp information about the directory-object.
/backup-dir/directory-object`/data.dict`	A NIS+ data dictionary for all of the objects contained in the NIS+ directory-object backup.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also NIS+(1), nisdefaults(1), nisrm(1), nisrestore(1M), rpc.nisd(1M), syslog(3C), nisfiles(4), syslog.conf(4), attributes(5)

Notes NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nis_cachemgr – NIS+ utility to cache location information about NIS+ servers

Synopsis

```
/usr/sbin/nis_cachemgr [-i] [-v]
```

Description The `nis_cachemgr` daemon maintains a cache of NIS+ directory objects and active servers for domains. It is responsible for locating servers for a domain on behalf of client processes. This improves performance because only one process has to search for servers. The cache contains location information necessary to contact the NIS+ servers. This includes transport addresses, information neeeded to authenticate the server, and a time to live field which gives a hint on how long the directory object can be cached. The cache helps to improve the performance of the clients that are traversing the NIS+ name space. `nis_cachemgr` should be running on all the machines that are using NIS+. However, it is not required that the `nis_cachemgr` program be running in order for NIS+ requests to be serviced.

The cache maintained by this program is shared by all the processes that access NIS+ on a machine. The cache is maintained in a file that is memory mapped by all the processes. See `mmap(2)`. On start up, `nis_cachemgr` initializes the cache from the cold start file and preserves unexpired entries that already exist in the cache file. See `nisinit(1M)`. Thus, the cache survives machine reboots.

The `nis_cachemgr` program is normally started from a system startup script. `nisshowcache(1M)` can be used to look at the cached objects and active servers.

The `nisprefadm(1M)` command can be used to control which NIS+ servers the `nis_cachemgr` program will try to select.

The `nis_cachemgr` program makes NIS+ requests under the NIS+ principal name of the host on which it runs. Before running `nis_cachemgr`, security credentials for the host should be added to the `cred.org_dir` table in the host's domain using `nisaddcred(1M)`. Credentials of type DES will be needed if the NIS+ service is operating at security level 2 (see `rpc.nisd(1M)`). See the Diagnostics section, below. Additionally, a "`keylogin -r` " should be done on the machine.

`svc:/network/rpc/keyserv:default` is required for NIS+ operation. See `NOTES`.

Options

`-i` Force `nis_cachemgr` to ignore the previous cache file and reinitialize the cache from just the cold start file. By default, the cache manager initializes itself from both the cold start file and the old cache file, thereby maintaining the entries in the cache across machine reboots.

`-v` This flag sets verbose mode. In this mode, the `nis_cachemgr` program logs not only errors and warnings, but also additional status messages. The additional messages are logged using `syslog(3C)` with a priority of `LOG_INFO`.

Files

`/var/nis/NIS_SHARED_DIRCACHE`	the shared cache file
`/var/nis/NIS_COLD_START`	the coldstart file

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also keylogin(1), svcs(1), nisaddcred(1M), nisinit(1M), nisprefadm(1M), nisshowcache(1M), rpc.nisd(1M), svcadm(1M), mmap(2), rpc(3NSL), syslog(3C), nisfiles(4), attributes(5), smf(5)

Diagnostics The nis_cachemgr daemon logs error messages and warnings using syslog(3C). Error messages are logged to the DAEMON facility with a priority of LOG_ERR . Warning messages are logged with a priority of LOG_WARNING. Additional status messages can be obtained using the -v option.

Notes NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

The nis_cachemgr service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/rpc/nisplus:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name nisclient – initialize NIS+ credentials for NIS+ principals

Synopsis

```
/usr/lib/nis/nisclient -c [-x] [-o] [-v]
     [-l <network_password>] [-d <NIS+_domain>] client_name...

/usr/lib/nis/nisclient -i [-x] [-v] -h <NIS+_server_host>
     [-a <NIS+_server_addr>]
     [-k <key_domain>] [-d <NIS+_domain>] [-S 0 | 2]

/usr/lib/nis/nisclient -u [-x] [-v]

/usr/lib/nis/nisclient -r [-x]
```

Description The nisclient shell script can be used to:

- create NIS+ credentials for hosts and users
- initialize NIS+ hosts and users
- restore the network service environment

NIS+ credentials are used to provide authentication information of NIS+ clients to NIS+ service.

Use the first synopsis (`-c` option) to create individual NIS+ credentials for hosts or users. You must be logged in as a NIS+ principal in the domain for which you are creating the new credentials. You must also have write permission to the local "cred" table. The *client_name* argument accepts any valid host or user name in the NIS+ domain (for example, the *client_name* must exist in the hosts or passwd table). `nisclient` verifies each *client_name* against both the `host` and `passwd` tables, then adds the proper NIS+ credentials for hosts or users. Note that if you are creating NIS+ credentials outside of your local domain, the host or user must exist in the `host` or `passwd` tables in both the local and remote domains.

By default, `nisclient` will not overwrite existing entries in the credential table for the hosts and users specified. To overwrite, use the `-o` option. After the credentials have been created, `nisclient` will print the command that must be executed on the client machine to initialize the host or the user. The `-c` option requires a network password for the client which is used to encrypt the secret key for the client. You can either specify it on the command line with the `-l` option or the script will prompt you for it. You can change this network password later with `passwd(1)` or `chkey(1)`.

`nisclient -c` is not intended to be used to create NIS+ credentials for all users and hosts which are defined in the passwd and hosts tables. To define credentials for all users and hosts, use `nispopulate(1M)`.

Use the second synopsis (`-i` option) to initialize a NIS+ client machine. The `-i` option can be used to convert machines to use NIS+ or to change the machine's domainname. You must be logged in as super-user on the machine that is to become a NIS+ client. Your administrator must have already created the NIS+ credential for this host by using `nisclient -c` or `nispopulate -C`. You will need the network password your administrator created. `nisclient` will prompt you for the network password to decrypt your secret key and then for this

machine's root login password to generate a new set of secret/public keys. If the NIS+ credential was created by your administrator using `nisclient -c`, then you can simply use the initialization command that was printed by the `nisclient` script to initialize this host instead of typing it manually.

To initialize an unauthenticated NIS+ client machine, use the `-i` option with `-S` *0*. With these options, the `nisclient -i` option will not ask for any passwords.

During the client initialization process, files that are being modified are backed up as *files*`.no_nisplus`. The files that are usually modified during a client initialization are: `/etc/defaultdomain`, `/etc/nsswitch.conf`, `/etc/inet/hosts`, and, if it exists, `/var/nis/NIS_COLD_START`. Notice that a file will not be saved if a backup file already exists.

The `-i` option does not set up a NIS+ client to resolve hostnames using DNS. Please refer to the DNS documentation for information on setting up DNS. (See `resolv.conf(4)`).

It is not necessary to initialize either NIS+ root master servers or machines that were installed as NIS+ clients using `suninstall(1M)`.

Use the third synopsis (`-u` option) to initialize a NIS+ user. You must be logged in as the user on a NIS+ client machine in the domain where your NIS+ credentials have been created. Your administrator should have already created the NIS+ credential for your username using `nisclient -c` or `nispopulate(1M)`. You will need the network password your administrator used to create the NIS+ credential for your username. `nisclient` will prompt you for this network password to decrypt your secret key and then for your login password to generate a new set of secret/public keys.

Use the fourth synopsis (`-r` option) to restore the network service environment to whatever you were using before `nisclient -i` was executed. You must be logged in as super-user on the machine that is to be restored. The restore will only work if the machine was initialized with `nisclient -i` because it uses the backup files created by the `-i` option.

Reboot the machine after initializing a machine or restoring the network service.

Options The following options are supported:

`-a` <*NIS+_server_addr*>	Specifies the IP address for the NIS+ server. This option is used *only* with the `-i` option.
`-c`	Adds `DES` credentials for NIS+ principals.
`-d` <*NIS+_domain*>	Specifies the NIS+ domain where the credential should be created when used in conjunction with the `-c` option. It specifies the name for the new NIS+ domain when used in conjunction with the `-i` option. The default is your current domainname.
`-h` <*NIS+_server_host*>	Specifies the NIS+ server's hostname. This option is used *only* with the `-i` option.

`-i`	Initializes a NIS+ client machine.
`-l` <*network_password*>	Specifies the network password for the clients. This option is used *only* with the `-c` option. If this option is not specified, the script will prompt you for the network password.
`-k` <*key_domain*>	This option specifies the domain where root's credentials are stored. If a domain is not specified, then the system default domain is assumed.
`-o`	Overwrites existing credential entries. The default is not to overwrite. This is used *only* with the `-c` option.
`-r`	Restores the network service environment.
`-S` *0*|*2*	Specifies the authentication level for the NIS+ client. Level `0` is for unauthenticated clients and level `2` is for authenticated (`DES`) clients. The default is to set up with level `2` authentication. This is used *only* with the `-i` option. `nisclient` always uses level `2` authentication (`DES`) for both `-c` and `-u` options. There is no need to run `nisclient` with `-u` and `-c` for level `0` authentication. To configure authentication mechanisms other than DES at security level 2, use `nisauthconf(1M)` before running `nisclient`.
`-u`	Initializes a NIS+ user.
`-v`	Runs the script in verbose mode.
`-x`	Turns the "echo" mode on. The script just prints the commands that it would have executed. Notice that the commands are not actually executed. The default is off.

Examples

EXAMPLE 1 Adding the DES Credential in the Local Domain

To add the `DES` credential for host `sunws` and user `fred` in the local domain:

```
example% /usr/lib/nis/nisclient -c sunws fred
```

EXAMPLE 2 Adding the DES Credential in a Specified Domain

To add the `DES` credential for host `sunws` and user `fred` in domain `xyz.example.com.`:

```
example% /usr/lib/nis/nisclient -c -d xyz.example.com. sunws fred
```

EXAMPLE 3 Initializing the Host in a Specific Domain

To initialize host `sunws` as a NIS+ client in domain `xyz.example.com.` where `nisplus_server` is a server for the domain `xyz.example.com.`:

```
example# /usr/lib/nis/nisclient -i -h nisplus_server -d xyz.example.com
```

EXAMPLE 3 Initializing the Host in a Specific Domain *(Continued)*

The script will prompt you for the IP address of `nisplus_server` if the server is not found in the `/etc/hosts` file. The `-d` option is needed only if your current domain name is different from the new domain name.

EXAMPLE 4 Initializing the Host as an Unauthenticated Client in a Specific Domain

To initialize host `sunws` as an unauthenticated NIS+ client in domain `xyz.example.com.` where `nisplus_server` is a server for the domain `xyz.example.com`:

```
example# /usr/lib/nis/nisclient -i -S 0 \
   -h nisplus_server -d xyz.example.com. -a 172.16.44.1
```

EXAMPLE 5 Initializing the User as a NIS+ principal

To initialize user `fred` as a NIS+ principal, log in as user `fred` on a NIS+ client machine.

```
example% /usr/lib/nis/nisclient -u
```

Files

`/var/nis/NIS_COLD_START`	This file contains a list of servers, their transport addresses, and their Secure RPC public keys that serve the machines default domain.
`/etc/defaultdomain`	The system default domainname.
`/etc/nsswitch.conf`	Configuration file for the name-service switch.
`/etc/inet/hosts`	Local host name database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `chkey(1)`, `keylogin(1)`, `NIS+(1)`, `passwd(1)`, `keyserv(1M)`, `nisaddcred(1M)`, `nisauthconf(1M)`, `nisinit(1M)`, `nispopulate(1M)`, `suninstall(1M)`, `nsswitch.conf(4)`, `resolv.conf(4)`, `attributes(5)`

Notes NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisinit – NIS+ client and server initialization utility

Synopsis

```
nisinit -r

nisinit -p Y | D | N parent_domain host...

nisinit -c [-k <key_domain>] -H host | -B | -C coldstart
```

Description `nisinit` initializes a machine to be a NIS+ client or an NIS+ root master server. It may be easier to use `nisclient(1M)` or `nisserver(1M)` to accomplish this same task.

Options

`-r`

Initialize the machine to be a NIS+ root server. This option creates the file `/var/nis/data/root.object` and initialize it to contain information about this machine. It uses the `sysinfo(2)` system call to retrieve the name of the default domain.

To initialize the machine as an NIS+ root server, it is advisable to use the "`-r`" option of `nisserver(1M)`, instead of using "`nisinit -r`".

`-p Y | D | N` *parent_domain host . . .*

This option is used on a root server to initialize a `/var/nis/data/parent.object` to make this domain a part of the namespace above it. Only root servers can have parent objects. A parent object describes the namespace "above" the NIS+ root. If this is an isolated domain, this option should not be used. The argument to this option tells the command what type of name server is serving the domain above the NIS+ domain. When clients attempt to resolve a name that is outside of the NIS+ namespace, this object is returned with the error `NIS_FOREIGNNS` indicating that a name space boundary has been reached. It is up to the client to continue the name resolution process.

The parameter *parent_domain* is the name of the parent domain in a syntax that is native to that type of domain. The list of host names that follow the domain parameter are the names of hosts that serve the parent domain. If there is more than one server for a parent domain, the first host specified should be the master server for that domain.

`Y` Specifies that the parent directory is a NIS version 2 domain.

`D` Specifies that the parent directory is a DNS domain.

`N` Specifies that the parent directory is another NIS+ domain. This option is useful for connecting a pre-existing NIS+ subtree into the global namespace.

Note that in the current implementation, the NIS+ clients do not take advantage of the `-p` feature. Also, since the parent object is currently not replicated on root replica servers, it is recommended that this option not be used.

`-c` Initializes the machine to be a NIS+ client. There are three initialization options available: initialize by coldstart, initialize by hostname, and initialize by broadcast. The most secure mechanism is to initialize from a trusted coldstart file. The second option is to initialize using a hostname that you specify as a trusted host. The third method is to initialize by broadcast and it is the least secure method.

`-C` *coldstart* Causes the file *coldstart* to be used as a prototype coldstart file when initializing a NIS+ client. This coldstart file can be copied from a machine that is already a client of the NIS+ namespace. For maximum security, an administrator can encrypt and encode (with `uuencode(1C)`) the coldstart file and mail it to an administrator bringing up a new machine. The new administrator would then decode (with `uudecode`), decrypt, and then use this file with the `nisinit` command to initialize the machine as an NIS+ client. If the coldstart file is from another client in the same domain, the `nisinit` command may be safely skipped and the file copied into

the `/var/nis` directory as `/var/nis/NIS_COLD_START`.

`-H` *hostname*
: Specifies that the host `hostname` should be contacted as a trusted NIS+ server. The `nisinit` command will iterate over each transport in the `NETPATH` environment variable and attempt to contact `rpcbind(1M)` on that machine. This hostname *must* be reachable from the client without the name service running. For IP networks this means that there must be an entry in `/etc/hosts` for this host when `nisinit` is invoked.

`-B`
: Specifies that the `nisinit` command should use an IP broadcast to locate a NIS+ server on the local subnet. Any machine that is running the NIS+ service may answer. No guarantees are made that the server that answers is a server of the organization's namespace. If this option is used, it is advisable to check with your system administrator that the server and domain served are valid. The binding information can be dumped to the standard output using the `nisshowcache(1M)` command.

Note that `nisinit -c` will just enable navigation of the NIS+ name space from this client. To make NIS+ your name service, modify the file `/etc/nsswitch.conf` to reflect that. See `nsswitch.conf(4)` for more details.

`-k` <*key_domain*>
: This option specifies the domain where root's credentials are stored. If it is not specified, then the system default domain is assumed. This domain name is used to create the

/var/nis/NIS_COLD_START file.

Return Values `nisinit` returns `0` on success and `1` on failure.

Examples

EXAMPLE 1 Initializing the Machine as an NIS+ Client using the Host *freddy* as a Trusted Server

This example initializes the machine as an NIS+ client using the host *freddy* as a trusted server.

```
example# nisinit -cH freddy
```

EXAMPLE 2 Setting up a Client using a Trusted Coldstart File

This example sets up a client using a trusted coldstart file.

```
example# nisinit -cC /tmp/colddata
```

EXAMPLE 3 Setting up a Client Using an IP Broadcast

This example sets up a client using an IP broadcast.

```
example# nisinit -cB
```

EXAMPLE 4 Setting up a Root Server

This example sets up a root server.

```
example# nisinit -r
```

Environment Variables

`NETPATH` This environment variable may be set to the transports to try when contacting the NIS+ server (see `netconfig(4)`). The client library will only attempt to contact the server using connection oriented transports.

Files

`/var/nis/NIS_COLD_START`	This file contains a list of servers, their transport addresses, and their Secure RPC public keys that serve the machine's default domain.
`/var/nis/data/root.object`	This file describes the root object of the NIS+ namespace. It is a standard XDR-encoded NIS+ directory object that can be modified by authorized clients using the `nis_modify()` interface.
`/var/nis/data/parent.object`	This file describes the namespace that is logically above the NIS+ namespace. The most common type of parent object is a DNS object. This object contains contact information for a server of that domain.
`/etc/hosts`	Internet host table.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also NIS+(1), uuencode(1C), nisclient(1M), nisserver(1M), nisshowcache(1M), sysinfo(2), hosts(4), netconfig(4), nisfiles(4), attributes(5)

Notes NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisldapmaptest – test NIS+ and LDAP mapping configuration files

Synopsis

```
nisldapmaptest [-s | -r | -d] [-l | -t object] [-v] [-i]
     [-o] [-m conffile] [-x attr=val...] [ col=val]...
```

Description Use the `nisldapmaptest` utility to test NIS+ to LDAP mapping configuration files. See `NIS+LDAPmapping(4)`. The `nisldapmaptest` utility uses much of the same internal interface as the `rpc.nisd(1M)` does to read, add, modify, or delete LDAP data, as specified by the column name and value operand pairs. `nisldapmaptest` does not read or modify any of the `rpc.nisd(1M)` database files.

See Notes for details on important differences between the ways that `nisldapmaptest` and `rpc.nisd(1M)` operate on LDAP data.

Options The `nisldapmaptest` utility supports the following options:

`-d` Delete data in LDAP.

`-i` Ignore failures when obtaining information from the NIS+ server. This enables `nisldapmaptest` to work to some extent, even if the NIS+ server is unreachable, or if the system is not a NIS+ client. However, NIS+ lookups are still attempted, so there may be NIS+ error messages.

In this mode, `nisldapmaptest` also tries to guess things such as NIS+ object types and derives table column information from the mapping rules in the configuration files. Avoid using the `-i` option to add, modify, or delete, until you have determined that the `nisldapmaptest`'s guesses are adequate for your needs.

`-l` Parse the configuration file into internal data structures, and then print out the configuration per those structures. Note that the printed data is not in configuration file format.

Either `-l` or `-t` must be specified. If both are present, `-l` is ignored.

`-m` *conffile* Specify the name of the `NIS+LDAPmapping(4)` configuration file. The default directory is `/var/nis` , and the default mapping file is `NIS+LDAPmapping`.

`-o` For NIS+ tables, work on the NIS+ object itself, specified by means of the `-t` option, not on the table entries.

`-r` Replace or add data in LDAP.

`-s` Search for data in LDAP. This is the default.

`-t` *object* Specify the NIS+ object on which to operate. If the object name is not fully qualified, that is, it does not end in a dot, the value of the `nisplusLDAPbaseDomain` attribute is appended.

`-v` Set the verbose flag. This flag produces extra diagnostic information.

`-x attr=val...` Specify mapping attribute and value pairs to override those obtained by means of the configuration file. Although any attributes defined on `NIS+LDAPmapping(4)` or `rpc.nisd(4)` can be specified, the ones that control `rpc.nisd(1M)` operation have no effect on `nisldapmaptest`.

Operands The following operands are supported:

`col=val...` NIS+ column and value pairs used to specify which entries should be looked up, added, modified, or deleted. For additions and modifications, use `col=val` to specify the new values.

Examples

EXAMPLE 1 Searching for a User

Use the following example to search for the user `xyzzy` in the LDAP container specified for the `passwd.org_dir` table.

```
example% nisldapmaptest -t passwd.org_dir name=xyzzy
```

EXAMPLE 2 Listing Table Entries

Use the following example to list all entries in the container specified for the `services.org_dir` table.

```
example% nisldapmaptest -t services.org_dir
```

EXAMPLE 3 Listing an Object

Use the following example to list the `services.org_dir` object itself, as it is stored in LDAP.

```
example% nisldapmaptest -o -t services.org_dir
```

EXAMPLE 4 Modifying a Table Entry

Use the following example to modify the membership list of the group `grp`, in the container specified for the `group.org_dir` table, to be `mem1`, `mem2`, and `mem3`.

```
example% nisldapmaptest -r -t group.org_dir name=grp \
     members=mem1,mem2,mem3
```

EXAMPLE 5 Deleting a Table Entry

Use the following example to delete the host called `bad` from the container specified for the `hosts.org_dir` table.

```
example% nisldapmaptest -d -t hosts.org_dir name=bad
```

Exit Status The following exit values are returned:

`0` The requested operation was successful.

`!= 0` An error occurred.

Files `/var/nis/NIS+LDAPmapping.template`

`/etc/default/rpd.nisd`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisr
Interface Stability	Obsolete

See Also `rpc.nisd(1M)`, `NIS+LDAPmapping(4)`, `rpc.nisd(4)`, `attributes(5)`

Notes There are several differences between the ways that `nisldapmaptest` and `rpc.nisd` operate:

1. `nisldapmaptest` obtains information about NIS+ by means of the NIS+ API calls, while `rpc.nisd` looks in its internal database. Thus, if the NIS+ server is not available, `nisldapmaptest` may be unable to determine NIS+ object types or table column information.
2. While `nisldapmaptest` can add, modify, or delete LDAP data, it does not modify any NIS+ data.
3. When operating on table entries, if `nisldapmaptest` is unable to obtain the entry from NIS+, it composes LDAP operations using only the supplied `col=val` operands. Depending on the mapping used, this can result in extra LDAP operations, for example, attempting to obtain a `DN` for add, modify, or delete.
4. The default value for `nisplusLDAPbaseDomain` is the system domain name per `sysinfo(2)` in `nisldapmaptest`, but the internal notion of the domain it serves in `rpc.nisd`. While the two usually are the same, this is not necessarily always the case.
5. When more than one NIS+ entry maps to a single LDAP entry, `nisldapmaptest` may be unable to perform a complete update, unless you make sure that the `col=val` specification picks up all relevant NIS+ entries. For example, if you have the `services.org_dir` NIS+ entries:

```
cname   name    proto   port

x       x       tcp     12345
x       y       tcp     12345
x       z       tcp     12345
```

then specifying `cname=x` will pick up all three entries and create or modify the corresponding LDAP entry to have three `CN` values: `x`, `y`, and `z`. However, specifying `name=x` will match just the first NIS+ entry, and create or modify the LDAP entry to have just one `CN`: `x`.

Name nislog – display the contents of the NIS+ transaction log

Synopsis

```
/usr/sbin/nislog [-h num | -t num] [-v] [directory]...
```

Description `nislog` displays the contents of the NIS+ server transaction log on the standard output. This command can be used to track changes in the namespace. The `/var/nis/trans.log` file contains the transaction log maintained by the NIS+ server. When updates occur, they are logged to this file and then propagated to replicas as log transactions. When the log is checkpointed, updates that have been propagated to the replicas are removed.

The `nislog` command can only be run on an NIS+ server by superuser. It displays the log entries for that server only.

If *directory* is not specified, the entire log is searched. Otherwise, only those logs entries that correspond to the specified directories are displayed.

Options

`-h` *num* Display *num* transactions from the "head" of the log. If the numeric parameter is `0`, only the log header is displayed.

`-t` *num* Display *num* transactions from the "tail" of the log. If the numeric parameter is `0`, only the log header is displayed.

`-v` Verbose mode.

Files

`/var/nis/trans.log` transaction log

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `NIS+(1)`, `rpc.nisd(1M)`, `nisfiles(4)`, `attributes(5)`

Notes NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisping – send ping to NIS+ servers

Synopsis

```
/usr/lib/nis/nisping [-uf] [-H hostname] [-r | directory]
/usr/lib/nis/nisping -C [-a] [-H hostname] [directory]
```

Description In the first Synopsis line, the `nisping` command sends a "ping" to all replicas of an NIS+ directory. Once a replica receives a ping, it will check with the master server for the directory to get updates. Prior to pinging the replicas, this command attempts to determine the last update "seen" by a replica and the last update logged by the master. If these two timestamps are the same, the ping is not sent. The `-f` (force) option will override this feature.

Under normal circumstances, NIS+ replica servers get the new information from the master NIS+ server within a short time. Therefore, there should not be any need to use `nisping`.

In the second Synopsis line, the `nisping -C` command sends a checkpoint request to the servers. If no *directory* is specified, the home domain, as returned by `nisdefaults(1)`, is checkpointed. If all directories, served by a given server, have to be checkpointed, then use the `-a` option.

On receiving a checkpoint request, the servers would commit all the updates for the given *directory* from the table log files to the database files. This command, if sent to the master server, will also send updates to the replicas if they are out of date. This option is needed because the database log files for NIS+ are not automatically checkpointed. `nisping` should be used at frequent intervals (such as once a day) to checkpoint the NIS+ database log files. This command can be added to the `crontab(1)` file. If the database log files are not checkpointed, their sizes will continue to grow.

If the server specified by the `-H` option does not serve the directory, then no ping is sent.

Per-server and per-directory access restrictions may apply; see `nisopaccess(1)`. `nisping` uses `NIS_CPTIME` and `NIS_PING` (resync (`ping`) of replicas), or `NIS_CHECKPOINT` (for checkpoint). Since the `NIS_PING` operation does not return a status, the `nisping` command is typically unable to indicate success or failure for resyncs.

Options

`-a`	Checkpoint all directories on the server.
`-C`	Send a request to checkpoint, rather than a ping, to each server. The servers schedule to commit all the transactions to stable storage.
`-H` *hostname*	Only the host `hostname` is sent the ping, checked for an update time, or checkpointed.
`-f`	Force a ping, even though the timestamps indicate there is no reason to do so. This option is useful for debugging.
`-r`	This option can be used to update or get status about the root object from the root servers, especially when new root replicas are added or deleted from the list.

If used without `-u` option, `-r` will send a ping request to the servers serving the root domain. When the replicas receive a ping, they will update their root object if needed.

The `-r` option can be used with all other options except with the `-C` option; the root object need not be checkpointed.

`-u` Display the time of the last update; no servers are sent a ping.

Return Values

`−1` No servers were contacted, or the server specified by the `-H` switch could not be contacted.

`0` Success.

`1` Some, but not all, servers were successfully contacted.

Examples

EXAMPLE 1 Using `nisping`

This example pings all replicas of the default domain:

```
example% nisping
```

Note that this example will not ping the `org_dir` and `groups_dir` subdirectories within this domain.

This example pings the server *example* which is a replica of the *org_dir.foo.com.* directory:

```
example% nisping -H example org_dir.foo.com.
```

This example checkpoints all servers of the *org_dir.bar.com.* directory.

```
example% nisping -C org_dir.bar.com.
```

Environment Variables

`NIS_PATH` If this variable is set, and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also

`crontab(1)`, `nisdefaults(1)`, `nisopaccess(1)`, `nislog(1M)`, `nisfiles(4)`, `attributes(5)`

Notes

NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nispopulate – populate the NIS+ tables in a NIS+ domain

Synopsis

```
/usr/lib/nis/nispopulate -Y [-x] [-f] [-n] [-u] [-v]
     [-S 0 | 2] [-l <network_passwd>]
     [-d <NIS+_domain>] -h <NIS_server_host>
     [-a <NIS_server_addr>] -y <NIS_domain>
     [table] ...

/usr/lib/nis/nispopulate -F [-x] [-f] [-u] [-v] [-S 0 | 2]
     [-d <NIS+_domain>] [-l <network_passwd>]
     [-p <directory_path>] [table] ...

/usr/lib/nis/nispopulate -C [-x] [-f] [-v]
     [-d <NIS+_domain>] [-l <network_passwd>]
     [hosts | passwd]
```

Description The `nispopulate` shell script can be used to populate NIS+ tables in a specified domain from their corresponding files or NIS maps. `nispopulate` assumes that the tables have been created either through `nisserver(1M)` or `nissetup(1M)`.

The table argument accepts standard names that are used in the administration of Solaris systems and non-standard *key-value* type tables. See `nisaddent(1M)` for more information on *key-value* type tables. If the table argument is not specified, `nispopulate` will automatically populate each of the standard tables. These standard (default) tables are: `auto_master`, `auto_home`, `ethers`, `group`, `hosts`, `ipnodes`, `networks`, `passwd`, `protocols`, `services`, `rpc`, `netmasks`, `bootparams`, `netgroup`, `aliases` and `shadow`. Note that the `shadow` table is only used when populating from files. The non-standard tables that `nispopulate` accepts are those of *key-value* type. These tables must first be created manually with the `nistbladm(1)` command.

Use the first synopsis (`-Y`) to populate NIS+ tables from NIS maps. `nispopulate` uses `ypxfr(1M)` to transfer the NIS maps from the NIS servers to the `/var/yp/`*<NIS_domain>* directory on the local machine. Then, it uses these files as the input source. Note that *<NIS_domain>* is case sensitive. Make sure there is enough disk space for that directory.

Use the second synopsis (`-F`) to populate NIS+ tables from local files. `nispopulate` will use those files that match the table name as input sources in the current working directory or in the specified directory.

Note that when populating the `hosts`, `ipnodes`, and `passwd` tables, `nispopulate` will automatically create the NIS+ credentials for all users and `hosts` (`ipnodes`) that are defined in the `hosts`, `ipnodes`, and `passwd` tables, respectively. A network passwd is required to create these credentials. This network password is used to encrypt the secret key for the new users and hosts. This password can be specified using the `-l` option or it will use the default password, `nisplus`. `nispopulate` will not overwrite any existing credential entries in the credential table. Use `nisclient(1M)` to overwrite the entries in the `cred` table. It creates both LOCAL and DES credentials for users, and only DES credentials for hosts. To disable automatic credential creation, specify the "`-S` *0*" option.

The third synopsis (`-C`) is used to populate NIS+ credential table with level 2 authentication (DES) from the `hosts`, `ipnodes` and `passwd` tables of the specified domain. The valid table arguments for this operation are `hosts`, `ipnodes` and `passwd`. If this argument is not specified then it will use `hosts`, `ipnodes` and `passwd` as the input source. If other authentication mechanisms are configured using `nisauthconf(1M)`, the NIS+ credential table will be loaded with credentials for those mechanisms.

If `nispopulate` was earlier used with "`-S` *0*" option, then no credentials were added for the hosts or the users. If later the site decides to add credentials for all users and hosts, then this (`-C`) option can be used to add credentials.

Options

`-a` <*NIS_server_addr*>	Specifies the IP address for the NIS server. This option is only used with the `-Y` option.
`-C`	Populate the NIS+ credential table from `hosts`, `ipnodes`, and `passwd` tables using DES authentication (security level 2). If other authentication mechanisms are configured using `nisauthconf(1M)`, the NIS+ credential table will be populated with credentials for those mechanisms.
`-d` <*NIS+_domain.*>	Specifies the NIS+ domain. The default is the local domain.
`-F`	Populates NIS+ tables from files.
`-f`	Forces the script to populate the NIS+ tables without prompting for confirmation.
`-h` <*NIS_server_host*>	Specifies the NIS server hostname from where the NIS maps are copied from. This is only used with the `-Y` option. This hostname must be present in the NIS+ `hosts` or `ipnodes` table, or in the `/etc/hosts` file. If the hostname is not defined, the script will prompt you for its IP address, or you can use the `-a` option to specify the address manually.
`-l` <*network_passwd*>	Specifies the network password for populating the NIS+ credential table. This is only used when you are populating the `hosts`, `ipnodes`, and `passwd` tables. The default passwd is "nisplus".
`-n`	Does not overwrite local NIS maps in `/var/yp/`<*NISdomain*> directory if they already exist. The default is to overwrite the existing NIS maps in the local `/var/yp/`<*NISdomain*> directory. This is only used with the `-Y` option.
`-p` <*directory_path*>	Specifies the directory where the files are stored. This is only used with the `-F` option. The default is the current working directory.
`-S` *0*|*2*	Specifies the authentication level for the NIS+ clients. Level 0 is for unauthenticated clients and no credentials will be created for users and hosts in the specified domain. Level 2 is for authenticated

(DES) clients and DES credentials will be created for users and hosts in the specified domain. The default is to set up with level 2 authentication (DES). There is no need to run `nispopulate` with `-C` for level 0 authentication. Also, if other authentication mechanisms are configured with `nisauthconf(1M)`, credentials for those mechanisms will also be populated for the NIS+ clients.

`-u` — Updates the NIS+ tables (ie., adds, deletes, modifies) from either files or NIS maps. This option should be used to bring an NIS+ table up to date when there are only a small number of changes. The default is to add to the NIS+ tables without deleting any existing entries. Also, see the `-n` option for updating NIS+ tables from existing maps in the `/var/yp` directory.

`-v` — Runs the script in verbose mode.

`-x` — Turns the "echo" mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.

`-Y` — Populate the NIS+ tables from NIS maps.

`-y` <*NIS_domain*> — Specifies the NIS domain to copy the NIS maps from. This is only used with the `-Y` option. The default domainname is the same as the local domainname.

Examples

EXAMPLE 1 Using `nispopulate`

To populate all the NIS+ standard tables in the domain *xyz.sun.com.* from NIS maps of the *yp.sun.COM* domain as input source where host *yp_host* is a YP server of *yp.sun.COM*:

```
nis_server# /usr/lib/nis/nispopulate -Y -y yp.sun.COM \
-h yp_host -d xyz.sun.com.
```

To update all of the NIS+ standard tables from the same NIS domain and hosts shown above:

```
nis_server# /usr/lib/nis/nispopulate -Y -u -y yp.sun.COM -h yp_host \
-d xyz.sun.com.
```

To populate the `hosts` table in domain *xyz.sun.com.* from the `hosts` file in the `/var/nis/files` directory and using `somepasswd` as the network password for key encryption:

```
nis_server# /usr/lib/nis/nispopulate -F -p \
 /var/nis/files -l somepasswd hosts
```

To populate the `passwd` table in domain *xyz.sun.com.* from the `passwd` file in the `/var/nis/files` directory without automatically creating the NIS+ credentials:

EXAMPLE 1 Using `nispopulate` *(Continued)*

```
nis_server# /usr/lib/nis/nispopulate -F -p /var/nis/files \
 -d xys.sun.com. -S 0 passwd
```

To populate the credential table in *domain xyz.sun.com.* for all users defined in the passwd table.

```
nis_server# /usr/lib/nis/nispopulate -C -d xys.sun.com. passwd
```

To create and populate a non-standard key-value type NIS+ table, `private`, from the file `/var/nis/files/private`: (nispopulate assumes that the private.org_dirkey-value type table has already been created).

```
nis_server# /usr/bin/nistbladm -D access=og=rmcd,nw=r \
-c private key=S,nogw= value=,nogw= private.org.dir
nis_server# /usr/lib/nis/nispopulate -F -p /var/nis/files private
```

Environment Variables

`nispopulate` normally creates temporary files in the directory `/tmp`. You may specify another directory by setting the environment variable `TMPDIR` to your chosen directory. If `TMPDIR` is not a valid directory, then `nispopulate` will use `/tmp`).

Files

`/etc/inet/hosts`	local database associating names of nodes with IP addresses
`/var/yp`	NIS (YP) domain directory
`/var/nis`	NIS+ domain directory
`/tmp`	

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also

`NIS+(1)`, `nistbladm(1)`, `nisaddcred(1M)`, `nisaddent(1M)`, `nisauthconf(1M)`, `nisclient(1M)`, `nisserver(1M)`, `nissetup(1M)`, `rpc.nisd(1M)`, `ypxfr(1M)`, `attributes(5)`

Notes

NIS+ might not be supported in future releases of the Solaris operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisprefadm – NIS+ utility to set server preferences for NIS+ clients

Synopsis

```
/usr/bin/nisprefadm -a {-L | -G} [-o opt-string]
     [-d domain] [-C client] server...

/usr/bin/nisprefadm -m {-L | -G} [-o opt-string]
     [-d domain] [-C client] oldserver=newserver...

/usr/bin/nisprefadm -r {-L | -G} [-o opt-string]
     [-d domain] [-C client] server...

/usr/bin/nisprefadm -u {-L | -G} [-o opt-string]
     [-d domain] [-C client] server...

/usr/bin/nisprefadm -x {-L | -G} [-d domain] [-C client]

/usr/bin/nisprefadm -l {-L | -G} [-d domain] [-C client]

/usr/bin/nisprefadm -F
```

Description `nisprefadm` defines which servers are to be preferred by NIS+ clients. This information is used by `nis_cachemgr(1M)` to control the order in which it selects which server to use for a particular domain. On a client system, the cache manager first looks for a local preferred server list in `/var/nis`. If it doesn't find one, it looks for an entry with its host name in the NIS+ table. Finally, if it doesn't find it there, it looks for an entry for its subnet.

By default, `nis_cachemgr` puts all servers that are on the same subnet as the client system (that is, local servers) are on the preferred server list. In some cases this default preferred server list is inadequate. For example, if all of the servers for a domain are remote, but some are *closer* than others, the cache manager should try to select the closer one. Because the cache manager has no reliable way to determine the distance to remote servers, `nisprefadm` is used to provide this information.

The preferred server information is stored either globally in a NIS+ table (with the `-G` option) or locally in a file, `/var/nis/client_info` (with the `-L` option). It is preferable to store the information globally so that it can be used by all clients on a subnet. The `nis_cachemgr` process on a client machine reloads the preferred server information periodically, depending on the machine's setup. If the local file is used, the information is reloaded every 12 hours. If the global table is used, the information is reloaded based on the TTL value of the client information table. This TTL value can be changed using `nischttl(1)`. If you want your changes to take effect immediately, use the `nisprefadm -F` command. When changing local information (`-L`), `nisprefadm` automatically forces `nis_cachemgr` to reload the information.

The cache manager assigns weights to all of the servers on the preferred list. By default, local servers (that is, servers on the same subnet) are given a weight of 0. Other servers are given the weight, “infinite”. This can be changed by using the `nisprefadm` command and giving a weight in parentheses after the server name. When selecting a server for a domain, the cache manager first tries to contact the servers with the lowest weight. If it doesn't get a response, it tries the servers with the next lowest weight, and so on. If it fails to get a response from any of the preferred servers, it tries to contact the non-preferred servers.

The use of weights gives fine control over the server selection process, but care must be given to avoid assigning too many different weights. For example, if weights 0, 1, 2, and 3 are used, but all of the servers with weight 0, 1, and 2, are unavailable, then there will be a noticeable delay in selecting a server. This is because the cache manager waits 5 seconds for a response at each weight level before moving on to the next one. As a general rule, one or two weight levels provides a good balance of server selection control and performance.

When specifying a server name, it is not necessary to fully qualify the name. When the cache manager tries to access a domain, it compares the list of servers for the domain with the list of preferred servers. It will find a match if a preferred server name is a prefix of the name of a server for the domain. If a domain is served by two servers with the same prefix, the preferred server name must include enough of the domain name to distinguish the two.

The `nis_cachemgr(1M)` process automatically adds local servers (same subnet as the client) to the preferred server list with a weight of 0. Thus, it is not necessary to specify them, though it does no harm.

If you specify a weight for a server, you probably should quote the parentheses to avoid having the shell interpret them. The following command illustrates this:

```
example% nisprefadm -G -a -C client1 "srv1(2)"
```

In general, `nis_cachemgr` does a fairly good job of selecting servers on its own. Therefore, the use of `nisprefadm` is not usually necessary. Some situations in which it is recommended are:

No local servers, many remote servers	In this case, `nis_cachemgr` needs to choose one of the remote servers. Because it doesn't have information on which is closest, it sends a ping to all of them and then selects the one that responds fastest. This may not always select the best server. If some of the servers are closer to the client than the others, they should be listed as preferred servers so that `nis_cachemgr` will try them first. This reduces the amount of network traffic for selecting a server.
Very remote servers	In some networks there are NIS+ servers that are only reachable through very slow network connections. It is usually best to avoid unnecessary traffic over that connection. If the *pref_type*=`pref_only` option is set along with preferred servers, then only the preferred servers are contacted for domains they serve. The non-preferred servers are not tried at all; even if all of the preferred servers are unavailable. For domains that are not served by any of the preferred servers, the `pref_only` option is ignored.

Options In the SYNOPSIS, when several options are surrounded by braces (that is, by '{' and '}') one of the options must be specified.

`-a` Add the specified servers to the preferred server list.

`-C` *client* Store the preferred server information with the key, *client*. The *client* can be either a hostname or a subnet number. When a hostname is specified, the preferred server information applies to that host only. When a subnet is specified, the preferred server information applies to all clients on that subnet. The cache manager searches for host specific entries first. It only searches for subnet entries if no host entry is found. If this option is not specified, then the hostname of the machine on which the command is run is used.

`-d` *domain* Specify the *domain* to which the command is to apply.

`-F` Tells `nis_cachemgr(1M)` to refresh its preferred server information. The program periodically does this anyway, but this option forces it to do the refresh immediately. When updating the local information, `nis_cachemgr` automatically refreshes the preferred server information.

This option must be executed as `root`.

`-l` List the current preferred server information.

`-L | -G` Store the preferred server information locally in the file, `/var/nis/client_info` (the `-L` option), or globally in a NIS+ table `client.info.org-dir.`*domain* (the `-G` option). If the information is stored locally, then it only applies to the system on which the command is run. If it is stored globally then it can apply to all systems on a subnet (depending on the value of the `-C` option).

The `-L` option must be run as `root`.

`-m` Modify the preferred server list. The server specified by *oldserver* is replaced by *newserver*. This is typically used to change the weight for a server.

`-o` Specify additional options to control server selection. Currently the only valid option is *pref_type*, which can have a value of either `all` (the default) or `pref_only`. If the value is `all`, then the cache manager tries to contact non-preferred servers if all of the preferred servers fail to respond. If `pref_only` is specified, then it won't try non-preferred servers. The only exception to this is when a domain is not served by any of the preferred servers. In this case, the cache manager ignores the option. This is to avoid requiring that preferred servers be defined for every domain.

`-r` Remove the specified servers from the preferred server list.

`-u` Clear the list of preferred servers and then add the specified servers to the preferred server list.

`-x` Remove the preferred server information completely.

Return Values `nisprefadm` returns the following values:

`0` On success.

`1` On failure.

Examples **EXAMPLE 1** Using `nisprefadm`

This command sets the preferred server list for the system on which it is run:

```
example% nisprefadm -L -a srv1 srv2
```

The information is stored in a file, `/var/nis/client_info`, so it will only affect this one system.

The following command has the same effect, but the information is stored in a NIS+ table in the default domain.

```
example% nisprefadm -G -a srv1 srv2
```

As a system administrator, you might want to set the preferred server information for a client system other than the one you are running the command on. The following command sets the preferred server information for a client system named *client1*:

```
example% nisprefadm -G -a -C client1 srv1 srv2
```

It is common for all client systems on a subnet to use the same set of preferred servers. The following command sets a preferred server list that applies to all clients on subnet, 192.85.18.0:

```
example% nisprefadm -G -a -C 192.85.18.0 srv1 srv2
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `nischttl(1)`, `nis_cachemgr(1M)`, `attributes(5)`

Notes NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisrestore – restore NIS+ directory backup

Synopsis

```
nisrestore [-fv] backup-dir directory...

nisrestore [-fv] -a backup-dir

nisrestore -t backup-dir
```

Description `nisrestore` restores an existing backup of a NIS+ directory object that was created using `nisbackup(1M)`. The *backup-dir* is the UNIX directory that contains the NIS+ backup on the server being restored. The `nisrestore` command can be used to restore a NIS+ directory object or a complete NIS+ database. It also can be used as an "out of band" fast replication for a new replica server being initialized. The `rpc.nisd(1M)` daemon must be stopped before running `nisrestore`.

The first synopsis is used to restore a single directory object or a specified list of directory objects. The directory can be partially qualified or fully qualified. The server being restored will be verified against the list of servers serving the directory. If this server is not configured to serve this object, `nisrestore` will exit with an error. The `-f` option will override this check and force the operation.

The second synopsis will restore all of the directory objects contained in the *backup-dir*. Again, the server will be validated against the serving list for each of the directory objects in the *backup-dir*. If one of the objects in the *backup-dir* are not served by this server, `nisrestore` will exit with an error. The `-f` option will override this check and force the operation.

The `-a` option will attempt to restore all NIS+ objects contained in the *backup-dir*. If any of these objects are not served by the server, `nisrestore` will exit with an error. If the *backup-dir* contains objects that are not served by the server, `nisrestore` must be executed without the `-a` option and the specific directory objects listed.

The `-f` option will disable verification of the server being configured to serve the objects being restored. This option should be used with care, as data could be inadvertently restored to a server that doesn't serve the restored data. This option is required in the case of restoring a single server domain (master server only) or if the other NIS+ servers are unavailable for NIS+ lookups.

The combination of options `-f` and `-a` should be used with caution, as no validation of the server serving the restored objects will be done.

New replicas can be quickly added to a namespace with the `nisrestore` command. The steps are as follows.

Configure the new replica on the master server (see `nisserver(1M)`):

```
master# nisserver -R -h replica
```

Temporarily stop the `rpc.nisd` server process on the new replica server:

```
replica# svcadm disable -t network/rpc/nisplus:default
```

Create a backup of the NIS+ database on the master, which will include the new replica information. See nisbackup(1M). The /backup will need to be exported to the new replica. See share_nfs(1M).

```
master# nisbackup -a /backup
```

Restore the backup of the NIS+ database on the new replica. Use the -f option if nisrestore is unable to lookup the NIS+ objects being restored. The backup should be available through nfs or similar means. See share_nfs(1M).

```
replica# nisrestore -f -a //nfs-mnt/backup
```

Restart the rpc.nisd(1M) process on the new replica, and the server will immediately be available for service:

```
replica# svcadm enable network/rpc/nisplus:default
```

Options The following options are supported:

-a Restores all directory objects included in the *backup-dir* partition.

-f Forces the restoration of a directory without the validation of the server in the directory object's serving list.

-t Lists all directory objects contained in *backup-dir*.

-v Verbose option. Additional output will be produced upon execution of the command.

Operands The following options are supported:

backup-dir The UNIX directory that contains the data files for the NIS+ directory objects to be restored.

directory The NIS+ directory object(s) to be restored. This can be a fully or partially qualified name.

Examples **EXAMPLE 1** Restoring the Directory Object on a Replica Server from a Local UFS Partition

To restore the org_dir directory object of the domain foo.com on a replica server from a local ufs partition named /var/backup:

```
replica_server# nisrestore /var/backup org_dir.foo.com.
```

EXAMPLE 2 Forcing the Restore of a Backed up NIS+ Namespace to a Replica Server From the Backup Partition

To force the restore of an entire backed up NIS+ namespace to a replica server from the backup partition named /var/backup:

```
replica_server# nisrestore -f -a /var/backup
```

EXAMPLE 3 Restoring the Subdomain on a Master Server From a Backup that Includes Other Directory Objects

To restore the subdomain `sub.foo.com` on a master server, from a backup that includes other directory objects:

```
master_server# nisrestore /var/backup sub.foo.com. \
 org_dir.sub.foo.com. groups_dir.sub.foo.com.
```

Exit Status

`0` Successful completion.

`1` An error occurred.

Files

/backup-dir`/backup_list`	This ASCII file contains a list of all the objects contained in this *backup-dir* directory. This information can be displayed with the `-t` option.
/backup-dir/directory-object	A subdirectory that is created in the *backup-dir* which contains the directory-object backup.
/backup-dir/directory-object`/data`	A subdirectory that contains the data files that are part of the directory-object backup.
/backup-dir/directory-object`/last.upd`	This data file contains timestamp information about the directory-object.
/backup-dir/directory-object`/data.dict`	A NIS+ data dictionary for all of the objects contained in this directory-object backup.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also

`svcs(1)`, `NIS+(1)`, `nisdefaults(1)`, `nisbackup(1M)`, `nisserver(1M)`, `rpc.nisd(1M)`, `share_nfs(1M)`, `svcadm(1M)`, `nisfiles(4)`, `attributes(5)`, `smf(5)`

Notes

NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

The NIS+ service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/nisplus:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name nisserver – set up NIS+ servers

Synopsis

```
/usr/lib/nis/nisserver -r [-x] [-f] [-v] [-Y]
     [-d NIS+_domain] [-g NIS+_groupname]
     [-l network_passwd]

/usr/lib/nis/nisserver -M [-x] [-f] [-v] [-Y] -d NIS+_domain
     [-g NIS+_groupname]
     [-h NIS+_server_host]

/usr/lib/nis/nisserver -R [-x] [-f] [-v] [-Y]
     [-d NIS+_domain] [-h NIS+_server_host]
```

Description The `nisserver` shell script can be used to set up a root master, non-root master, and replica NIS+ server with level 2 security (DES). If other authentication mechanisms are configured with `nisauthconf(1M)`, `nisserver` will set up a NIS+ server using those mechanisms. `nisauthconf(1M)` should be used before `nisserver`.

When setting up a new domain, this script creates the NIS+ directories (including `groups_dir` and `org_dir`) and system table objects for the domain specified. It does not populate the tables. `nispopulate(1M)` must be used to populate the tables.

Options

`-d` *NIS+_domain*	Specifies the name for the NIS+ domain. The default is your local domain.
`-f`	Forces the NIS+ server setup without prompting for confirmation.
`-g` *NIS+_groupname*	Specifies the NIS+ group name for the new domain. This option is not valid with `-R` option. The default group is admin.<domain>.
`-h` *NIS+_server_host*	Specifies the hostname for the NIS+ server. It must be a valid host in the local domain. Use a fully qualified hostname (for example, *hostx.xyz.sun.com.*) to specify a host outside of your local domain. This option is *only* used for setting up non-root master or replica servers. The default for non-root master server setup is to use the same list of servers as the parent domain. The default for replica server setup is the local hostname.
`-l` *network_password*	Specifies the network password with which to create the credentials for the root master server. This option is *only* used for master root server setup (`-r` option). If this option is not specified, the script prompts you for the login password.
`-M`	Sets up the specified host as a master server. Make sure that `rpc.nisd(1M)` is running on the new master server before this command is executed.
`-R`	Sets up the specified host as a replica server. Make sure that `rpc.nisd` is running on the new replica server.

`-r`	Sets up the server as a root master server. Use the `-R` option to set up a root replica server.
`-v`	Runs the script in verbose mode.
`-x`	Turns the echo mode on. The script just prints the commands that it would have executed. Note that the commands are not actually executed. The default is off.
`-Y`	Sets up a NIS+ server with NIS-compatibility mode. The default is to set up the server without NIS-compatibility mode.

Usage Use the first synopsis of the command (`-r`) to set up a root master server. To run the command, you must be logged in as super-user on the server machine.

Use the second synopsis of the command (`-M`) to set up a non-root master server for the specified domain. To run the command, you must be logged in as a NIS+ principal on a NIS+ machine and have write permission to the parent directory of the domain that you are setting up. The new non-root master server machine must already be an NIS+ client (see `nisclient(1M)`) and have the `rpc.nisd(1M)` daemon running.

Use the third synopsis of the command (`-R`) to set up a replica server for both root and non-root domains. To run the command, you must be logged in as a NIS+ principal on a NIS+ machine and have write permission to the parent directory of the domain that you are replicating. The new non-root replica server machine must already be an NIS+ client and have the `rpc.nisd` daemon running.

Examples **EXAMPLE 1** Setting up Servers

To set up a root master server for domain `sun.com.`:

```
root_server# /usr/lib/nis/nisserver -r -d sun.com.
```

For the following examples make sure that the new servers are NIS+ clients and that `rpc.nisd` is running on these hosts before executing `nisserver`. To set up a replica server for the `sun.com.` domain on host *sunreplica*:

```
root_server# /usr/lib/nis/nisserver -R -d sun.com. -h sunrep
```

To set up a non-root master server for domain *xyz.sun.com.* on host *sunxyz* with the NIS+ groupname as *admin-mgr.xyz.sun.com.*:

```
root_server# /usr/lib/nis/nisserver -M -d xyz.sun.com. -h sunxyz \
-g admin-mgr.xyz.sun.com.
```

To set up a non-root replica server for domain *xyz.sun.com.* on host *sunabc*:

```
sunxyz# /usr/lib/nis/nisserver -R -d xyz.sun.com. -h sunabc
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also NIS+(1), nisgrpadm(1), nismkdir(1), nisaddcred(1M), nisauthconf(1M), nisclient(1M), nisinit(1M), nispopulate(1M), nisprefadm(1M), nissetup(1M), rpc.nisd(1M), attributes(5)

Notes NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nissetup – initialize a NIS+ domain

Synopsis

```
/usr/lib/nis/nissetup [-Y] [domain]
```

Description `nissetup` is a shell script that sets up a NIS+ domain to service clients that wish to store system administration information in a domain named *domain*. This domain should already exist prior to executing this command. See `nismkdir(1)` and `nisinit(1M)`.

A NIS+ domain consists of a NIS+ directory and its subdirectories: `org_dir` and `groups_dir`. `org_dir` stores system administration information and `groups_dir` stores information for group access control.

`nissetup` creates the subdirectories `org_dir` and `groups_dir` in *domain*. Both subdirectories will be replicated on the same servers as the parent domain. After the subdirectories are created, `nissetup` creates the default tables that NIS+ serves. These are `auto_master`, `auto_home`, `bootparams`, `cred`, `ethers`, `group`, `hosts`, `mail_aliases`, `netmasks`, `networks`, `passwd`, `protocols`, `rpc`, `services`, and `timezone`. The `nissetup` script uses the `nistbladm(1)` command to create these tables. The script can be easily customized to add site specific tables that are created at setup time.

This command is normally executed just once per domain.

While this command creates the default tables, it does not initialize them with data. This is accomplished with the `nisaddent(1M)` command.

It is easier to use the `nisserver(1M)` script to create subdirectories and the default tables.

Options

`-Y` Specify that the domain will be served as both a NIS+ domain as well as an NIS domain using the backward compatibility flag. This will set up the domain to be less secure by making all the system tables readable by unauthenticated clients as well.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `NIS+(1)`, `nismkdir(1)`, `nistbladm(1)`, `nisaddent(1M)`, `nisinit(1M)` `nisserver(1M)`, `attributes(5)`

Notes NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisshowcache – NIS+ utility to print out the contents of the shared cache file

Synopsis

```
/usr/lib/nis/nisshowcache [-v]
```

Description `nisshowcache` prints out the contents of the per-machine NIS+ directory cache that is shared by all processes accessing NIS+ on the machine. By default, `nisshowcache` only prints out the directory names in the cache along with the list of active servers. The shared cache is maintained by `nis_cachemgr(1M)`.

Options

`-v` Verbose mode. Print out the contents of each directory object, including the information on the server name and its universal addresses.

Files `/var/nis/NIS_SHARED_DIRCACHE`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `nis_cachemgr(1M)`, `syslogd(1M)`, `nisfiles(4)`, `attributes(5)`

Diagnostics Error messages are sent to the `syslogd(1M)` daemon.

Notes NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisstat – report NIS+ server statistics

Synopsis `/usr/lib/nis/nisstat [-H` *host*`] [`*directory*`]`

Description The `nisstat` command queries a NIS+ server for various statistics about its operations. These statistics may vary between implementations and from release to release. Not all statistics are available from all servers. If you request a statistic from a server that does not support that statistic, it is never a fatal error. The message "unknown statistic" is returned.

By default, statistics are fetched from the server(s) of the NIS+ directory for the default domain. If *directory* is specified, servers for that directory are queried.

Supported statistics for this release are as follows:

root server	This reports whether the server is a root server.
NIS compat mode	This reports whether the server is running in NIS compat mode.
DNS forwarding in NIS mode	This reports whether the server in NIS compat mode will forward host lookup calls to DNS.
security level	This reports the security level of this server.
serves directories	This lists the directories served by this server.
Operations	This statistic returns results in the form: `OP=`*opname*`:C=`*calls*`:E=`*errors*`:T=`*micros* Where *opname* is replaced by the RPC procedure name or operation, *calls* is the number of calls to this procedure that have been made since the server started running. *errors* is the number of errors that have occurred while processing a call, and *micros* is the average time in microseconds to complete the last 16 calls.
Directory Cache	This statistic reports the number of calls to the internal directory object cache, the number of hits on that cache, the number of misses, and the hit rate percentage.
Group Cache	This statistic reports the number of calls to the internal NIS+ group object cache, the number of hits on that cache, the number of misses, and the hit rate percentage.
Static Storage	This statistic reports the number of bytes the server has allocated for its static storage buffers.
Dynamic Storage	This statistic reports the amount of heap the server process is currently using.
Uptime	This statistic reports the time since the service has been running.

Per-server and per-directory access restrictions may apply. See `nisopaccess(1)`. `nisstat` uses `NIS_STATUS`.

Options

`-H` *host* Normally all servers for the directory are queried. With this option, only the machine named *host* is queried. If the named machine does not serve the directory, no statistics are returned.

Environment Variables

`NIS_PATH` If this variable is set, and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found. See `nisdefaults(1)`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also

`nisdefaults(1)`, `nisopaccess(1)`, `attributes(5)`

Notes

NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nisupdkeys – update the public keys in a NIS+ directory object

Synopsis

```
/usr/lib/nis/nisupdkeys [-a | -C] [-H host] [directory]

/usr/lib/nis/nisupdkeys -s [-a | -C] -H host
```

Description This command updates the public keys in an NIS+ directory object. When the public key(s) for a NIS+ server are changed, `nisupdkeys` reads a directory object and attempts to get the public key data for each server of that directory. These keys are placed in the directory object and the object is then modified to reflect the new keys. If *directory* is present, the directory object for that directory is updated. Otherwise the directory object for the default domain is updated. The new key must be propagated to all directory objects that reference that server.

On the other hand, `nisupdkeys -s` gets a list of all the directories served by *host* and updates those directory objects. This assumes that the caller has adequate permission to change all the associated directory objects. The list of directories being served by a given server can also be obtained by `nisstat(1M)`. Before you do this operation, make sure that the new address/public key has been propagated to all replicas. If multiple authentication mechanisms are configured using `nisauthconf(1M)`, then the keys for those mechanisms will also be updated or cleared.

The user executing this command must have modify access to the directory object for it to succeed. The existing directory object can be displayed with the `niscat(1)` command using the `-o` option.

This command does not update the directory objects stored in the `NIS_COLD_START` file on the NIS+ clients.

If a server is also the root master server, then `nisupdkeys -s` cannot be used to update the root directory.

Options

`-a` Update the universal addresses of the NIS+ servers in the directory object. Currently, this only works for the TCP/IP family of transports. This option should be used when the IP address of the server is changed. The server's new address is resolved using `getipnodebyname(3SOCKET)` on this machine. The `/etc/nsswitch.conf` file must point to the correct source for *ipnodes* and *hosts* for this resolution to work.

`-C` Specify to clear rather than set the public key(s). Communication with a server that has no public key(s) does not require the use of secure RPC.

`-H` *host* Limit key changes only to the server named *host*. If the hostname is not a fully qualified NIS+ name, then it is assumed to be a host in the default domain. If the named host does not serve the directory, no action is taken.

`-s` Update all the NIS+ directory objects served by the specified server. This assumes that the caller has adequate access rights to change all the associated directory objects. If the NIS+ principal making this call does not have adequate permissions to update the directory objects, those particular updates will fail and the caller will

be notified. If the `rpc.nisd` on *host* cannot return the list of servers it serves, the command will print an error message. The caller would then have to invoke `nisupdkeys` multiple times (as in the first synopsis), once per NIS+ directory that it serves.

Examples

EXAMPLE 1 Using `nisupdkeys`

The following example updates the keys for servers of the *foo.bar.* domain.

```
example% nisupdkeys foo.bar.
```

This example updates the key(s) for host *fred* that serves the *foo.bar.* domain.

```
example% nisupdkeys -H fred foo.bar.
```

This example clears the public key(s) for host *wilma* in the *foo.bar.* directory.

```
example% nisupdkeys -CH wilma foo.bar.
```

This example updates the public key(s) in all directory objects that are served by the host *wilma*.

```
example% nisupdkeys -s -H wilma
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also

`chkey(1)`, `niscat(1)`, `nisaddcred(1M)`, `nisauthconf(1M)`, `nisstat(1M)`, `getipnodebyname(3SOCKET)`, `nis_objects(3NSL)`, `attributes(5)`

Notes

NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name nlsadmin – network listener service administration

Synopsis

```
/usr/sbin/nlsadmin -x

/usr/sbin/nlsadmin [options] net_spec

/usr/sbin/nlsadmin [options] -N port_monitor_tag

/usr/sbin/nlsadmin -V

/usr/sbin/nlsadmin -c cmd | -o streamname [-p modules]
     [-A address | -D] [-R prognum : versnum]
```

Description `nlsadmin` is the administrative command for the network listener process(es) on a machine. Each network has at least one instance of the network listener process associated with it; each instance (and thus, each network) is configured separately. The listener process "listens" to the network for service requests, accepts requests when they arrive, and invokes servers in response to those service requests. The network listener process may be used with any network (more precisely, with any connection-oriented transport provider) that conforms to the transport provider specification.

`nlsadmin` can establish a listener process for a given network, configure the specific attributes of that listener, and start and kill the listener process for that network. `nlsadmin` can also report on the listener processes on a machine, either individually (per network) or collectively.

net_spec represents a particular listener process. Specifically, *net_spec* is the relative path name of the entry under `/dev` for a given network (that is, a transport provider). *address* is a transport address on which to listen and is interpreted using a syntax that allows for a variety of address formats. By default, *address* is interpreted as the symbolic ASCII representation of the transport address. An *address* preceded by `\x` will let you enter an address in hexadecimal notation. Note that *address* must appear as a single word to the shell, thus it must be quoted if it contains any blanks.

Changes to the list of services provided by the listener or the addresses of those services are put into effect immediately.

Options `nlsadmin` may be used with the following combinations of options and arguments:

`-x`	Report the status of all of the listener processes installed on this machine.
net_spec	Print the status of the listener process for *net_spec* .
`-q` *net_spec*	Query the status of the listener process for the specified network, and reflects the result of that query in its exit code. If a listener process is active, `nlsadmin` will exit with a status of `0`; if no process is active, the exit code will be `1`; the exit code will be greater than `1` in case of error.

`-v` *net_spec*	Print a verbose report on the servers associated with *net_spec,* giving the service code, status, command, and comment for each. It also specifies the `uid` the server will run as and the list of modules to be pushed, if any, before the server is started.
`-z` *service_code net_spec*	Print a report on the server associated with *net_spec* that has service code *service_code,* giving the same information as in the `-v` option.
`-q -z` *service_code net_spec*	Query the status of the service with service code *service_code* on network *net_spec,* and exits with a status of `0` if that service is enabled, `1` if that service is disabled, and greater than `1` in case of error.
`-l` *address net_spec*	Change or set the transport address on which the listener listens (the general listener service). This address can be used by remote processes to access the servers available through this listener (see the `-a` option, below). If *address* is just a dash (" – "), `nlsadmin` reports the address currently configured, instead of changing it. A change of address takes effect immediately.
`-t` *address net_spec*	Change or set the address on which the listener listens for requests for terminal service but is otherwise similar to the `-l` option above. A terminal service address should not be defined unless the appropriate remote login software is available; if such software is available, it must be configured as service code 1 (see the `-a` option, below).
`-i` *net_spec*	Initialize an instance of the listener for the network specified by *net_spec;* that is, create and initialize the files required by the listener as well as starting that instance of the listener. Note that a particular instance of the listener should be initialized only once. The listener must be initialized before assigning addresses or services.
`-a` *service_code*	[`-p` *modules*] [`-w` *name*] `-c` *cmd* `-y` *comment net_spec* Add a new service to the list of services available through the indicated listener. *service_code* is the code for the service, *cmd* is the command to be invoked in response to that service code, comprised of the full path name of the server and its arguments, and *comment* is a brief (free-form) description of the service for use in various reports. Note that *cmd* must appear as a single word to the shell; if arguments are required,

the *cmd* and its arguments must be enclosed in quotation marks. The *comment* must also appear as a single word to the shell. When a service is added, it is initially enabled (see the `-e` and `-d` options, below).

Service codes are alphanumeric strings, and are administered by AT&T. The numeric service codes 0 through 100 are reserved for internal use by the listener. Service code 0 is assigned to the nlps server, which is the service invoked on the general listening address. In particular, code 1 is assigned to the remote login service, which is the service automatically invoked for connections to the terminal login address.

If the `-p` option is specified, then *modules* will be interpreted as a list of STREAMS modules for the listener to push before starting the service being added. The modules are pushed in the order they are specified. *modules* should be a comma-separated list of modules, with no white space included.

If the `-w` option is specified, then *name* is interpreted as the user name from `/etc/passwd` that the listener should look up. From the user name, the listener obtains the user ID, the group ID(s), and the home directory for use by the server. If `-w` is not specified, the default is to use the user name `listen`.

A service must explicitly be added to the listener for each network on which that service is to be available. This operation will normally be performed only when the service is installed on a machine, or when populating the list of services for a new network.

`-r` *service_code net_spec*

Remove the entry for the *service_code* from that listener's list of services. This is normally done only in conjunction with the de-installation of a service from a machine.

`-e` *service_code net_spec*
`-d` *service_code net_spec*

Enable or disable (respectively) the service indicated by *service_code* for the specified network. The service must previously have been added to the listener for that network (see the `-a` option, above). Disabling a service will cause subsequent service requests for that service to be denied, but the processes from any prior service requests that are still running will continue unaffected.

`-s` *net_spec*
`-k` *net_spec*
: Start and kill (respectively) the listener process for the indicated network. These operations are normally performed as part of the system startup and shutdown procedures. Before a listener can be started for a particular network, it must first have been initialized (see the `-i` option, above). When a listener is killed, processes that are still running as a result of prior service requests will continue unaffected.

Under the Service Access Facility, it is possible to have multiple instances of the listener on a single *net_spec*. In any of the above commands, the option `-N` *port_monitor_tag* may be used in place of the *net_spec* argument. This argument specifies the tag by which an instance of the listener is identified by the Service Access Facility. If the `-N` option is not specified (that is, the *net_spec* is specified in the invocation), then it will be assumed that the last component of the *net_spec* represents the tag of the listener for which the operation is destined. In other words, it is assumed that there is at least one listener on a designated *net_spec*, and that its tag is identical to the last component of the *net_spec*. This listener may be thought of as the primary, or default, listener for a particular *net_spec*.

`nlsadmin` is also used in conjunction with the Service Access Facility commands. In that capacity, the following combinations of options can be used:

`-V`
: Write the current version number of the listener's administrative file to the standard output. It is used as part of the `sacadm` command line when `sacadm` adds a port monitor to the system.

`-c` *cmd* | `-o` *streamname* [`-p` *modules*] [`-A` *address* | `-D`] [`-R` *prognum* : *versnum*]
: Format the port monitor-specific information to be used as an argument to `pmadm(1M)`

  The `-c` option specifies the full path name of the server and its arguments. *cmd* must appear as a single word to the shell, and its arguments must therefore be surrounded by quotes.

  The `-o` option specifies the full path name of a FIFO or named stream through which a standing server is actually receiving the connection.

  If the `-p` option is specified, then *modules* will be interpreted as a list of STREAMS modules for the listener to push before starting the service being added. The modules are pushed in the order in which they are specified. *modules* must be a comma-separated list, with no white space included.

  If the `-A` option is specified, then *address* will be interpreted as the server's private address. The listener will monitor this address on behalf of the service and will dispatch all calls arriving on this address directly to the designated service. This option may not be used in conjunction with the `-D` option.

If the `-D` option is specified, then the service is assigned a private address dynamically, that is, the listener will have the transport provider select the address each time the listener begins listening on behalf of this service. For RPC services, this option will be often be used in conjunction with the `-R` option to register the dynamically assigned address with the rpcbinder. This option may not be used in conjunction with the `-A` option.

When the `-R` option is specified, the service is an RPC service whose address, program number, and version number should be registered with the rpcbinder for this transport provider. This registration is performed each time the listener begins listening on behalf of the service. *prognum* and *versnum* are the program number and version number, respectively, of the RPC service.

`nlsadmin` may be invoked by any user to generate reports; all operations that affect a listener's status or configuration may only be run by a super-user.

The options specific to the Service Access Facility may not be used together with any other options.

Errors If successful, `nlsadmin` exits with a status of 0. If `nlsadmin` fails for any reason, it exits with a status greater than or equal to 2. See `-q` option for a return status of 1.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `listen(1M)`, `pmadm(1M)`, `rpcbind(1M)`, `sacadm(1M)`, `attributes(5)`

System Administration Guide: Basic Administration

Notes Dynamically assigned addresses are not displayed in reports as statically assigned addresses are.

Name nscd – name service cache daemon

Synopsis

```
/usr/sbin/nscd  [-f configuration-file] [-g] [-e cachename, yes
 |  no] [-i cachename]
```

Description The nscd daemon is a process that provides a cache for most name service requests. The default *configuration-file* /etc/nscd.conf determines the behavior of the cache daemon. See nscd.conf(4).

nscd provides caching for the passwd(4), group(4), hosts(4), ipnodes(4), exec_attr(4), prof_attr(4), user_attr(4), ethers(4), rpc(4), protocols(4), networks(4), bootparams(4), audit_user(4), auth_attr(4), services(4), netmasks(4), printers(4), project(4) databases through standard libc interfaces, such as gethostbyname(3NSL), getipnodebyname(3SOCKET), gethostbyaddr(3NSL), and others. Each cache has a separate time-to-live for its data; modifying the local database (/etc/hosts, /etc/resolv.conf, and so forth) causes that cache to become invalidated upon the next call to nscd. The shadow file is specifically not cached. getspnam(3C) calls remain uncached as a result.

nscd also acts as its own administration tool. If an instance of nscd is already running, commands are passed to the running version transparently.

When running with per-user lookups enabled (see nscd.conf(4)), nscd forks one and only one child process (that is, a per-user nscd) on behalf of the user making the request. The per-user nscd will use the credentials of the user to open a per-user connection to the name repository configured for the per-user style of lookups. The lookup will be performed in the child process. The results are cached in the process and are available only to the same user. The caches are managed exactly the same as the main nscd daemon manages its own caches. Subsequent requests from the user will be handled by that per-user nscd until it terminates. The per-user nscd uses a configurable inactivity time-to-live (TTL) value and terminates itself after the inactivity TTL expires.

The maximum number of per-user nscds that can be created by the main nscd is configurable (see nscd.conf(4)). After the maximum number of them are created, the main nscd will use an LRU algorithm to terminate less active child nscds as needed.

The main nscd daemon creates, monitors, and manages all the child nscds. It creates a user's own nscd upon receiving the user's first per-user lookup. When the nscd daemon is started, if per-user lookups are enabled, it checks to ensure all conditions are met before getting ready to create a per-user nscd. When the daemon is stopped, it terminates all the per-user nscds under its control.

Per-user nscds use the same configuration as the main nscd. They read and use the same default configuration file or the one specified with the -f command line option. Once the configuration is read, the per-user nscd will use it for its entire lifetime.

Options Several of the options described below require a *cachename* specification. Supported values for *cachename* are: `passwd`, `group`, `hosts`, `ipnodes`, `exec_attr`, `prof_attr`, `user_attr`, `ethers`, `rpc`, `protocols`, `networks`, `bootparams`, `audit_user`, `auth_attr`, `services`, `netmasks`, `printers`, and `project`.

`-f` *configuration-file*
Causes `nscd` to read its configuration data from the specified file.

`-g`
Prints current configuration and statistics to standard output. This is the only option executable by non-root users.

`-e` *cachename*`, yes|no`
Enables or disables the specified cache.

`-i` *cachename*
Invalidate the specified cache.

Examples **EXAMPLE 1** Stopping and restarting the `nscd` daemon.

```
example# svcadm disable system/name-service-cache

example# svcadm enable system/name-service-cache
```

Files `/etc/nscd.conf` Determines athe behavior of the cache daemon

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `svcs(1)`, `svcadm(1M)`, `getspnam(3C)`, `gethostbyname(3NSL)`, `getipnodebyname(3SOCKET)`, `audit_user(4)`, `auth_attr(4)`, `bootparams(4)`, `ethers(4)`, `exec_attr(4)`, `group(4)`, `hosts(4)`, `netmasks(4)`, `networks(4)`, `nscd.conf(4)`, `nsswitch.conf(4)`, `passwd(4)`, `printers(4)`, `prof_attr(4)`, `project(4)`, `protocols(4)`, `rpc(4)`, `services(4)`, `user_attr(4)`, `attributes(5)`

Notes The output from the `-g` option to `nscd` is subject to change. Do not rely upon it as a programming interface.

The `nscd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/name-service-cache
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name nslookup – query Internet name servers interactively

Synopsis

```
nslookup [-option] [name | -] [server]
```

Description The `nslookup` utility is a program to query Internet domain name servers. It has two modes: interactive and non-interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and requested information for a host or domain.

Parameters Interactive mode is entered in the following cases:

1. No arguments are given (the default name server is used).
2. The first argument is a hyphen (-) and the second argument is the host name or Internet address of a name server.

Non-interactive mode is used when the name or Internet address of the host to be looked up is given as the first argument. The optional second argument specifies the host name or address of a name server.

Options can also be specified on the command line if they precede the arguments and are prefixed with a hyphen. For example, to change the default query type to host information, and the initial timeout to 10 seconds, type:

```
nslookup -query=hinfo  -timeout=10
```

Interactive Commands

`host` [*server*]
Look up information for host using the current default server or using server, if specified. If host is an Internet address and the query type is A or PTR, the name of the host is returned. If host is a name and does not have a trailing period, the search list is used to qualify the name. To look up a host not in the current domain, append a period to the name.

`server` *domain*
`lserver` *domain*
Change the default server to *domain*; `lserver` uses the initial server to look up information about *domain*, while `server` uses the current default server. If an authoritative answer can't be found, the names of servers that might have the answer are returned.

`root`
Not implemented.

`finger`
Not implemented.

`ls`
Not implemented.

`view`
Not implemented.

`help`
Not implemented.

`?`
Not implemented.

`exit`
Exits the program.

`set` *keyword*[=*value*]
This command is used to change state information that affects the lookups. Valid keywords are:

`all`
Prints the current values of the frequently used options to `set`. Information about the current default server and host is also printed.

`class=`*value*
Change the query class to one of:

`IN`
the Internet class

`CH`
the Chaos class

`HS`
the Hesiod class

`ANY`
wildcard

The class specifies the protocol group of the information.

(Default = `IN`; abbreviation = `cl`)

`[no]debug`
Turn on or off the display of the full response packet and any intermediate response packets when searching.

(Default = `nodebug`; abbreviation = `[no]deb`)

`[no]d2`
Turn debugging mode on or off. This displays more about what `nslookup` is doing.

(Default = `nod2`)

`domain=`*name*
Sets the search list to *name*.

`[no]search`
If the lookup request contains at least one period but doesn't end with a trailing period, append the domain names in the domain search list to the request until an answer is received.

(Default = `search`)

`port=`*value*
Change the default TCP/UDP name server port to *value*.

(Default = `53`; abbreviation = `po`)

`querytype=`*value*
`type=`*value*
Change the top of the information query.

(Default = `A`; abbreviations = `q`, `ty`)

`[no]recurse`
Tell the name server to query other servers if it does not have the information. (Default = `recurse`; abbreviation = `[no]rec`)

`retry=`*number*
Set the number of retries to number.

`timeout=`*number*
Change the initial timeout interval for waiting for a reply to number seconds.

`[no]vc`
Always use a virtual circuit when sending requests to the server.

(Default = `novc`)

`[no]fail`
Try the next nameserver if a nameserver responds with `SERVFAIL` or a referral (`nofail`) or terminate query (`fail`) on such a response.

(Default = `nofail`)

Files `/etc/resolv.conf`
resolver configuration file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dns/bind
Interface Stability	Volatile

See Also dig(1M), host(1M), named(1M), attributes(5)

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at https://www.isc.org/software/bind/documentation.

Notes BIND 9 nslookup is deprecated and not as full featured as its BIND 8 version. For more features and functionality refer to dig(1M).

nslookup and dig(1M) now report "Not Implemented" as NOTIMP rather than NOTIMPL. This will have impact on scripts that are looking for NOTIMPL.

Name nsupdate – Dynamic DNS update utility

Synopsis

```
nsupdate [-dv] [-y keyname:secret | -k keyfile] [-t timeout]
     [-u udptimeout] [-r udpretries] [filename]
```

Description The `nsupdate` utility submits Dynamic DNS Update requests as defined in RFC 2136 to a name server. This utility allows resource records to be added or removed from a zone without manually editing the zone file. A single update request can contain requests to add or remove more than one resource record.

Zones that are under dynamic control with `nsupdate` or a DHCP server should not be edited by hand. Manual edits could conflict with dynamic updates and cause data to be lost.

The resource records that are dynamically added or removed with `nsupdate` must be in the same zone. Requests are sent to the zone's master servers identified by the `MNAME` field of the zone's SOA record.

Transaction signatures can be used to authenticate the Dynamic DNS updates using the TSIG resource record type described in RFC 2845. The signatures rely on a shared secret that should only be known to `nsupdate` and the name server. Currently, the only supported encryption algorithm for TSIG is HMAC-MD5, which is defined in RFC 2104. Once other algorithms are defined for TSIG, applications will need to ensure that they select the appropriate algorithm as well as the key when authenticating each other. For instance, suitable `key` and `server` statements would be added to `/etc/named.conf` so that the name server can associate the appropriate secret key and algorithm with the IP address of the client application that will be using TSIG authentication. The `nsupdate` utility does not read `/etc/named.conf`.

The `nsupdate` utility uses the `-y` or `-k` option to provide the shared secret needed to generate a TSIG record for authenticating Dynamic DNS update requests. These options are mutually exclusive. See OPTIONS.

Options The following options are supported:

`-d` Operate in debug mode. This provides tracing information about the update requests that are made and the replies received from the name server.

`-k` *keyfile* Read the shared secret from the file *keyfile*, whose name is of the form `K{`*name*`}.+157.+{`*random*`}.private`. For historical reasons, the file `K{`*name*`}.+157.+{`*random*`}.key` must also be present.

`-r` *udpretries* Set the number of UDP retries. The default is 3 seconds. If *udpretries* is set to zero, only one update request is made.

`-t` *timeout* Set *timeout* interval in seconds before update is aborted. The default is 300 seconds. A setting of zero disables the timeout.

`-u` *udptimeout* Set interval in seconds between UDP retires, the default is 3 seconds. A setting of zero causes the interval to be calculated based on the timeout (`-t`) and the number of UDP retries (`-r`).

`-v` Use a TCP connection. Using a TCP connection could be preferable when a batch of update requests is made. By default, `nsupdate` uses UDP to send update requests to the name server.

`-y` *keyname*:*secret* Generate a signature from *keyname*:*secret*, where*keyname* is the name of the key and *secret* is the base64 encoded shared secret.

Use of the `-y` option is discouraged because the shared secret is supplied as a command line argument in clear text and could be visible in the output from `ps(1)` or in a history file maintained by the user's shell.

Input Format The `nsupdate` utility reads input from *filename* or the standard input. Each command is supplied on exactly one line of input. Some commands are for administrative purposes. The others are either update instructions or prerequisite checks on the contents of the zone. These checks set conditions that some name or set of resource records (RRset) either exists or is absent from the zone. These conditions must be met if the entire update request is to succeed. Updates will be rejected if the tests for the prerequisite conditions fail.

Every update request consists of zero or more prerequisites and zero or more updates. This condition allows a suitably authenticated update request to proceed if some specified resource records are present or missing from the zone. A blank input line (or the `send` command) causes the accumulated commands to be sent as one Dynamic DNS update request to the name server.

The command formats and their meaning are as follows:

`server` *servername* [*port*]
Send all dynamic update requests to the name server *servername*. When no `server` statement is provided, `nsupdate` sends updates to the master server of the correct zone. The `MNAME` field of that zone's SOA record identifies the master server for that zone. The *port* argument is the port number on *servername* where the dynamic update requests get sent. If no port number is specified, the default DNS port number of 53 is used.

`local` *address* [*port*]
Send all dynamic update requests using the local *address*. When no `local` statement is provided, `nsupdate` sends updates using an address and port chosen by the system. The *port* argument can also be used to make requests come from a specific port. If no port number is specified, the system assigns one.

`zone` *zonename*
Specify that all updates are to be made to the zone *zonename*. If no `zone` statement is provided, `nsupdate` attempts to determine the correct zone to update based on the rest of the input.

`class` *classname*
Specify the default class. If no class is specified the default class is IN.

`key` *name secret*
Specify that all updates are to be TSIG signed using the *name secret* pair. The `key` command overrides any key specified on the command line with `-y` or `-k`.

`prereq nxdomain` *domain-name*
Require that no resource record of any type exists withthe name *domain-name*.

`prereq yxdomain` *domain-name*
Require that *domain-name* exists (has as at least one resource record, of any type).

`prereq nxrrset` *domain-name* [*class*] *type*
Require that no resource record exists of the specified *type*, *class* and *domain-name*. If *class* is omitted, IN (internet) is assumed.

`prereq yxrrset` *domain-name* [*class*] *type*
Require that a resource record of the specified *type*, *class* and *domain-name* must exist. If *class* is omitted, IN (internet) is assumed.

`prereq yxrrset` *domain-name* [*class*] *type data...*
The *data* from each set of prerequisites of this form sharing a common *type*, *class*, and *domain-name* are combined to form a set of RRs. This set of RRs must exactly match the set of RRs existing in the zone at the given *type*, *class*, and *domain-name*. The *data* are written in the standard text representation of the resource record's RDATA.

`update delete` *domain-name* [*ttl*] [*class*] [*type* [*data...*]]
Delete any resource records named *domain-name*. If *type* and *data* are provided, only matching resource records are removed. The internet class is assumed if *class* is not supplied. The *ttl* is ignored, and is only provided for compatibility.

`update add` *domain-name ttl* [*class*] *type data...*
Add a new resource record with the specified *ttl*, *class* and *data*.

`show`
Display the current message, containing all of the prerequisites and updates specified since the last send.

`send`
Sens the current message. This is equivalent to entering a blank line.

Lines beginning with a semicolon are comments and are ignored.

Examples

EXAMPLE 1 Insert and delete resource records from the zone.

The examples below show how `nsupdate` could be used to insert and delete resource records from the `example.com` zone. Notice that the input in each example contains a trailing blank line so that a group of commands are sent as one dynamic update request to the master name server for `example.com`.

```
# nsupdate
> update delete oldhost.example.com A
```

EXAMPLE 1 Insert and delete resource records from the zone. *(Continued)*

```
> update add newhost.example.com 86400 A 172.16.1.1
> send
```

Any A records for `oldhost.example.com` are deleted. An A record for `newhost.example.com` with IP address 172.16.1.1 is added. The newly-added record has a 1 day TTL (86400 seconds).

```
# nsupdate
> prereq nxdomain nickname.example.com
> update add nickname.example.com 86400 CNAME somehost.example.com
> send
```

The prerequisite condition gets the name server to check that there are no resource records of any type for `nickname.example.com`. If there are, the update request fails. If this name does not exist, a `CNAME` for it is added. This action ensures that when the `CNAME` is added, it cannot conflict with the long-standing rule in RFC 1034 that a name must not exist as any other record type if it exists as a `CNAME`. (The rule has been updated for DNSSEC in RFC 2535 to allow `CNAME`s to have `SIG`, `KEY`, and `NXT` records.)

Files

`/etc/resolv.conf`	used to identify default name server
`K{`*name*`}.+157.+{`*random*`}.key`	base-64 encoding of HMAC-MD5 key created by `dnssec-keygen`(1M).
`K{`*name*`}.+157.+{`*random*`}.private`	base-64 encoding of HMAC-MD5 key created by `dnssec-keygen(1M)`

Bugs The TSIG key is redundantly stored in two separate files. This is a consequence of `nsupdate` using the DST library for its cryptographic operations and could change in future releases.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbind
Interface Stability	External

See Also `named(1M)`, `dnssec-keygen(1M)`, `attributes(5)`

RFC 2136, *RFC 3007*, *RFC 2104*, *RFC 2845*, *RFC 1034*, *RFC 2535*, *RFC 2931*

Notes Source for BIND9 is available in the `SUNWbind9S` package.

Name ntpdate – set the date and time by way of NTP

Synopsis

```
/usr/sbin/ntpdate [-bBdoqsuv] [-a key#] [-e authdelay]
     [-k keyfile] [-m] [-o version] [-p samples]
     [-t timeout] [-w] server...
```

Description The `ntpdate` utility sets the local date and time. To determine the correct time, it polls the Network Time Protocol (NTP) servers on the hosts given as arguments. This utility must be run as root on the local host. It obtains a number of samples from each of the servers and applies the standard NTP clock filter and selection algorithms to select the best of these.

The reliability and precision of `ntpdate` improve dramatically with a greater number of servers. While a single server may be used, better performance and greater resistance to inaccuracy on the part of any one server can be obtained by providing at least three or four servers, if not more.

The `ntpdate` utility makes time adjustments in one of two ways. If it determines that your clock is off by more than 0.5 seconds it simply steps the time by calling `gettimeofday(3C)`. If the error is less than 0.5 seconds, by default, it slews the clock's time with the offset, by way of a call to `adjtime(2)`. The latter technique is less disruptive and more accurate when the offset is small; it works quite well when `ntpdate` is run by `cron` every hour or two. The adjustment made in the latter case is actually 50% larger than the measured offset. This adjustment tends to keep a badly drifting clock more accurate, at some expense to stability. This tradeoff is usually advantageous. At boot time, however, it is usually better to step the time. This can be forced in all cases by specifying the `-b` option on the command line.

The `ntpdate` utility declines to set the date if an NTP server daemon like `xntpd(1M)` is running on the same host. It can be run on a regular basis from `cron(1M)` as an alternative to running a daemon. Doing so once every one to two hours results in precise enough timekeeping to avoid stepping the clock.

Options The following options are supported:

`-a` *key#* Authenticate transactions, using the key number, *key#*.

`-b` Step the time by calling `gettimeofday(3C)`.

`-B` Force the time to always be slewed using the `adjtime(2)` system call, even if the measured offset is greater than +-128 ms. The default is to step the time using `settimeofday(3C)` if the offset is greater than +-128 ms. If this option is used and the offset is much greater than +-128 ms, it can take a long time (hours) to slew the clock to the correct value. During this time the host should not be used to synchronize clients.

`-d` Display what will be done without actually doing it. Information useful for general debugging is also printed.

`-e` *authdelay* Specify an authentication processing delay, *authdelay* in seconds. See `xntpd(1M)` for details. This number is usually small enough to be negligible for purposes of `ntpdate`. However, specifying a value may improve timekeeping on very slow CPU's.

`-k` *keyfile* Read keys from the file *keyfile* instead of the default file, `/etc/inet/ntp.keys`. *keyfile* should be in the format described in `xntpd(1M)`.

`-m` Join multicast group specified in *server* and synchronize to multicast NTP packets. The standard NTP group is 224.0.1.1.

`-o` *version* Force the program to poll as a version 1 or version 2 implementation. By default `ntpdate` claims to be an NTP version 3 implementation in its outgoing packets. However, some older software declines to respond to version 3 queries. This option can be used in these cases.

`-p` *samples* Set the number of samples `ntpdate` acquires from each server. *samples* can be between 1 and 8 inclusive. The default is 4.

`-q` Query only. Do not set the clock.

`-s` Log actions by way of the `syslog(3C)` facility rather than to the standard output — a useful option when running the program from `cron(1M)`.

`-t` *timeout* Set the time `ntpdate` spends, waiting for a response. *timeout* is rounded to a multiple of 0.2 seconds. The default is 1 second, a value suitable for polling across a LAN.

`-u` Use an unprivileged port to send the packets from. This option is useful when you are behind a firewall that blocks incoming traffic to privileged ports, and you want to synchronize with hosts beyond the firewall. The `-d` option always uses unprivileged ports.

`-v` Be verbose. This option causes `ntpdate`'s version identification string to be logged.

`-w` Wait until able to synchronize with a server. When the `-w` option is used together with `-m`, `ntpdate` waits until able to join the group and synchronize.

Files `/etc/inet/ntp.keys` Contains the encryption keys used by `ntpdate`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWntpu

See Also cron(1M), xntpd(1M), adjtime(2), gettimeofday(3C), settimeofday(3C)syslog(3C), attributes(5)

Notes The technique of compensating for clock oscillator errors to improve accuracy is inadequate. However, to further improve accuracy would require the program to save *state* from previous runs.

Name ntpq – standard Network Time Protocol query program

Synopsis `/usr/sbin/ntpq [-inp] [-c` *command*`] [`*host*`] [...]`

Description `ntpq` queries NTP servers which implement the recommended NTP mode 6 control message format, about current state. It can also request changes in that state. The program can be run in interactive mode; or it can be controlled using command line arguments. Requests to read and write arbitrary variables can be assembled, with raw and pretty-`printed` output options available. By sending multiple queries to the server, `ntpq` can also obtain and print a list of peers in a common format.

If one or more request options are included on the command line, `ntpq` sends each of the requests to NTP servers running on each of the hosts given as command line arguments. By default, `ntpq` sends its requests to `localhost`, if hosts are not included on the command line. If no request options are given, `ntpq` attempts to read commands from the standard input and execute them on the NTP server running on the first host given on the command line. Again, `ntpq` defaults to `localhost` if no other host is specified.

`ntpq` uses NTP mode 6 packets to communicate with an NTP server. Thus, it can be used to query any compatible server on the network that permits queries. Since NTP is a UDP protocol, this communication will be somewhat unreliable, especially over large distances. `ntpq` makes one attempt to retransmit requests; requests timeout if the remote host is not heard from within a suitable period.

Options Command line options are described below. Specifying a command line option other than `-i` or `-n` causes the specified query (queries) to be sent, immediately to the indicated host(s). Otherwise, `ntpq` attempts to read interactive format commands from standard input.

`-c` Interpret the next argument as an interactive format command and add it to the list of commands to be executed on the specified host(s). Multiple `-c` options may be given.

`-i` Operate in interactive mode; write prompts to standard output and read commands from standard input.

`-n` Output all host addresses in dotted-quad numeric format rather than converting them to canonical host names.

`-p` Print a list of the peers known to the server as well as a summary of their state. This is equivalent to the `peers` interactive command. See `USAGE` below.

Usage Interactive format commands consist of a keyword followed by up to four arguments. Only enough characters of the full keyword to uniquely identify the command need be typed. Normally, the output of a command is sent to standard output; but this output may be written to a file by appending a '>', followed by a file name, to the command line.

Interactive Commands A number of interactive format commands are executed entirely within the `ntpq` program itself. They do not result in NTP mode 6 requests being sent to a server. If no request options are included on the command line, and if the standard input is a terminal device, `ntpq` prompts for these commands. The interactive commands are described below:

`? [`*command_keyword* `]`	A '?' by itself prints a list of all the command keywords known to the current version of `ntpq`. A '?' followed by a command keyword prints function and usage information about the command.
`timeout`*milliseconds*	Specifies a time out period for responses to server queries. The default is about 5000 milliseconds. Since `ntpq` retries each query once after a time out, the total waiting time for a time out is twice the time out value that is set.
`delay`*milliseconds*	Specifies a time interval to be added to timestamps included in requests which require authentication. This command is used to enable (unreliable) server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized. Currently, the server does not require time stamps in authenticated requests. Thus, this command may be obsolete.
`host`*hostname*	Set the name of the host to which future queries are to be sent. *Hostname* may be either a host name or a numeric address.
`keyid` #	Specify of a key number to be used to authenticate configuration requests. This number must correspond to a key number the server has been configured to use for this purpose.
`passwd`	Allow the user to specify a password at the command line. This will be used to authenticate configuration requests. If an authenticating key has been specified (see `keyid` above), this password must correspond to this key. `ntpq` does not echo the password as it is typed.
`hostnames yes | no`	If "yes" is specified, host names are printed in information displays. If "no" is given, numeric addresses are printed instead. The default is "yes" unless modified using the command line `-n` switch.
`raw`	Print all output from query commands exactly as it is received from the remote server. The only formatting/filtering done on the data is to transform non-ASCII data into printable form.
`cooked`	Causes output from query commands to be "cooked". The values of variables recognized by the server are reformatted,

so that they can be more easily read. Variables which `ntpq` thinks should have a decodable value, but do not, are marked with a trailing '?'.

`ntpversion[ 1 | 2 | 3 ]` Sets the NTP version number which `ntpq` claims in packets (defaults is 3). Note that mode 6 control messages (and modes, for that matter) did not exist in NTP version 1. There appear to be no servers left which demand version 1.

`authenticate[ yes | no ]` The command `authenticate yes` instructs `ntpq` to send authentication with all requests it makes. Normally `ntpq` does not authenticate requests unless they are write requests. Authenticated requests cause some servers to handle requests slightly differently, and can occasionally cause a slowed response if you turn authentication on before doing a peer display. `addvars` *variable_name*[=*value*] [,. . .] `rmvars` *variable_name* [,. . .] `clearvars`

The data carried by NTP mode 6 messages consists of a list of items of the form

variable_name=*value*

where the "=*value*" is ignored, and can be omitted, in requests to the server to read variables. `ntpq` maintains an internal list in which data to be included in control messages can be assembled, and sent. This is accomplished with the `readlist` and `writelist` commands described below. The `addvars` command allows variables and their optional values to be added to the list. If more than one variable is to be added, the list should be comma-separated, and it should not contain white space. The `rmvars` command can be used to remove individual variables from the list; the `clearlist` command removes all variables from the list.

`debug[ more | less | off ]` Turns internal query program debugging on and off.

`quit` Exit `ntpq`.

Control Message Commands Each peer known to an NTP server has a 16 bit integer *association identifier* assigned to it. NTP control messages which carry peer variables must identify the peer that the values correspond to, by including its association ID. An association ID of 0 is special. It indicates the variables are system variables, whose names are drawn from a separate name space.

Control message commands send one or more NTP mode 6 messages to the server, and cause the data returned to be printed in some format. Most commands currently implemented send a single message and expect a single response. The current exceptions are the `peers` `mreadlist` and `mreadvar` commands. The `peers` command sends a preprogrammed series of messages to obtain the data it needs. The `mreadlist` and `mreadvar` commands, iterate over a range of associations.

Control message commands are described below:

`associations`
Obtains and prints a list of association identifiers and peer statuses for in-spec peers of the server being queried. The list is printed in columns. The first of these is an index that numbers the associations from 1, for internal use. The second column contains the actual association identifier returned by the server and the third the status word for the peer. This is followed by a number of columns containing data decoded from the status word. Note that the data returned by the `associations` command is cached internally in `ntpq`. The index is then of use when dealing with "dumb" servers which use association identifiers that are hard for humans to type. For any subsequent commands which require an association identifier as an argument, the identifier can be specified by using the form, *&index*. Here *index* is taken from the previous list.

`lassociations`
Obtains and prints a list of association identifiers and peer statuses for all associations for which the server is maintaining *state*. This command differs from the `associations` command only for servers which retain *state* for out-of-spec client associations. Such associations are normally omitted from the display when the `associations` command is used, but are included in the output of `lassociations`.

`passociations`
Prints association data concerning in-spec peers from the internally cached list of associations. This command performs identically to the `associations` command except that it displays the internally stored data rather than making a new query.

`lpassociations`
Print data for all associations, including out-of-spec client associations, from the internally cached list of associations. This command differs from `passociations` only when dealing with servers which retain *state* for out-of-spec client associations.

`pstatus`*assocID*
Sends a read status request to the server for the given association. The names and values of the peer variables returned will be printed. Note that the status word from the header is displayed preceding the variables, both in hexadecimal and in pigeon English.

`readvar` [*assoc*] [*variable_name*[=*value*] [,. . .]]
Requests that the values of the specified variables be returned by the server by sending a read variables request. If the association ID is omitted or is given as zero the variables are system variables, otherwise they are peer variables and the values returned will be those of

the corresponding peer. Omitting the variable list will send a request with no data which should induce the server to return a default display.

`rv` [*assocID*] [*variable_name*[=*value*] [,. . .]]
An easy-to-type short form for the `readvar` command.

`writevar` *assocID variable_name*=*value* [,. . .]
Like the `readvar` request, except the specified variables are written instead of read.

`readlist` [*assocID*]
Requests that the values of the variables in the internal variable list be returned by the server. If the association ID is omitted or is 0 the variables are assumed to be system variables. Otherwise they are treated as peer variables. If the internal variable list is empty a request is sent without data, which should induce the remote server to return a default display.

`rl` [*assocID*]
An easy-to-type short form of the `readlist` command.

`writelist` [*assocID*]
Like the `readlist` request, except the internal list variables are written instead of read.

`mreadvar` *assocID assocID* [*variable_name*[=*value*] [,. . .]]
Like the `readvar` command except the query is done for each of a range of (nonzero) association IDs. This range is determined from the association list cached by the most recent `associations` command.

`mrv` *assocID assocID* [*variable_name*[=*value*] [,. . .]]
An easy-to-type short form of the `mreadvar` command.

`mreadlist`*assocID assocID*
Like the `readlist` command except the query is done for each of a range of (nonzero) association IDs. This range is determined from the association list cached by the most recent `associations` command.

`mrl`*assocID assocID*
An easy-to-type short form of the `mreadlist` command.

`clockvar` [*assocID*] [*variable_name*[=*value*] [,. . .]]
Requests that a list of the server's clock variables be sent. Servers which have a radio clock or other external synchronization respond positively to this. If the association identifier is omitted or zero the request is for the variables of the “system clock”. This request generally gets a positive response from all servers with a clock. Some servers may treat clocks as pseudo-peers and, hence, can possibly have more than one clock connected at once. For these servers, referencing the appropriate peer association ID shows the variables of a particular clock. Omitting the variable list causes the server to return a default variable display.

`cv` [*assocID*] [*variable_name*[=*value*] [,. . .]]
An easy-to-type short form of the `clockvar` command.

`peers`
Obtains a list of in-spec peers of the server, along with a summary of each peer's state. Summary information includes:

- The address of the remote peer
- The reference ID (0.0.0.0 if the ref ID is unknown)
- The stratum of the remote peer
- The type of the peer (local, unicast, multicast or broadcast) when the last packet was received
- The polling interval in seconds
- The reachability register, in octal
- The current estimated delay offset and dispersion of the peer, all in milliseconds.

The character in the left margin indicates the fate of this peer in the clock selection process. The codes mean:

`SPACE`	Discarded due to high stratum and/or failed sanity checks.
`x`	Designated falsticker by the intersection algorithm.
`.`	Culled from the end of the candidate list.
–	Discarded by the clustering algorithm.
`+`	Included in the final selection set.
`#`	Selected for synchronization; but distance exceeds maximum.
`*`	Selected for synchronization.
`o`	Selected for synchronization, pps signal in use.

Since the `peers` command depends on the ability to parse the values in the responses it gets, it may fail to work from time to time with servers which poorly control the data formats.

The contents of the host field may be given in one of four forms. It may be a host name, an IP address, a reference clock implementation name with its parameter or, `REFCLK`(*implementation number*, *parameter*). On "hostnames no" only IP–addresses will be displayed.

`lpeers`
Like `peers`, except a summary of all associations for which the server is maintaining state is printed. This can produce a much longer list of peers from inadequate servers.

`opeers`
An old form of the `peers` command with the reference ID replaced by the local interface address.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWntpu

See Also attributes(5)

Bugs The peers command is non-atomic. It may occasionally result in spurious error messages about invalid associations occurring and terminating the command.

The timeout value is a fixed constant. As a result, it often waits a long time to timeout, since the fixed value assumes sort of a worst case. The program should improve the time out estimate as it sends queries to a particular host; but it does not.

Name ntptrace – trace a chain of NTP hosts back to their master time source

Synopsis

```
/usr/sbin/ntptrace [-vdn] [-r retries] [-t timeout]
     [server]
```

Description `ntptrace` determines where a given Network Time Protocol (NTP) server gets its time from, and follows the chain of NTP servers back to their master time source. If given no arguments, it starts with *localhost*.

Options The following options are supported:

`-d` Turns on some debugging output.

`-n` Turns off the printing of host names; instead, host IP addresses are given. This may be necessary if a nameserver is down.

`-r` *retries* Sets the number of retransmission attempts for each host.

`-t` *timeout* Sets the retransmission timeout (in seconds); default = 2.

`-v` Prints verbose information about the NTP servers.

Examples **EXAMPLE 1** Sample Output From the `ntptrace` Command

The following example shows the output from the `ntptrace` command:

```
% ntptrace


localhost: stratum 4, offset 0.0019529, synch distance 0.144135
server2.bozo.com: stratum 2, offset 0.0124263, synch distance 0.115784
usndh.edu: stratum 1, offset 0.0019298, synch distance 0.011993, refid 'WWVB'
```

On each line, the fields are (left to right):

- The server's host name
- The server's stratum
- The time offset between that server and the local host (as measured by `ntptrace`; this is why it is not always zero for *localhost*)
- The host's synchronization distance
- The reference clock `ID` (only for stratum-1 servers)

All times are given in seconds. Synchronization distance is a measure of the goodness of the clock's time.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWntpu

See Also xntpd(1M), attributes(5)

Bugs This program makes no attempt to improve accuracy by doing multiple samples.

Name obpsym – Kernel Symbolic Debugging for OpenBoot Firmware

Synopsis

```
modload -p misc/obpsym
```

Description `obpsym` is a kernel module that installs OpenBoot callback handlers that provide kernel symbol information to OpenBoot. OpenBoot firmware user interface commands use the callbacks to convert numeric *addresses* to kernel symbol names for display purposes, and to convert kernel symbol names to numeric *literals* allowing symbolic names to be used as input arguments to user interface commands.

Once `obpsym` is installed, kernel symbolic names may be used anywhere at the OpenBoot firmware's user interface command prompt in place of a literal (numeric) string. For example, if `obpsym` is installed, the OpenBoot firmware commands `ctrace` and `dis` typically display symbolic names and offsets in the form *modname:symbolname + offset*. User interface Commands such as `dis` can be given a kernel symbolic name such as `ufs:ufs_mount` instead of a numeric address.

Placing the command

```
forceload: misc/obpsym
```

into the `system(4)` file forces the kernel module `misc/obpsym` to be loaded and activates the kernel callbacks during the kernel startup sequence.

`obpsym` may be useful as a kernel debugger in situations where other kernel debuggers are not useful. For example, on SPARC machines, if `obpsym` is loaded, you may be able to use the OpenBoot firmware's `ctrace` command to display symbolic names in the stack backtrace after a watchdog reset.

Kernel Symbolic Name Syntax The syntax for a kernel symbolic name is:

[*module-name* `:`] *symbol-name*

Where *module-name* is the name of the kernel module that the symbol *symbol-name* appears in. A `NULL` module name is taken as "all modules, in no particular order" by `obpsym`. The module name `unix` is equivalent to a `NULL` module name, so that conflicts with words defined in the firmware's vocabulary can be avoided.

Typically, OpenBoot firmware reads a word from the input stream and looks the word up in its internal *vocabulary* before checking if the word is a *literal*. Thus, kernel symbols, such as `reset` may be given as `unix:reset` to avoid the unexpected side effect of the firmware finding and executing a matching word in its vocabulary.

Files

`/etc/system`	system configuration information file
`/platform/`*platform-name*`/kernel/misc/obpsym`	

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcar

See Also kadb(1M), kernel(1M), modload(1M), modunload(1M), uname(1), system(4), attributes(5)

OpenBoot 2.x Command Reference Manual

Warnings Some OpenBoot firmware user interface commands may use system resources incompatibly with the way they are used by the Unix kernel. These commands and the use of this feature as a kernel debugger may cause interactions that the Unix kernel is not prepared to deal with. If this occurs, the Unix kernel and/or the OpenBoot firmware user interface commands may react unpredictably and may panic the system, or may hang or may cause other unpredictable results. For these reasons, the use of this feature is only minimally supported and recommended to be used only as a kernel debugger of "last resort".

If a breakpoint or watchpoint is triggered while the console frame buffer is powered off, the system can crash and be left in a state from which it is difficult to recover. If one of these is triggered while the monitor is powered off, you will not be able to see the debugger output.

Notes *platform-name* can be found using the `-i` option of uname(1)

`obpsym` is supported only on architectures that support OpenBoot firmware.

On some systems, OpenBoot must be completely RAM resident so the `obpsym` symbol callback support can be added to the firmware, if the firmware doesn't include support for the symbol callbacks. On these systems, `obpsym` may complain that it requires that "you must use ramforth to use this module".

See the *OpenBoot 2.x Command Reference Manual* for details on how to use the *ramforth* command, how to place the command into *nvramrc*, and how to set *use-nvramrc?* to `true`. On systems with version 1.x OpenBoot firmware, *nvramrc* doesn't exist, and the *ramforth* command must be typed manually after each reset, in order to use this module.

Once installed, the symbol table callbacks can be disabled by using the following OpenBoot firmware command:

```
0 0 set-symbol-lookup
```

Name ocfserv – OCF server

Synopsis

```
ocfserv [-D] [-p path]
```

Description The OCF server, `ocfserv`, is a per-host daemon that acts as the central point of communications with all smartcards connected to the host. Applications that need to use a smartcard can do so by using the APIs in `libsmartcard.so` or `smartcard.jar`. The internal implementation of these APIs communicates with `ocfserv` to perform the requested function.

`inetd(1M)` automatically starts the `ocfserv` command when it is needed. Once started, `ocfserv` runs forever. If `ocfserv` is killed or crashes, it restarts automatically if necessary.

Because `ocfserv` is run automatically, there really is not a reason to run it manually. You must have root privileges to execute this utility.

Options The following options are supported:

`-D` Run `ocfserv` in debug mode.

`-p` *path* Specify property file name.

Exit Status The following exit values are returned:

`0` Successful completion.

>`0` An error occurred.

Files `/etc/smartcard/opencard.properties` File where server stores properties

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWocf

See Also `svcs(1)`, `inetd(1M)`, `inetadm(1M)`, `smartcard(1M)`, `svcadm(1M)`, `attributes(5)smartcard(5)`, `smf(5)`

Notes The `ocfserv` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/ocfserv
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name oplhpd – Hot plug daemon for SPARC Enterprise Server line

Synopsis

```
/usr/platform/SUNW,SPARC-Enterprise/lib/sparcv9/lib/oplhpd
```

Description The hot plug daemon for SPARC Enterprise Servers is a daemon process that runs on the SUNW,SPARC-Enterprise family of servers. The daemon is started by the service management facility (see smf(5)) and communicates with the service processor when hot plug PCI cassettes change their dynamic reconfiguration state.

The service FMRI for `oplhpd` is:

```
svc:/platform/sun4u/oplhpd:default
```

A domain supports only one running `oplhpd` process at a time.

Errors OPLHPD uses syslog(3C) to report status and error messages. All of the messages are logged with the LOG_DAEMON facility.

Error messages are logged with the LOG_ERR and LOG_NOTICE priorities, and informational messages are logged with the LOG_DEBUG priority. The default entries in the `/etc/syslog.conf` file log all of the OPLHPD error messages to the `/var/adm/messages` `log`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdcsu, SUNWdcsr
Interface Stability	Evolving

See Also svcs(1), inetadm(1M), svcadm(1M), syslog(3C), syslog.conf(4), attributes(5), smf(5)

Name parse_dynamic_clustertoc – parse clustertoc file based on dynamic entries

Synopsis

cdrom/export/exec/sparc.Solaris_2.x/sbin/install.d/parse_dynamic_clustertoc

cdrom/export/exec/i386.Solaris_2.x/sbin/install.d/parse_dynamic_clustertoc

Description This script parses the clustertoc file before the suninstall(1M) process is run. parse_dynamic_clustertoc is called by a modified sysconfig script on the install CD. When parse_dynamic_clustertoc runs, it reads the clustertoc and when it encounters SUNW_CSRMBRIFF lines, it either checks the platform using the script's builtin function, or calls an external script. The script exits with a 0 if the cluster entry is included, otherwise it will be ignored. If the cluster entry is to be included, the SUNW_CSRMBRIFF =(*test test_arg*) *cluster* line is converted to SUNW_CSRMEMBER =*cluster*.

Examples **EXAMPLE 1** Checking For an SX Framebuffer

The following is an example of a simple external test to check for an SX Framebuffer. The entry in the clustertoc file is shown and following that is the script that must be placed in the install.d/dynamic_test directory.

```
SUNW_CSRMBRIFF=(smcc.dctoc sx)SUNWCsx
#! /bin/sh
#
# Likewise, this file is expected to live under $(TESTDIR).
#
case "$1"
in
        sx)     prtconf -p  | grep 'SUNW,sx' 1> /dev/null;;
esac
```

Files *cdrom*/Solaris_2.x/locale/C/.clustertoc.dynamic
Dynamic version of the clustertoc file

cdrom/export/exec/sparc.Solaris_2.x/sbin/install.d/dynamic_test
Directory that contains any additional tests

cdrom/export/exec/i386.Solaris_2.x/sbin/install.d/dynamic_test
Directory that contains any additional tests

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SHWPcdrom (Solaris CD)

See Also suninstall(1M), clustertoc(4), attributes(5)

Name passmgmt – password files management

Synopsis

```
passmgmt -a options name
passmgmt -m options name
passmgmt -d name
```

Description The `passmgmt` command updates information in the password files. This command works with both `/etc/passwd` and `/etc/shadow`.

`passmgmt -a` adds an entry for user *name* to the password files. This command does not create any directory for the new user and the new login remains locked (with the string `*LK*` in the password field) until the `passwd(1)` command is executed to set the password.

`passmgmt -m` modifies the entry for user *name* in the password files. The name field in the `/etc/shadow` entry and all the fields (except the password field) in the `/etc/passwd` entry can be modified by this command. Only fields entered on the command line will be modified.

`passmgmt -d` deletes the entry for user *name* from the password files. It will not remove any files that the user owns on the system; they must be removed manually.

`passmgmt` can be used only by the super-user.

Options

`-c` *comment*	A short description of the login, enclosed in quotes. It is limited to a maximum of 128 characters and defaults to an empty field.
`-e` *expire*	Specify the expiration date for a login. After this date, no user will be able to access this login. The expire option argument is a date entered using one of the date formats included in the template file `/etc/datemsk`. See `getdate(3C)`.
`-f` *inactive*	The maximum number of days allowed between uses of a login ID before that ID is declared invalid. Normal values are positive integers. A value of `0` defeats the status. Changing the password reactivates an account for the inactivity period.
`-g` *gid*	GID of *name*. This number must range from 0 to the maximum non-negative value for the system. The default is 1.
`-h` *homedir*	Home directory of *name*. It is limited to a maximum of 256 characters and defaults to `/usr/`*name*.
`-K` *key=value*	Set a *key=value* pair. See `user_attr(4)`, `auth_attr(4)`, and `prof_attr(4)`. The valid *key=value* pairs are defined in `user_attr(4)`, but the "type" key is subject to the `usermod(1M)` and `rolemod(1M)` restrictions. Multiple *key=value* pairs may be added with multiple `-K` options.

`-k` *skel_dir* — A directory that contains skeleton information (such as `.profile`) that can be copied into a new user's home directory. This directory must already exist. The system provides the `/etc/skel` directory that can be used for this purpose.

`-l` *logname* — This option changes the *name* to *logname*. It is used only with the `-m` option. The total size of each login entry is limited to a maximum of 511 bytes in each of the password files.

`-o` — This option allows a UID to be non-unique. It is used only with the `-u` option.

`-s` *shell* — Login shell for *name*. It should be the full pathname of the program that will be executed when the user logs in. The maximum size of *shell* is 256 characters. The default is for this field to be empty and to be interpreted as `/usr/bin/sh`.

`-u` *uid* — UID of the *name*. This number must range from 0 to the maximum non-negative value for the system. It defaults to the next available UID greater than 99. Without the `-o` option, it enforces the uniqueness of a UID.

Files

```
/etc/passwd
/etc/shadow
/etc/opasswd
/etc/oshadow
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also `passwd(1)`, `rolemod(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `auth_attr(4)`, `passwd(4)`, `prof_attr(4)`, `shadow(4)`, `user_attr(4)`, `attributes(5)`

Exit Status The `passmgmt` command exits with one of the following values:

`0` Success.

`1` Permission denied.

`2` Invalid command syntax. Usage message of the `passmgmt` command is displayed.

`3` Invalid argument provided to option.

`4` UID in use.

`5` Inconsistent password files (for example, *name* is in the `/etc/passwd` file and not in the `/etc/shadow` file, or vice versa).

6 Unexpected failure. Password files unchanged.

7 Unexpected failure. Password file(s) missing.

8 Password file(s) busy. Try again later.

9 *name* does not exist (if `-m` or `-d` is specified), already exists (if `-a` is specified), or `logname` already exists (if `-m -l` is specified).

Notes Do not use a colon (`:`) or RETURN as part of an argument. It is interpreted as a field separator in the password file. The `passmgmt` command will be removed in a future release. Its functionality has been replaced and enhanced by `useradd`, `userdel`, and `usermod`. These commands are currently available.

This command only modifies password definitions in the local `/etc/passwd` and `/etc/shadow` files. If a network nameservice such as NIS or NIS+ is being used to supplement the local files with additional entries, `passmgmt` cannot change information supplied by the network nameservice.

Name patchadd – apply a patch package to a system running the Solaris operating system

Synopsis

```
patchadd [-dun] [-G] [-B backout_dir] [-k keystore]
     [-P passwd] [-t] [-x proxy] {patch} |
     {-M patch_location [patch_list]} [-C net_install_image |
     -R client_root_path | -S service]

patchadd -p
     [-C net_install_image | -R client_root_path | -S service]
```

Description `patchadd` applies a patch package to a system running the Solaris 2.*x* operating environment or later Solaris environments (such as Solaris 10) that are compatible with Solaris 2.*x*. This patch installation utility cannot be used to apply Solaris 1 patches. `patchadd` must be run as root.

The `patchadd` command has the following forms:

- The first form of `patchadd` installs one or more patches to a system, client, service, or to the miniroot of a Net Install Image.
- The second form of `patchadd` displays installed patches on the client, service, or to the miniroot of a Net Install Image.

Starting with version 10 of the Solaris operating system, `patchadd` performs validity and dependency checking among a collection of patches that you specify with the `-M` source specifier. See the description of `-M` under OPERANDS, below.

With respect to `zones(5)`, when invoked in the global zone, by default, `patchadd` patches all appropriate packages in all zones. Patching behavior on system with `zones` installed varies according to the following factors:

- use of the `-G` option (described below)
- setting of the `SUNW_PKG_ALLZONES` variable in the `pkginfo` file (see `pkginfo(4)`)
- type of zone, global or local (non-global) in `patchadd` which is invoked

The interaction of the factors above is specified in "Interaction of `-G` and `pkginfo` Variable in Zones," below.

When you add patches to packages on a Solaris system with zones installed, you will see numerous zones-related messages, the frequency and content of which depend on whether you invoke `patchadd` in a global or local zone, the setting of `SUNW_PKG_ALLZONES`, and the use of the `-G` option.

Note that if you apply a patch that modifies objects in the boot archive, you will need to run the `bootadm` command shown below. This is true whether the patch is installed individually or as part of a cluster, using `install_cluster` and specifying an alternate root.

```
# bootadm update_archive -R /altroot
```

If you do not enter the preceding command, you will need to boot twice to build a clean boot archive.

The *patch*, `-M`, `-C`, `-R`, and `-S` arguments shown in the SYNOPSIS are described under OPERANDS, following OPTIONS.

Options The following options are supported:

`-B` *backout_dir*
Saves backout data to a directory other than the package database. Specify *backout_dir* as an absolute path name.

`-d`
Does not back up the files to be patched. *The patch cannot be removed.*

`-G`
Add patch(es) to packages in the current zone only. When used in the global zone, the patch is added to packages in the global zone only and is not propagated to packages in any existing or yet-to-be-created non-global zone. When used in a non-global zone, the patch is added to packages in the non-global zone only. See "Interaction of `-G` and `pkginfo` Variable in Zones,", below.

`-k` *keystore*
Use *keystore* as the location to get trusted certificate authority certificates when verifying digital signatures found in each patch. If no keystore is specified, then the default keystore locations are searched for valid trusted certificates. See `KEY STORE LOCATIONS` in `pkgadd(1M)` for more information.

`-n`
Tells `patchadd` to ignore the signature and not to validate it. This should be used only when the content of the patch is known and trusted, and is primarily included to allow `patchadd` to apply a patch on systems without the ability to verify the patch signature, such as Solaris 8.

`-p`
In the second form, displays a list of the patches currently applied.

`-P` *passwd*
Password to use to decrypt the keystore specified with `-k`, if required. See `PASS PHRASE ARGUMENTS` in `pkgadd(1M)` for more information about the format of this option's argument.

`-t`
Maintains the `patchadd` return codes from the Solaris release prior to Solaris 10. On a system with `zones(5)` installed, a return code of 0 indicates success. Any other return code indicates failure.

`-u`
Turns off validation against other required or incompatible patches. Use extreme caution when using this option. Its use can precipitate unanticipated bad consequences.

`-x` *proxy*
Specify a HTTP[S] proxy to use when downloading packages The format of proxy is *host*:*port*, where *host* is the hostname of the HTTP[S] proxy, and *port* is the port number associated with the proxy. This switch overrides all other methods of specifying a proxy. See ENVIRONMENT VARIABLES in `pkgadd(1M)` for more information on alternate methods of specifying a default proxy.

Operands The following operands are supported:

Sources `patchadd` must be supplied a source for retrieving the patch. Specify sources using the syntax shown below.

patch
The absolute path name to *patch_id* or a URI pointing to a signed patch. `/var/sadm/spool/patch/104945-02` is an example of a *patch*. `https://syrinx.eng:8887/patches/104945-02` is an example of a URI pointing to a signed patch.

`-M` *patch_location* [*patch_list*]
Specifies the patches to be installed by directory location or URL and, optionally, the name of a file containing a patch list.

When using a directory as the *patch_location*, specify that directory as an absolute path name. Specify a URL as the server and path name that contains the spooled patches. The optional *patch_list* is the name of the file at a specified location containing the patches to be installed.

`-M` *patch_location patch_id* [*patch_id...*]
Specifies the patches to be installed by directory location or URL, and patch number.

To use the directory location or URL and the patch number, specify *patch_location* as the absolute path name of the directory that contains spooled patches. Specify a URL as the server and path name that contains the spooled patches. Specify *patch_id* as the patch number of a given patch. `104945-02` is an example of a *patch_id*. `104945-02` is also an example of a patchid in `104945-02.jar`.

Note that `patchadd` does not require a list of patches. Among a collection of patches—residing in a directory, specified in a list, or entered on a command line—`patchadd` performs validity and dependency checking. Specifically, the command does the following:

- Determines whether a patch is applicable for a system. For example, if the package to be patched is not installed, `patchadd` does not attempt to add the patch.
- Establishes dependencies among valid patches and orders the installation of patches accordingly.

With the "Zones Parallel Patching" feature, patches can be applied to zones in parallel. After a patch is installed successfully in the global zone, the patch installation software will start a number of processes whose task is to apply patches to zones. The number of processes to be started would be determined by the `num_proc` parameter in the configuration file `/etc/patch/pdo.conf`.

The number of processes to be started is determined in the following order:

1. The value of the `num_proc` parameter. Setting this to 1 retains the current behavior of the patch system.
2. The number of online CPUs in the system.

The upper bound is the number of configured Solaris zones.

Most users will find the easiest way to specify a source for `patchadd` is to specify only a *patch_location* containing a set of patches.

Destinations

By default, `patchadd` applies a patch to the specified destination. If no destination is specified, then the current system (the one with its root filesystem mounted at `/`) is assumed to be the destination for the patch. You can specify a destination in the following ways:

`-C` *net_install_image*
Patches the files located on the miniroot on a Net Install Image created by `setup_install_server`. Specify *net_install_image* as the absolute path name to a Solaris 8 or compatible version boot directory. See `EXAMPLES`.

You should use the `-C` option only to install patches that are recommended for installation to the miniroot. Patches that are recommended for installation to the miniroot usually include install-related patches such as package commands, and Sun install and patch installation tools. If you apply too many patches to the miniroot it can grow too large to fit into memory during a net installation of Solaris. Use the `-B` option and the `-C` option together so the miniroot does not get too large. See `-B`, above.

Note that in the current release and in certain versions of Solaris 10, the miniroot is compressed. To determine whether the miniroot is compressed on your system, look for a file called `sparc.miniroot` or `x86.miniroot` under `/boot`, on the boot medium. Before you can patch a compressed miniroot, you must perform certains steps. See "Patching a Compressed Miniroot" below.

`-R` *client_root_path*
Locates all patch files generated by `patchadd` under the directory *client_root_path*. *client_root_path* is the directory that contains the bootable root of a client from the server's perspective. Specify *client_root_path* as the absolute path name to the beginning of the directory tree under which all patch files generated by `patchadd` are to be located. `-R` cannot be specified with the `-S` option. See `NOTES`.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-S` *service*
Specifies an alternate service (for example, `Solaris_8`). This service is part of the server and client model, and can only be used from the server's console. Servers can contain shared `/usr` file systems that are created by `smosservice(1M)`. These service areas can then be made available to the clients they serve. `-S` cannot be specified with the `-R` option. See `NOTES`.

Patching a Compressed Miniroot

The Solaris operating system uses a compressed miniroot. The compressed miniroot was adopted first in Solaris for x86 and then in Solaris for SPARC over the course of Solaris 10 update releases. See below for an easy way to determine whether your Solaris system uses a compressed miniroot.

To patch a system with a compressed miniroot (full or partial), you must unpack and then repack the miniroot before and after running `patchadd` with the `-C` destination specifier. Use the procedure shown below and accompanying example commands.

1. Unpack the compressed miniroot:

```
# /boot/solaris/bin/root_archive unpackmedia \
/export/home/altuser/testdir /export/home/altuser/mr
```

2. Run `patchadd` with `-C` to patch the miniroot:

```
# patchadd -C /export/home/altuser/mr \
/var/sadm/spool/104945-02
```

3. Repack the miniroot:

```
# /boot/solaris/bin/root_archive packmedia \
/export/home/altuser/testdir /export/home/altuser/mr
```

At this point, you can use `setup_install_server(1M)` to install the patched miniroot on an install server. See `root_archive(1M)` for a description of that command.

To determine whether a Solaris image uses a compressed miniroot, check for the presence of either an `x86.miniroot` or `sparc.miniroot` file under `/boot` on the boot medium.

Interaction of `-G` and `pkginfo` Variable in Zones

The following list specifies the interaction between the `-G` option and the `SUNW_PKG_ALLZONES` variable (see `pkginfo(4)`) when adding a patch in global and local (non-global) zones.

global zone, `-G` specified
If *any* packages have `SUNW_PKG_ALLZONES` set to true: Error; nothing changes.

If *no* packages have `SUNW_PKG_ALLZONES` set to true: Apply patch to package(s) in global zone only.

global zone, `-G` not specified
If *any* packages have `SUNW_PKG_ALLZONES` set to true: Apply patch to appropriate package(s) in all zones.

If *no* packages have `SUNW_PKG_ALLZONES` set to true: Apply patch to appropriate package(s) in all zones.

local zone, `-G` specified or not specified
If *any* packages have `SUNW_PKG_ALLZONES` set to true: Error; nothing changes.

If *no* packages have `SUNW_PKG_ALLZONES` set to true: Apply patch package(s) in local zone only.

Keystore Locations See the section KEYSTORE LOCATIONS in the `pkgadd(1M)` man page for details.

Keystore And Certificate Formats See the section KEYSTORE AND CERTIFICATE FORMATS in the `pkgadd(1M)` man page for details.

Examples The examples in this section are all relative to the `/usr/sbin` directory.

EXAMPLE 1 Installing a Patch to a Standalone Machine

The following example installs a patch to a standalone machine:

```
example# patchadd /var/sadm/spool/104945-02
```

EXAMPLE 2 Installing a Patch to a Client From the Server's Console

The following example installs a patch to a client from the server's console:

```
example# patchadd -R /export/root/client1  /var/sadm/spool/104945-02
```

EXAMPLE 3 Installing a Patch to a Service From the Server's Console

The following example installs a patch to a service from the server's console:

```
example# patchadd -S Solaris_8 /var/sadm/spool/104945-02
```

EXAMPLE 4 Installing Multiple Patches in a Single Invocation

The following example installs multiple patches in a single `patchadd` invocation:

```
example# patchadd -M /var/sadm/spool 104945-02 104946-02 102345-02
```

EXAMPLE 5 Installing Multiple Patches Specifying List of Patches to Install

The following example installs multiple patches specifying a file with the list of patches to install:

```
example# patchadd -M /var/sadm/spool patchlist
```

EXAMPLE 6 Installing Multiple Patches to a Client and Saving the Backout Data

The following example installs multiple patches to a client and saves the backout data to a directory other than the default:

```
example# patchadd -M /var/sadm/spool -R /export/root/client1 \
     -B /export/backoutrepository 104945-02 104946-02 102345-02
```

EXAMPLE 7 Installing a Patch to a Solaris 8 or Compatible Version Net Install Image

The following example installs a patch to a Solaris 8 or compatible version Net Install Image:

```
example# patchadd -C /export/Solaris_8/Tools/Boot \
    /var/sadm/spool/104945-02
```

EXAMPLE 8 Installing a Patch to a Compressed Miniroot

The following example installs a patch to a compressed miniroot, such as one finds on a Solaris x86 machine that supports GRUB–style booting. This example assumes that `/export/Solaris_11/Tools/Boot` contains the unpacked miniroot. After applying the patch, the miniroot needs to be repacked

```
example# patchadd -C /export/Solaris_11/Tools/Boot \
/var/sadm/spool/104945-02
```

See "Patching a Compressed Miniroot," above, for information on Solaris versions that use a compressed miniroot.

EXAMPLE 9 Installing a Patch to an Uncompressed Miniroot

The following example installs a patch to a miniroot on a Solaris machine that does not have a compressed miniroot.

```
example# patchadd -C /export/Solaris_9/Tools/Boot \
/var/sadm/spool/104945-02
```

See "Patching a Compressed Miniroot," above, for information on Solaris versions that use a compressed miniroot.

EXAMPLE 10 Displaying the Patches Installed on a Client

The following example displays the patches installed on a client:

```
example# patchadd -R /export/root/client1 -p
```

Note the caveat on the use of the `-R` option in the description of that option, above.

EXAMPLE 11 Installing a Digitally Signed Set of Patches

The following example installs multiple patches, some of which have been signed, using the supplied keystore, password, and HTTP proxy.

```
example# patchadd -k /etc/mycerts -P pass:abcd -x webcache.eng:8080 \
-M http://www.sun.com/solaris/patches/latest 101223-02 102323-02
```

Files One configuration file of note:

`/etc/patch/pdo.conf` Patch configuration file. Can be used to configure "Zones Parallel Patching" feature.

Exit Status The following exit values are returned:

`0`
Successful completion.

`>0`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWswmt, SUNWcsu
Interface Stability	Evolving

See Also `cpio(1)`, `pkginfo(1)`, `patchrm(1M)`, `pkgadd(1M)`, `pkgadm(1M)`, `pkgchk(1M)`, `pkgrm(1M)`, `setup_install_server(1M)`, `smpatch(1M)`, `showrev(1M)`, `pdo.conf(4)`, `pkginfo(4)`, `attributes(5)`, `grub(5)`, `zones(5)`

Diagnostics The following messages might help in determining some of the most common problems associated with installing a patch.

Patch Installation errors Message

```
The prepatch script exited with return code retcode.
patchadd is terminating.
```

Explanation and Recommended Action
The prepatch script supplied with the patch exited with a return code other than `0`. Run a script trace of the prepatch script and find out why the prepatch had a bad return code. Add the `-x` option to the first line of the prepatch script to fix the problem and run `patchadd` again.

Message

```
The signature on patch patch_id was unable to be verified.
patchadd is terminating.
```

Explanation and Recommended Action
The digital signature on a patch was unable to be verified given the keystore in use and the signature on the patch. Check the keystore to make sure it has the requisite trust anchor(s) required to validate the signature on the package and that the package has not been tampered with.

Message

```
The postpatch script exited with return code retcode.
Backing out patch.
```

Explanation and Recommended Action

The postpatch script provided with the patch exited with an error code other than `0`. This script is mostly used to cleanup files (that is, when a package is known to have ownership or permission problems) attributes that do not correspond to the patch package's objects. After the user has noted all validation errors and taken the appropriate action for each one, the user should re-run `patchadd` using the `-u` (unconditional) option. This time, the patch installation will ignore validation errors and install the patch anyway.

Message

```
Insufficient space in /var/sadm/patch to save old files.
(For 2.4 systems and previous)
```

Explanation and Recommended Action

There is insufficient space in the `/var/sadm/patch` directory to save old files. The user has three options for handling this problem: Use the `-B` option while invoking `patchadd`. This option will direct `patchadd` to: save the backout data to the user specified file system, generate additional disk space by deleting unneeded files, or override the saving of the old files by using the `-d` (do not save) option when running `patchadd`.

If the user elects not to save the old versions of the files to be patched, `patchrm` *cannot* be used. One way to regain space on a system is to remove the save area for previously applied patches. Once the user has decided that it is unlikely that a patch will be backed out, the user can remove the files that were saved by `patchadd`. The following commands should be executed to remove the saved files for `patch`*patch_id*:

```
cd /var/sadm/patch/patch_id
rm -r save/*
rm .oldfilessaved
```

After these commands have been executed, `patch` *patch_id* can no longer be backed out.

Message

```
Insufficient space in /var/sadm/pkg/PKG/save to save old files.
(For 2.5 systems and later)
```

Explanation and Recommended Action

There is insufficient space in the `/var/sadm/pkg/PKG/save` directory to save old files. The user has three options for handling this problem: (1) Use the `-B` option while invoking `patchadd`. This option will direct `patchadd` to save the backout data to the user specified file system. (See synopsis above.) (2) Generate additional disk space by deleting unneeded files, or (3) override the saving of the old files by using the `-d` (do not save)

option when running `patchadd`. However, if the user elects not to save the old versions of the files to be patched, `patchrm` *cannot* be used. One way to regain space on a system is to remove the save area for previously applied patches. Once the user has decided that it is unlikely that a patch will be backed out, the user can remove the files that were saved by `patchadd`. The following commands should be executed to remove the saved files for patch *patch_id*:

```
cd /var/sadm/pkg/pkgabbrev/save
rm -r patch_id
```

After these commands have been executed, `patch` *patch_id* can no longer be backed out.

Message

```
Save of old files failed.
(For 2.4 systems and previous)
```

Explanation and Recommended Action

Before applying the patch, the patch installation script uses `cpio` to save the old versions of the files to be patched. This error message means that the `cpio` failed. The output of the `cpio` would have been preceded this message. The user should take the appropriate action to correct the `cpio` failure. A common reason for failure will be insufficient disk space to save the old versions of the files. The user has two options for handling insufficient disk space: (1) generate additional disk space by deleting unneeded files, or (2) override the saving of the old files by using the `-d` option when running `patchadd`. However if the user elects not to save the old versions of the files to be patched, the patch *cannot* be backed out.

Message

```
Pkgadd of pkgname package failed with error code code.
See /tmp/log.patch_id for reason for failure.
```

Explanation and Recommended Action

The installation of one of the patch packages failed. `patchadd` will backout the patch to leave the system in its pre-patched state. See the log file for the reason for failure. Correct the problem and reapply the patch.

Message

```
Pkgadd of pkgname package failed with error code code.
Will not backout patch...patch re-installation.
Warning: The system may be in an unstable state!
See /tmp/log.patch_id for reason for failure.
```

Explanation and Recommended Action

The installation of one of the patch packages failed. `patchadd` will *not* backout the patch. You may manually backout the patch using `patchrm`, then re-apply the entire patch. Look in the log file for the reason `pkgadd` failed. Correct the problem and re-apply the patch.

Message

```
patchadd is unable to find the INST_RELEASE file. This file
must be present for patchadd to function correctly.
```

Explanation and Recommended Action

The `INST_RELEASE` file is missing from the system. This file is created during either initial installation or during an update.

Message

```
A previous installation of patch patch_id was invoked
that saved files that were to be patched. Since files
were saved, you must run this instance of patchadd
without the -d option.
```

Explanation and Recommended Action

If a patch was previously installed without using the `-d` option, then the re-installation attempt must also be invoked without the `-d` option. Execute `patchadd` without the `-d` option.

Message

```
A previous installation of patch patch_id was invoked
with the -d option. (i.e. Do not save files that would
be patched) Therefore, this invocation of patchadd
must also be run with the -d option.
```

Explanation and Recommended Action

If a patch was previously installed using the `-d` option, then the re-installation attempt must also be invoked with the `-d` option. Execute `patchadd` with the `-d'` option.

Message

```
Illegal character found during parsing. Read the man page
for pdo config file.
```

Explanation and Recommended Action

The `/etc/patch/pdo.conf` follows a specific layout. Each entry in this file should conform to this layout. See `pdo.conf(4)`.

Diagnostic Reference

The patch installation messages listed below are not necessarily considered errors, as indicated in the explanations given. These messages are, however, recorded in the patch installation log for diagnostic reference.

Message

```
Package not patched:
PKG=SUNxxxx
Original package not installed
```

Explanation and Recommended Action

One of the components of the patch would have patched a package that is not installed on your system. This is not necessarily an error. A patch may fix a related bug for several packages.

For example, suppose a patch fixes a bug in both the online-backup and `fddi` packages. If you had online-backup installed but didn't have `fddi` installed, you would get the message :

```
Package not patched:
PKG=SUNWbf
Original package not installed
```

This message only indicates an error if you thought the package was installed on your system. If this is the case, take the necessary action to install the package, backout the patch (if it installed other packages) and re-install the patch.

Message

```
Package not patched:
PKG=SUNxxx
ARCH=xxxxxxx
VERSION=xxxxxxx
Architecture mismatch
```

Explanation and Recommended Action

One of the components of the patch would have patched a package for an architecture different from your system. This is not necessarily an error. Any patch to one of the architecture-specific packages might contain one element for each of the possible architectures. For example, assume you are running on a `sun4u`. If you were to install a patch to package `SUNWcar`, you would see the following (or similar) messages:

```
Package not patched:
PKG=SUNWcar
ARCH=sparc.sun4c
VERSION=11.5.0,REV=2.0.18
Architecture mismatch

Package not patched:
PKG=SUNWcar
ARCH=sparc.sun4u
VERSION=11.5.0,REV=2.0.18
Architecture mismatch

Package not patched:
PKG=SUNWcar
ARCH=sparc.sun4e
VERSION=11.5.0,REV=2.0.18

Package not patched:
```

```
PKG=SUNWcar
ARCH=sparc.sun4
VERSION=11.5.0,REV=2.0.18
Architecture mismatch
```

These messages indicate an error condition only if `patchadd` does not correctly recognize your architecture.

Message

```
Package not patched:
PKG=SUNxxxx
ARCH=xxxx
VERSION=xxxxxxx
Version mismatch
```

Explanation and Recommended Action

The version of software to which the patch is applied is not installed on your system. For example, if you were running Solaris 8, and you tried to install a patch against Solaris 9, you would see the following (or similar) message:

```
Package not patched:
PKG=SUNWcsu
ARCH=sparc
VERSION=10.0.2
Version mismatch
```

This message does not necessarily indicate an error. If the version mismatch was for a package you needed patched, either get the correct patch version or install the correct package version. Then backout the patch (if necessary) and reapply.

Message

```
Re-installing Patch.
```

Explanation and Recommended Action

The patch has already been applied, but there is at least one package in the patch that could be added. For example, if you applied a patch that had both Openwindows and Answerbook components, but your system did not have Answerbook installed, the Answerbook parts of the patch would not have been applied. If, at a later time, you `pkgadd` Answerbook, you could re-apply the patch, and the Answerbook components of the patch would be applied to the system.

Message

```
patchadd Interrupted.
patchadd is terminating.
```

Explanation and Recommended Action

`patchadd` was interrupted during execution (usually through pressing CTRL-c). `patchadd` will clean up its working files and exit.

Message

```
patchadd Interrupted.
Backing out Patch...
```

Explanation and Recommended Action

`patchadd` was interrupted during execution (usually through pressing CTRL-c). `patchadd` will clean up its working files, backout the patch, and exit.

Message

```
Warning: Cannot open configuration file %s for reading. Using
default serial patching behavior
```

Explanation and Recommended Action

The `/etc/patch/pdo.conf` file is missing from the system. This file is typically created during an initial install or update or by applying the patch for the "Zones Parallel Patching" feature. If the file is not present, the default, one-at-time behavior of adding or removing patches from a zoned system would ensue.

Notes To successfully install a patch to a client or server, `patchadd` must be issued twice, once with the `-R` option and once with the `-S` option. This guarantees that the patch is installed to both the `/usr` and `root` partitions. This is necessary if there are both `/usr` and `root` packages in the patch.

`pkgadd` is invoked by `patchadd` and executes the installation scripts in the *pkg*`/install` directory. The `checkinstall` script is executed with its ownership set to user `install`, if there is no user `install` then `pkgadd` executes the `checkinstall` script as noaccess. The SVR4 ABI states that the `checkinstall` shall only be used as an information gathering script. If the permissions for the `checkinstall` script are changed to something other than the initial settings, `pkgadd` may not be able to open the file for reading, thus causing the patch installation to abort with the following error:

```
pkgadd: ERROR: checkinstall script did not complete successfully.
```

The permission for the `checkinstall` script should not be changed. Contents of log file for a successfull installation: `patchadd` redirects `pkgadd`'s output to the patch installation log file. For a successfull installation, `pkgadd` will produce the following message that gets inserted into the log file:

```
This appears to be an attempt to install the same architecture
and version of a package which is already installed. This
installation will attempt to overwrite this package.
This message does not indicate a failure, it represents the
correct behavior by pkgadd when a patch installs correctly.
```

This message does not indicate a failure, it represents the correct behavior by `pkgadd` when a patch installs correctly.

On client server machines the patch package is *not* applied to existing clients or to the client root template space. Therefore, when appropriate, *all client machines will need the patch applied directly using this same patchadd method on the client.* See instructions above for applying patches to a client. A bug affecting a package utility (for example, `pkgadd`, `pkgrm`, `pkgchk`) could affect the reliability of `patchadd` or `patchrm`, which use package utilities to install and backout the patch package. It is recommended that any patch that fixes package utility problems be reviewed and, if necessary, applied before other patches are applied. Existing patches are:

Solaris 2.5.1 Sparc Platform Edition:
104578

Solaris 2.5.1 Intel Platform Edition:
104579

Solaris 2.6 Sparc Platform Edition:
106292

Solaris 2.6 Intel Platform Edition:
106293

Warnings Certain patches are classified as "deferred activation" patches (sometimes with initial capitals, as "Deferred Activation" patches). Under conditions indicated below, such patches require special treatment. A patch's `README` file specifies whether that patch is of the deferred activation variety. (Search on "Deferred Activation" in the `README` file.)

If you are installing or removing a patch that uses deferred activation patching, you must check on the following:

- On a system running zones, all non-global zones must be in a halted state for adding or removing a patch.
- Deferred activation patching requires the loopback file system (`lofs`) in order to complete safely. Systems running Sun Cluster 3.1 or Sun Cluster 3.2 are likely to have `lofs` turned off because of restrictions on HA-NFS functionality when `lofs` is enabled. Therefore, before a deferred activation patch is installed or removed, you must re-enable the loopback file system by commenting out the following line in the `/etc/system` file:

  ```
  exclude:lofs
  ```

  Then, reboot your system and install or remove the patch. After you have completed the patch operation, uncomment the line cited above, then reboot to resume normal operation.

Name patchrm – remove a Solaris patch package and restore previously saved files

Synopsis

```
patchrm [-f] [-G] [-B backout_dir]
     [-C net_install_image | -R client_root_path | -S service]
     [-t] patch_id
```

Description `patchrm` removes a patch package and restores previously saved files to a system running the Solaris 2.x operating environment or later Solaris environments (such as Solaris 8) that are compatible with Solaris 2.x. `patchrm` cannot be used with Solaris 1 patches. `patchrm` must be run as root.

With respect to `zones(5)`, when invoked in the global zone, by default, `patchrm` patches all appropriate packages in all zones. Patch removal behavior in a `zones` environment varies according to the following factors:

- use of the `-G` option (described below)
- setting of the `SUNW_PKG_ALLZONES` variable in the `pkginfo` file (see `pkginfo(4)`).
- type of zone, global or local (non-global) in `patchrm` which is invoked

The interaction of the factors above is specified in "Interaction of `-G` and `pkginfo` Variable in Zones," below.

When you remove patches from packages on a Solaris system with zones installed, you will see numerous zones-related messages, the frequency and content of which depend on whether you invoke `patchrm` in a global or local zone, the setting of `SUNW_PKG_ALLZONES`, and the use of the `-G` option.

With the "Zones Parallel Patching" feature, patches can be removed from zones in parallel. Using this feature, patches are removed from all zones first and, once they are removed from all zones, removed from the global zone. For this removal to occur, the patch removal software starts a number of processes whose task is to remove patches from zones. The number of processes to be started would be determined by the `num_proc` parameter in the configuration file `/etc/patch/pdo.conf`.

The number of processes to be started is determined in the following order:

1. The value of the `num_proc` parameter. Setting this to 1 retains the current behavior of the patch system.
2. The number of online CPUs in the system.

The upper bound is the number of configured Solaris zones.

Options The following options are supported:

`-B` *backout_dir*
Removes a patch whose backout data has been saved to a directory other than the package database. This option is only needed if the original backout directory, supplied to the `patchadd` command at installation time, has been moved. Specify *backout_dir* as an absolute path name.

`-C` *net_install_image*
Removes the patched files located on the mini root on a Net Install Image created by `setup_install_server`. Specify *net_install_image* as the absolute path name to a Solaris 2.6 or compatible version boot directory. See `EXAMPLES`.

`-f`
Forces the patch removal regardless of whether the patch was superseded by another patch.

`-G`
Remove patch(es) to packages in the current zone only. When used in the global zone, the patch is removed from packages in the global zone only and is not removed from packages in any existing non-global zone. When used in a non-global zone, the patch is removed from packages in the non-global zone only. See "Interaction of `-G` and `pkginfo` Variable in Zones,", below.

`-R` *client_root_path*
Locates all patch files generated by `patchrm` under the directory *client_root_path*. *client_root_path* is the directory that contains the bootable root of a client from the server's perspective. Specify *client_root_path* as the absolute path name to the beginning of the directory tree under which all patch files generated from `patchrm` will be located. `-R` cannot be specified with the `-S` option.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-S` *service*
Specifies an alternate service (for example, Solaris_2.3). This service is part of the server and client model, and can only be used from the server's console. Servers can contain shared `/usr` file systems that are created by `smosservice(1M)`. These service areas can then be made available to the clients they serve. `-S` cannot be specified with the `-R` option.

`-t`
Maintains the `patchrm` return codes from the Solaris release prior to Solaris 10. On a system with `zones(5)` installed, a return code of 0 indicates success. Any other return code indicates failure.

Interaction of `-G` and `pkginfo` Variable in Zones

The following list specifies the interaction between the `-G` option and the `SUNW_PKG_ALLZONES` variable (see `pkginfo(4)`) when removing a patch in global and local (non-global) zones.

global zone, `-G` specified
If *any* packages have `SUNW_PKG_ALLZONES` set to true: Error; nothing changes.

If *no* packages have `SUNW_PKG_ALLZONES` set to true: Remove patch from package(s) in global zone only.

global zone, `-G` not specified
If *any* packages have `SUNW_PKG_ALLZONES` set to true: Remove patch from appropriate package(s) in all zones.

If *no* packages have `SUNW_PKG_ALLZONES` set to true: Remove patch from appropriate package(s) in all zones.

local zone, `-G` specified or not specified
If *any* packages have `SUNW_PKG_ALLZONES` set to true: Error; nothing changes.

If *no* packages have `SUNW_PKG_ALLZONES` set to true: Remove patch from package(s) in local zone only.

Operands

The following operands are supported:

patch_id
The patch number of a given patch. `104945-02` is an example of a *patch_id*.

Examples

The examples in this section assume that patch `104945-02` has been installed to the system prior to removal. All of the examples are relative to the `/usr/sbin` directory.

EXAMPLE 1 Removing a Patch From a Stand-alone System

The following example removes a patch from a standalone system:

```
example# patchrm 104945-02
```

EXAMPLE 2 Removing a Patch From a Client's System From the Server's Console

The following example removes a patch from a client's system from the server's console:

```
example# patchrm -R /export/root/client1 104945-02
```

Note the caveat on the use of the `-R` option in the description of that option, above.

EXAMPLE 3 Removing a Patch From a Server's Service Area

The following example removes a patch from a server's service area:

```
example# patchrm -S Solaris_2.3 104945-02
```

EXAMPLE 4 Removing a Patch From a Net Install Image

The following example removes a patch from a Net Install Image:

```
example# patchrm -C /export/Solaris_2.6/Tools/Boot 104945-02
```

Files

One configuration file of note:

`/etc/patch/pdo.conf` Patch configuration file. Can be used to configure “Zones Parallel Patching” feature.

Exit Status The following exit values are returned:

`0`
Successful completion.

`>0`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWswmt, SUNWcsu

See Also `cpio(1)`, `pkginfo(1)`, `patchadd(1M)`, `pkgadd(1M)`, `pkgchk(1M)`, `pkgrm(1M)`, `showrev(1M)`, `pdo.conf(4)`, `pkginfo(4)`, `attributes(5)`, `zones(5)`

Diagnostics The following messages may help in determining some of the most common problems associated with backing out a patch.

Message

```
prebackout patch exited with return code code.
patchrm exiting.
```

Explanation and Recommended Action
The prebackout script supplied with the patch exited with a return code other than `0`. Generate a script trace of the prebackout script to determine why the prebackout script failed. Add the `-x` option to the first line of the prepatch script to fix the problem and run `patchadd` again.

Message

```
postbackout patch exited with return code code.
patchrm exiting.
```

Explanation and Recommended Action
The postbackout script supplied with the patch exited with a return code other than `0`. Look at the postbackout script to determine why it failed. Add the `-x` option to the first line of the prepatch script to fix the problem, and, if necessary, *re-exececute the postbackout script only*.

Message

```
Only one service may be defined.
```

Explanation and Recommended Action
You have attempted to specify more than one service from which to backout a patch. Different services must have their patches backed out with different invocations of `patchrm`.

Message

```
The -S and -R arguments are mutually exclusive.
```

Explanation and Recommended Action

You have specified both a non-native service and a *client_root_path* from which to backout a patch. These two arguments are mutually exclusive. If backing out a patch from a non-native `usr` partition, the `-S` option should be used. If backing out a patch from a client's root partition (either native or non-native), the `-R` option should be used.

Message

```
The service service cannot be found on this system
```

Explanation and Recommended Action

You have specified a non-native service from which to backout a patch, but the specified service is not installed on your system. Correctly specify the service when backing out the patch.

Message

```
Only one client_root_path may be defined.
```

Explanation and Recommended Action

You have specified more than one *client_root_path* using the `-R` option. The `-R` option may be used only once per invocation of `patchrm`.

Message

```
The dir directory cannot be found on this system.
```

Explanation and Recommended Action

You have specified a directory using the `-R` option which is either not mounted, or does not exist on your system. Verify the directory name and re-backout the patch.

Message

```
Patch patch_id has not been successfully installed to this system.
```

Explanation and Recommended Action

You have attempted to backout a patch that is not installed on this system. If you must restore previous versions of patched files, you may have to restore the original files from the initial installation CD.

Message

```
Patch patch_id has not been successfully applied to this system.
Will remove directory dir.
```

Explanation and Recommended Action

You have attempted to back out a patch that is not applied to this system. While the patch has not been applied, a residual `/var/sadm/patch/`*patch_id* (perhaps from an unsuccessful `patchadd`) directory still exists. The patch cannot be backed out. If you

must restore old versions of the patched files, you may have to restore them from the initial installation CD.

Message

```
This patch was obsoleted by patch patch_id.
Patches must be backed out in the reverse order in
which they were installed. Patch backout aborted.
```

Explanation and Recommended Action

You are attempting to backout patches out of order. Patches should never be backed-out out of sequence. This could undermine the integrity of the more current patch.

Message

```
Patch patch_id is required to be installed by an already
installed patch_id.
It cannot be backed out until the required patch is backed out first.
```

Explanation and Recommended Action

Backout the patch that is required to be installed then backout the desired patch.

Message

```
The installation of patch patch_id was interrupted.
```

Explanation and Recommended Action

A previous installation was interrupted. The interrupted patch needs to be installed before backing out the desired patch.

Message

```
Patch patch_id was installed without backing up the original
files. It cannot be backed out.
```

Explanation and Recommended Action

Either the `-d` option of `patchadd` was set when the patch was applied, or the save area of the patch was deleted to regain space. As a result, the original files are not saved and `patchrm` cannot be used. The original files can only be recovered from the original installation CD.

Message

```
pkgadd of pkgname package failed return code code.
See /var/sadm/patch/patch_id/log for reason for failure.
```

Explanation and Recommended Action

The installation of one of patch packages failed. See the log file for the reason for failure. Correct the problem and run the backout script again.

Message

```
Restore of old files failed.
```

Explanation and Recommended Action
The backout script uses the `cpio` command to restore the previous versions of the files that were patched. The output of the cpio command should have preceded this message. The user should take the appropriate action to correct the `cpio` failure. This is for Solaris 2.4 or previous versions.

Message

```
Illegal character found during parsing. Read the man page
for pdo config file.
```

Explanation and Recommended Action
The `/etc/patch/pdo.conf` follows a specific layout. Each entry in this file should conform to this layout. See `pdo.conf(4)`.

Message

```
Warning: Cannot open configuration file %s for reading. Using
default serial patching behavior
```

Explanation and Recommended Action
The `/etc/patch/pdo.conf` file is missing from the system. This file is typically created during an initial install or update or by applying the patch for the "Zones Parallel Patching" feature. If the file is not present, the default, one-at-time behavior of adding or removing patches from a zoned system would ensue.

Notes On client server machines the patch package is *not* removed from existing clients or from client root template space. Therefore, when appropriate, *all client machines will need the patch removed directly using this same patchrm method on the client.* A bug affecting a package utility (for example, `pkgadd`, `pkgrm`, `pkgchk`) could affect the reliability of `patchadd` or `patchrm` which use package utilities to install and backout the patch package. It is recommended that any patch that fixes package utility problems be reviewed and, if necessary, applied before other patches are applied. Existing patches are:

Solaris 2.1:
patch 100901

Solaris 2.2:
101122

Solaris 2.3:
10133

Solaris 2.4 Sparc Platform Edition:
102039

Solaris 2.4 Intel Platform Edition:
102041

Solaris 2.5.1 Sparc Platform Edition:
104578

Solaris 2.51 Intel Platform Edition:
104579

Solaris 2.6 Sparc Platform Edition:
106292

Solaris 2.6 Intel Platform Edition:
106293

Warnings Certain patches are classified as "deferred activation" patches (sometimes with initial capitals, as "Deferred Activation" patches). Under conditions indicated below, such patches require special treatment. A patch's `README` file specifies whether that patch is of the deferred activation variety. (Search on "Deferred Activation" in the `README` file.)

If you are installing or removing a patch that uses deferred activation patching, you must check on the following:

- On a system running zones, all non-global zones must be in a halted state for adding or removing a patch.
- Deferred activation patching requires the loopback file system (`lofs`) in order to complete safely. Systems running Sun Cluster 3.1 or Sun Cluster 3.2 are likely to have `lofs` turned off because of restrictions on HA-NFS functionality when `lofs` is enabled. Therefore, before a deferred activation patch is installed or removed, you must re-enable the loopback file system by commenting out the following line in the `/etc/system` file:

  ```
  exclude:lofs
  ```

  Then, reboot your system and install or remove the patch. After you have completed the patch operation, uncomment the line cited above, then reboot to resume normal operation.

Name pbind – control and query bindings of processes or LWPs

Synopsis

```
pbind -b processor_id pid [/lwpid]...

pbind [-q] [pid [/lwpid]]...

pbind -Q [processor_id]...

pbind -u pid [/lwpid]...

pbind -U [processor_id]...
```

Description `pbind` controls and queries bindings of processes and LWPs (lightweight processes) to processors. `pbind` can also remove processor bindings that were previously established.

When an LWP is bound to a processor, it will be executed only by that processor except when the LWP requires a resource that is provided only by another processor. The binding is not exclusive, that is, the processor is free to execute other LWPs as well.

Bindings are inherited, so new LWPs and processes created by a bound LWP will have the same binding. Binding an interactive shell to a processor, for example, binds all commands executed by the shell.

Superusers may bind or unbind any process or LWP, while other users can bind or unbind any process or LWP for which they have permission to signal, that is, any process that has the same effective user ID as the user.

Options The following options are supported:

`-b` *processor_id*
: Binds all or a subset of the LWPs of the specified processes to the processor *processor_id*. Specify *processor_id* as the processor ID of the processor to be controlled or queried. *processor_id* must be present and on-line. Use the `psrinfo` command to determine whether or not *processor_id* is present and on-line. See `psrinfo(1M)`.

`-q`
: Displays the bindings of the specified processes or of all processes. If a process is composed of multiple LWPs which have different bindings and the LWPs are not explicitly specified, the bindings of only one of the bound LWPs will be displayed. The bindings of a subset of LWPs can be displayed by appending "/lwpids" to the process IDs. Multiple LWPs may be selected using "-" and "," delimiters. See EXAMPLES.

`-Q`
: Displays the LWPs bound to the specified list of processors, or all LWPs with processor bindings. For processes composed of multiple LWPs, the bindings of individual LWPs will be displayed.

`-u`
: Removes the bindings of all or a subset of the LWPs of the specified processes, allowing them to be executed on any on-line processor.

`-U`
Removes the bindings of all LWPs bound to the specified list of processors, or to any processor if no argument is specified.

Operands The following operands are supported:

pid
The process ID of the process to be controlled or queried.

lwpid
The set of LWP IDs of the specified process to be controlled or queried. The syntax for selecting LWP IDs is as follows:

```
2,3,4-8       LWP IDs 2, 3, and 4 through 8
-4            LWPs whose IDs are 4 or below
4-            LWPs whose IDs are 4 or above
```

processor_id
The processor ID of the processor to be controlled or queried.

Examples **EXAMPLE 1** Binding Processes

The following example binds processes 204 and 223 to processor 2:

```
example% pbind -b 2 204 223
process id 204: was 2, now 2
process id 223: was 3, now 2
```

EXAMPLE 2 Unbinding a Process

The following example unbinds process 204:

```
example% pbind -u 204
```

EXAMPLE 3 Querying Bindings

The following example queries bindings. It demonstrates that process 1 is bound to processor 0, process 149 has at least one LWP bound to CPU3, and process 101 has no bound LWPs.

```
example% pbind -q 1 149 101
process id 1: 0
process id 149: 3
process id 101: not bound
```

EXAMPLE 4 Querying LWP Bindings

The following example queries bindings of LWPs. It demonstrates that LWP 1 of process 149 is bound to CPU3, and LWP 2 of process 149 is not bound.

```
example% pbind -q 149/1-2
lwp id 149/1: 3
lwp id 149/2: not bound
```

EXAMPLE 5 Querying LWP Bindings for Processor 2:

The following example queries all LWPs bound to processor 2:

```
example% pbind -Q 2
lwp id 149/4: 2
lwp id 149/5: 2
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

Exit Status

The following exit values are returned:

`0`
Successful completion.

`>0`
An error occurred.

See Also

`psradm(1M)`, `psrinfo(1M)`, `psrset(1M)`, `processor_bind(2)`, `processor_info(2)`, `sysconf(3C)`, `attributes(5)`

Diagnostics

`pbind: cannot query pid 31: No such process`
The process specified did not exist or has exited.

`pbind: cannot bind pid 31: Not owner`
The user does not have permission to bind the process.

`pbind: cannot bind pid 31: Invalid argument`
The specified processor is not on-line.

Name pcmciad – PCMCIA user daemon

Synopsis

```
/usr/lib/pcmciad
```

Description The PCMCIA user daemon provides user-level services for the PCMCIA nexus driver and PCMCIA card client drivers. There are no user-configurable options for this daemon.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcmcu

See Also pcmcia(7D), attributes(5)

Diagnostics

```
pcmciad: can't open /dev/pem: No such file or directory
```

The user daemon could not communicate with the PCMCIA event management driver.

Name pfinstall – tests installation profiles

Synopsis

```
/usr/sbin/install.d/pfinstall -D | -d disk_config
     [-c CDpath] profile
```

Description After you create a profile, you can use the `pfinstall` command to test the profile and see if it does what you want before using it to install or upgrade a system. `pfinstall` enables you to test a profile against:

- The system's disk configuration where `pfinstall` is being run.
- Other disks by using a disk configuration file that represents a structure of a disk. See `NOTES` on how to create a disk configuration file.

To successfully and accurately test a profile for a particular Solaris release, you must test a profile within the Solaris environment of the same release. For example, if you want to test a profile for Solaris 2.10, you have to run the `pfinstall` command on a system running Solaris 2.10.

So, on a system running Solaris 2.10, you can test Solaris 2.10 initial installation profiles. However, if you want to test a Solaris 2.10 upgrade profile on a system running a previous version of Solaris, or if you don't have a Solaris 2.10 system installed yet to test Solaris 2.10 initial installation profiles, you have to boot a system from a Solaris 2.10 CD image and temporarily create a Solaris 2.10 install environment. Then, you can run `pfinstall` in the Solaris 2.10 install environment to test your profiles.

To create a temporary Solaris operating system install environment, boot a system from a Solaris CD image (just as you would to install), answer any system identification questions, choose the Solaris Interactive Installation program, and exit out of the first screen that is presented. Then, from the shell, you can execute the `pfinstall` command.

Options The following options are supported:

`-c` *CDpath* — The path to the Solaris installation image. This is required if the image is not mounted on `/cdrom`. (For example, use this option if you copied the installation image to disk or mounted the CD-ROM on a directory other than `/cdrom`.) When testing a profile on an x86 machine, the miniroot needs to be unpacked before using `-c` option. See EXAMPLES, below, for the procedure to unpack the miniroot.

`-d` *disk_config* — `pfinstall` uses a disk configuration file, *disk_config*, to test the profile. See `NOTES` on how to create a disk configuration file. You must specify either this option or the `-D` option to test the profile (see `WARNINGS`). This option cannot be used with an upgrade profile (`install_type upgrade`). You must always test an upgrade profile against a system's disk configuration (`-D` option).

`-D` — `pfinstall` uses the system's disk configuration to test the profile. You must specify either this option or the `-d` option to test the profile (see `WARNINGS`).

Operands The following operands are supported:

profile The file name of the profile to test. If *profile* is not in the directory where `pfinstall` is being run, you must specify the path.

Examples **EXAMPLE 1** Testing an Upgrade Profile

The following example tests an upgrade profile, `upgrade.prof`, on a system with a previous version of the Solaris software installed.

1. Boot the system to be upgraded from the Solaris image chosen for the upgrade, just as you would to install. The image can be located in the system's local CD-ROM or on an install server.
2. Answer the system configuration questions, if prompted.
3. If you are presented with a choice of installation options, choose the Solaris Interactive Installation program.
4. Exit from the first screen of the Solaris Interactive Installation program.

   After the Solaris Interactive Installation program exits, a shell prompt is displayed.
5. Create a temporary mount point:

   ```
   example# mkdir /tmp/mnt
   ```
6. Mount the directory that contains the profile(s) you want to test.

   If you want to mount a remote NFS file system (for systems on the network), enter:

   ```
   example# mount -F nfs server_name:path /tmp/mnt
   ```

   If you want to mount a UFS-formatted diskette, enter:

   ```
   example# mount -F ufs /dev/diskette /tmp/mnt
   ```

   If you want to mount a PCFS-formatted diskette, enter:

   ```
   example# mount -F pcfs /dev/diskette /tmp/mnt
   ```
7. Change directory to `/tmp/mnt` where the profile resides:

   ```
   example# cd /tmp/mnt
   ```
8. Test the `upgrade.prof` profile:

   ```
   example# /usr/sbin/install.d/pfinstall -D upgrade.prof
   ```

EXAMPLE 2 Testing the `basic.prof` Profile on an x86 Machine (Includes Unpacking miniroot)

The following example tests the `basic.prof` profile against the disk configuration on a Solaris 8 system where `pfinstall` is being run. The path to the Solaris CD image is specified because Volume Management is being used.

```
example# /usr/sbin/install.d/pfinstall -D -c /cdrom/cdrom0/s0 basic.prof
```

EXAMPLE 2 Testing the `basic.prof` Profile on an x86 Machine (Includes Unpacking miniroot) *(Continued)*

When testing a profile on an x86 machine, before using `pfinstall` with `-c` option, use the following procedure to unpack the miniroot:

1. Uncompress the miniroot archive into a temporary file. The miniroot archive is under `/cdrom/cdrom0/s0`.

   ```
   # /usr/bin/gzcat /cdrom/cdrom0/boot/x86.miniroot > \
   /tmp/x86_miniroot_gz
   ```

2. Create the miniroot device using `lofiadm(1M)`:

   ```
   /usr/sbin/lofiadm -a /tmp/x86_miniroot_gz
   /dev/lofi/1
   ```

3. Mount the miniroot under miniroot directory using the `lofi` device returned by `lofiadm` in the previous step:

   ```
   # /usr/sbin/mount -F ufs /dev/lofi/1 \
   /cdrom/cdrom0/Solaris_10/Tools/Boot
   ```

4. Now run `pfinstall` with `-D` option:

   ```
   # /usr/sbin/install.d/pfinstall -D -c /cdrom/cdrom0/ jumpstart.profile
   ```

5. After `pfinstall` testing is completed, unmount the `lofi` device:

   ```
   # /usr/sbin/umount /dev/lofi/1
   ```

6. Delete the `lofi` device:

   ```
   # /usr/sbin/lofiadm -d /tmp/x86_miniroot_gz
   ```

EXAMPLE 3 Testing the `basic.prof` Profile Against Disk Config File

The following example tests the `basic.prof` profile against the `535_test` disk configuration file. This example uses a Solaris CD image located in the `/export/install` directory, and `pfinstall` is being run on a Solaris 2.6 system.

```
example# /usr/sbin/install.d/pfinstall -d 535_test \
     -c /export/install basic.prof
```

Exit Status

`0` Successful (system rebooted).

`1` Successful (system not rebooted).

`2` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWinst

See Also fdisk(1M), lofiadm(1M), prtvtoc(1M), attributes(5)

Solaris 10 Installation Guide: Basic Installations

Warnings If the -d or -D option is not specified, pfinstall may perform an actual installation on the system by using the specified profile, and the data on the system may be overwritten.

Notes You have to test a profile on a system with the same platform type for which the profile was created.

SPARC To create a disk configuration file (-d option) for a SPARC based system:

1. Locate a SPARC based system with a disk that you want to test.
2. Create a disk configuration file by redirecting the output of the prtvtoc(1M) command to a file.

   ```
   example# prtvtoc /dev/rdsk/c0t3d0s2 > 535_disk
   ```

3. (Optional.) Concatenate disk configuration files into a single file to test a profile against multiple disks. The target numbers in the disk device names must be unique.

   ```
   example# cat 535_disk 1G_disk > mult_disks
   ```

x86 To create a disk configuration file (-d option) for an x86 based system:

1. Locate an x86 based system with a disk that you want to test.
2. Create part of the disk configuration file by saving the output of the fdisk(1M) command to a file:

   ```
   example# fdisk -R -W 535_disk /dev/rdsk/c0t3d0p0
   ```

3. Append the output of the prtvtoc(1M) command to the disk configuration file.

   ```
   example# prtvtoc /dev/rdsk/c0t3d0s2 >> 535_disk
   ```

4. (Optional.) Concatenate disk configuration files into a single file to test a profile against multiple disks. The target numbers in the disk device names must be unique.

   ```
   example# cat 535_disk 1G_disk > mult_disks
   ```

To test a profile with a specific system memory size, set SYS_MEMSIZE to the specific memory size (in Mbytes) before running pfinstall:

```
example# SYS_MEMSIZE=memory_size

example# export SYS_MEMSIZE
```

Name pgxconfig, GFXconfig, TSIgfxp_config – configure the PGX32 (Raptor GFX) Graphics Accelerator

Synopsis

```
/usr/sbin/pgxconfig [-dev device-filename]
     [-res video-mode [try | noconfirm | nocheck]]
     [-file machine | system] [-depth 8 | 24]
     [-24only true | false] [-cachedpixmap true | false]
     [-defaults]

/usr/sbin/pgxconfig [-propt] [-prconf]

/usr/sbin/pgxconfig [-help] [-res ?]

/usr/sbin/pgxconfig [-i]
```

Description The `pgxconfig` utility configures the PGX32 (Raptor GFX) Graphics Accelerator and some of the X11 window system defaults for PGX32 (Raptor GFX). A previous version of this utility was named `GFXconfig`.

The first form of `pgxconfig` shown in the synopsis above stores the specified options in the OWconfig file. These options are used to initialize the PGX32 (Raptor GFX) device the next time the window system is run on that device. Updating options in the OWconfig file provides persistence of these options across window system sessions and system reboots.

The second, third, and fourth forms, which invoke only the `-prconf`, `-propt`, `-help`, and `-res ?` options, do not update the OWconfig file. For the third form all other options are ignored.

The `-i` option starts `pgxconfig` in interactive mode.

Options may be specified for only one PGX32 (Raptor GFX) device at a time.

Only PGX32 (Raptor GFX)-specific options can be specified through `pgxconfig`. The normal window system options for specifying default depth, default visual class and so forth are still specified as device modifiers on the `openwin` command line. See the `Xsun`(1) manual page available with the SUNWxwman package.

The user can also specify the OWconfig file that is to be updated. By default, the machine-specific file in the `/usr/openwin` directory tree is updated. The `-file` option can be used to specify an alternate file to use. For example, the system-global OWconfig file in the `/etc/openwin` directory tree can be updated instead.

Both of these standard OWconfig files can only be written by root.

Options The following options are supported:

`-cachedpixmap` true | false
: When set to `false`, it forces the PGX32 (Raptor GFX) device to use 24–bit only when running OpenWindows. The default value is `true`.

Certain applications make use of a cached pixmap when writing to the display device. Such a technique can cause garbled output and can cause the X server to crash. If you experience such behavior, try setting the `-cachedpixmap` option to `false`.

`-defaults` — Reset all option values to their default values.

`-depth` 8 | 24 — Sets the screen depth to 8 or 24 bits per pixel. 24 bits per pixel enables TrueColor graphics in the window system.

`-dev` *device-filename* — Specify the PGX32 (Raptor GFX) special file. The default is `/dev/fbs/gfxp0`, or `/dev/fbs/raptor0` if applicable.

`-file machine| system` — Specifies which `OWconfig` file to update. If *machine* is specified, the machine-specific `OWconfig` file in the `/etc/openwin` directory tree is updated. If *system* is specified, the global `OWconfig` file in the `/usr/openwin` directory tree is updated. If the specified file does not exist, it is created. This option has no effect unless other options are specified. The default is *machine*.

`-help` — Print a list of the `pgxconfig` command line options, along with a brief explanation of each.

`-i` — Start `pgxconfig` in interactive mode.

`-prconf` — Print the PGX32 (Raptor GFX) hardware configuration. Thie following is a typical display:

```
--- Hardware Configuration for /dev/fbs/gfxp0 ---
DAC: version 0x0
Type:
Board:
PROM: version 0x0
PROM Information:
RAM:
EDID Data:
Monitor Sense ID:
```

```
Card possible resolutions:  640x480x60, 800x600x75, 1024x768x60
               1024x768x70, 1024x768x75, 1280x1024x75, 1280x1024x76
               1280x1024x60, 1152x900x66, 1152x900x76, 1280x1024x67
               960x680x112S, 960x680x108S, 640x480x60i, 768x575x50i,
               1280x800x76, 1440x900x76, 1600x1000x66, 1600x1000x76,
               vga, svga, 1152, 1280, stereo, ntsc, pal
Monitor possible resolutions:  720x400x70, 720x400x88, 640x480x60
               640x480x67, 640x480x72, 640x480x75, 800x600x56,
               800x600x60, 800x600x72, 800x600x75, 832x624x75,
               1024x768x87, 1024x768x60, 1024x768x70, 1024x768x75,
               1280x1024x75, 1280x1024x76, 1152x900x66, 1152x900x76,
               1280x1024x67, 960x680x112S, vga, svga, 1152, 1280
               stereo
Current resolution setting: 1280x1024x76
Possible depths: 8, 24, 8+24
Current depth: 8
```

`-propt`

Print the current values of all PGX32 (Raptor GFX) options in the `OWconfig` file specified by the `-file` option for the device specified by the `-dev` option. Print the values of options as they would be in the `OWconfig` file after the call to `pgxconfig` would have completed. The following is a typical display:

```
--- OpenWindows Configuration for /dev/fbs/gfxp0 ---
OWconfig: machine
Video Mode: not set
Depth: 8+24
```

`-res` *video-mode* [`try` | `noconfirm` | `nocheck`]

Specify the built-in video mode used to drive the monitor connected to the specified PGX32 (Raptor GFX) device.

The format for *video-mode* can be one of the following:

*width*x*height*x*rate*

The *width* is the screen width in pixels, *height* is the screen height in pixels, and *rate* is the vertical frequency of the screen refresh. As a convenience, `-res`

also accepts formats with @ prepended to the refresh rate rather than `x`. For example: 1280x1024@76. The list can be obtained by running `pgxconfig` with the `-res ?` option (the third form shown in the command synopsis above). Note that not all resolutions are supported by both the video board and by the monitor. The `pgxconfig` utility will not permit you to set a resolution not supported by the board unless the `noconfirm` or `nocheck` option is specified. It will also request confirmation before setting a resolution not supported by the monitor if the `nocheck` option is not specified.

Symbolic names

For convenience, the video modes listed below have symbolic names defined. Rather than the form *width*x*height*x*rate*, the symbolic name may be supplied as the argument to

`-res`. If the symbolic name is `none`, the screen resolution will be the video mode that is currently programmed in the device when the window system is run.

`svga`	1024x768x60
`1152`	1152x900x76
`1280`	1280x1024x76
`vga`	640x480x60
`none`	default console resolution

The `-res` option also accepts additional, optional arguments immediately following the video mode specification. Any or all of these may be present.

`noconfirm` Using the `-res` option, the user could put the system into an unusable state, a state where there is no video output. This can happen if there is ambiguity in the monitor sense codes for the particular code read. To reduce the chance of this occurring, the default behavior of `pgxconfig` is to print a warning message to this effect and to prompt the user to find out if it is okay to continue. The `noconfirm` option instructs `pgxconfig` to bypass this confirmation and to

program the requested video mode anyway. This option is useful when `pgxconfig` is being run from a shell script.

`nocheck` — If present, normal error checking based on the monitor sense code is suspended. The video mode specified by the user will be accepted regardless of whether it is appropriate for the currently attached monitor. (This option is useful if a different monitor is to be connected to the PGX32 (Raptor GFX) device). Use of this option implies `noconfirm` as well.

`try` — This option allows the user to test the specified resolution before committing it. It displays a pattern on the screen with the specified resolution. If the test pattern appears correctly, the user may answer "y" to the query. The other permissable answer is "n".

This sub-option should not be used with `pgxconfig` while the configured device is being used (for example, while running the window system) as unpredictable results may occur. To run `pgxconfig` with the `try` sub-option, the window system should be brought down first.

`-res ?`	Print the list of possible resolutions supported by the PGX32 and the monitor.
`-24only`	Force the PGX32 (Raptor GFX) device to use 24 bit only when running Openwindows.

Defaults For a given invocation of `pgxconfig`, if an option does not appear on the command line, the corresponding OWconfig option is not updated; it retains its previous value, except for `-depth` and `-24only`.

A default value is used if a PGX32 (Raptor GFX) option has not been specified with `pgxconfig` when the window system is run. The option defaults are as follows:

`-dev` `/dev/fbs/gfxp0`

`-file` `system`

`-res` none

The default of none for the `-res` option indicates that when the window system is run, the screen resolution will be the video mode that is currently programmed in the device.

Examples **EXAMPLE 1** Switching the Resolution on the Monitor Type

The following example switches the monitor type to the resolution of `1280` x `1024` at `76` Hz:

```
example# /usr/sbin/pgxconfig -res 1280x1024x76
```

Files

`/dev/fbs/gfxp0`	device special file
`/usr/openwin/server/etc/OWconfig`	system configuration file
`/etc/openwin/server/etc/OWconfig`	machine configuration file

See Also *PGX32 Installation Manual*

Name picld – PICL daemon

Synopsis `/usr/lib/picl/picld`

Description The Platform Information and Control Library (PICL) provides a mechanism to publish platform-specific information for clients to access in a platform-independent way. `picld` maintains and controls access to the PICL information from clients and plug-in modules. The daemon is started in both single-user and multi-user boot mode.

Upon startup, the PICL daemon loads and initializes the plug-in modules. These modules use the `libpicltree(3PICLTREE)` interface to create nodes and properties in the PICL tree to publish platform configuration information. After the plug-in modules are initialized, the daemon opens the PICL daemon door to service client requests to access information in the PICL tree.

PICL Tree The PICL tree is the repository of all the nodes and properties created by the plug-in modules to represent the platform configuration. Every node in the PICL tree is an instance of a well-defined PICL class. The name of the base PICL class is `picl`, which defines a basic set of properties that all nodes in the tree must possess. Two of those properties are `name` and `_class`, where `name` contains the name of the node, and the `_class` contains the PICL class name of the node. Certain nodes in the PICL tree have well-known names. For example, the name of the root node of the PICL tree is `/` and the name of the root node of the sub-tree containing platform device nodes is `platform`.

PICL plug–in Modules The PICL plug-in modules are shared objects that publish platform-specific data in the PICL tree. They are located in well-known directories so that the daemon can locate and load them.

Plug-in modules are located in one of the following plug-in directories depending on the plaform-specific nature of the data that they collect and publish:

```
/usr/platform/‘uname -i‘/lib/picl/plugins
/usr/platform/‘uname -m‘/lib/picl/plugins
```

A plug-in module can specify its dependency on another plug-in module using the `-l` or `-R` linker option. The plug-ins are loaded by the daemon using `dlopen(3C)` according to the specified dependencies. Each plug-in module must define a `.init` section, which is executed when the plug-in module is loaded, to register themselves with the daemon. See `picld_plugin_register(3PICLTREE)` for additional information on plug-in registration.

The plug-in modules use the `libpicltree(3PICLTREE)` interface to publish nodes and properties in the PICL tree so that clients can access them.

When the PICL daemon invokes the initialization routine of the plug-in module, the plug-in collects the platform information and creates nodes and/or properties to represent the configuration in the PICL tree. A plug-in can create additional threads to monitor the platform configuration and update the PICL tree with any changes. This enables a PICL plug-in to operate as a daemon within the PICL framework.

An environmental monitor is an example of a plug-in module that uses a thread to monitor the temperatures and fan speeds of the platform, then publishes the environmental information in the PICL tree so clients can access them.

Clients use the libpicl(3PICL) interface to send requests to picld for accessing the PICL tree.

Exit Status picld does not return an exit status.

Files

/var/run/picld_door	PICL daemon door
/usr/lib/picl/picld	PICL daemon

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpiclu

See Also svcs(1), svcadm(1M), dlopen(3C), libpicl(3PICL), libpicltree(3PICLTREE), picld_log(3PICLTREE), picld_plugin_register(3PICLTREE), attributes(5), smf(5)

Notes The picld service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/picl
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name ping – send ICMP (ICMP6) ECHO_REQUEST packets to network hosts

Synopsis

```
/usr/sbin/ping  host [timeout]

/usr/sbin/ping  -s [-l |  -U] [-adlLnrRv] [-A addr_family]
 [-c traffic_class] [-g gateway  [-g  gateway...]] [-F flow_label]
 [-I interval] [-i interface] [-P tos] [-p port] [-t ttl]
 host [data_size] [npackets]
```

Description The utility `ping` utilizes the ICMP (ICMP6 in IPv6) protocol's `ECHO_REQUEST` datagram to elicit an ICMP (ICMP6) `ECHO_RESPONSE` from the specified *host* or network *gateway*. If *host* responds, `ping` will print:

```
host is alive
```

on the standard output and exit. Otherwise, after *timeout* seconds, it will write:

```
no answer from host
```

The default value of *timeout* is `20` seconds.

When you specify the s flag, sends one datagram per second (adjust with `-I`) and prints one line of output for every `ECHO_RESPONSE` that it receives. `ping` produces no output if there is no response. In this second form, `ping` computes round trip times and packet loss statistics; it displays a summary of this information upon termination or timeout. The default *data_size* is `56` bytes, or you can specify a size with the *data_size* command-line argument. If you specify the optional *npackets*, `ping` sends `ping` requests until it either sends *npackets* requests or receives *npackets* replies.

When using `ping` for fault isolation, first `ping` the local host to verify that the local network interface is running.

Options The following options are supported:

`-A` *addr_family*

Specify the address family of the target host. *addr_family* can be either `inet` or `inet6`. Address family determines which protocol to use. For an argument of `inet`, IPv4 is used. For `inet6`, IPv6 is used.

By default, if the name of a host is provided, not the literal IP address, and a valid IPv6 address exists in the name service database, `ping` will use this address. Otherwise, if the name service database contains an IPv4 address, it will try the IPv4 address.

Specify the address family `inet` or `inet6` to override the default behavior. If the argument specified is `inet`, `ping` will use the IPv4 address associated with the host name. If none

exists, `ping` will state that the host is unknown and exit. It does not try to determine if an IPv6 address exists in the name service database.

If the specified argument is `inet6`, `ping` uses the IPv6 address that is associated with the host name. If none exists, `ping` states that the host is unknown and exits.

`-F` *flow_label*
: Specify the flow label of probe packets. The value must be an integer in the range from 0 to 1048575. This option is valid only on IPv6.

`-I` *interval*
: Turn on the statistics mode and specify the interval between successive transmissions. The default is one second. See the discussion of the `-s` option.

`-L`
: Turn off loopback of multicast packets. Normally, members are in the host group on the outgoing interface, a copy of the multicast packets will be delivered to the local machine.

`-P` *tos*
: Set the type of service (*tos*) in probe packets to the specified value. The default is zero. The value must be an integer in the range from 0 to 255. Gateways also in the path can route the probe packet differently, depending upon the value of *tos* that is set in the probe packet. This option is valid only on IPv4.

`-R`
: Record route. Sets the IPv4 record route option, which stores the route of the packet inside the IPv4 header. The contents of the record route are only printed if the `-v` and `-s` options are given. They are only set on return packets if the target host preserves the record route option across echos, or the `-l` option is given. This option is valid only on IPv4.

`-U`
: Send UDP packets instead of ICMP (ICMP6) packets. `ping` sends UDP packets to consecutive ports expecting to receive back ICMP (ICMP6) `PORT_UNREACHABLE` from the target host.

`-a`
: `ping` all addresses, both IPv4 and IPv6, of the multihomed destination. The output appears as if `ping` has been run once for each IP address of the destination. If this option is used together with `-A`, `ping` probes only the addresses that are of the specified address family. When used with the `-s` option and *npackets* is not specified, `ping` continuously probes the destination addresses in a round robin fashion. If *npackets* is

specified, `ping` sends *npackets* number of probes to each IP address of the destination and then exits.

`-c` *traffic_class*
: Specify the traffic class of probe packets. The value must be an integer in the range from 0 to 255. Gateways along the path can route the probe packet differently, depending upon the value of *traffic_class* set in the probe packet. This option is valid only on IPv6.

`-d`
: Set the `SO_DEBUG` socket option.

`-g` *gateway*
: Specify a loose source route gateway so that the probe packet goes through the specified host along the path to the target host. The maximum number of gateways is 8 for IPv4 and 127 for IPv6. Note that some factors such as the link MTU can further limit the number of gateways for IPv6.

`-i` *interface_address*
: Specify the outgoing interface address to use for multicast packets for IPv4 and both multicast and unicast packets for IPv6. The default interface address for multicast packets is determined from the (unicast) routing tables. *interface_address* can be a literal IP address, for example, `10.123.100.99`, or an interface name, for example, `eri0`, or an interface index, for example `2`.

`-l`
: Use to send the probe packet to the given host and back again using loose source routing. Usually specified with the `-R` option. If any gateways are specified using `-g`, they are visited twice, both to and from the destination. This option is ignored if the `-U` option is used.

`-n`
: Show network addresses as numbers. `ping` normally does a reverse name lookup on the IP addresses it extracts from the packets received. The `-n` option blocks the reverse lookup, so `ping` prints IP addresses instead of host names.

`-p` *port*
: Set the base UDP *port* number used in probes. This option is used with the `-U` option. The default base *port* number is 33434. The `ping` utility starts setting the destination port number of UDP packets to this base and increments it by one at each probe.

`-r`
: Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly attached network, an error is returned. This option can be used to `ping` a local host through an interface that has been dropped by the router daemon. See `in.routed(1M)`.

`-s`	Send one datagram per second and collect statistics.
`-t` *ttl*	Specify the IPv4 time to live, or IPv6 hop limit, for unicast and multicast packets. The default time to live (hop limit) for unicast packets can be set with the `ndd` module, `/dev/icmp`, using the `icmp_ipv4_ttl` variable for IPv4 and the `icmp_ipv6_hoplimit` variable for IPv6. The default time to live (hop limit) for multicast is one hop. See `EXAMPLES`. For further information, see`ndd(1M)`.
`-v`	Verbose output. List any ICMP (ICMP6) packets, other than replies from the target host.

Operands *host* The network host

Examples **EXAMPLE 1** Using `ping` With IPv6

This example shows `ping` sending probe packets to all the IPv6 addresses of the host `xyz`, one at a time. It sends an ICMP6 `ECHO_REQUEST` every second until the user interrupts it.

```
istanbul% ping -s -A inet6 -a xyz
PING xyz: 56 data bytes
64 bytes from xyz (4::114:a00:20ff:ab3d:83ed): icmp_seq=0. time=0.479 ms
64 bytes from xyz (fec0::114:a00:20ff:ab3d:83ed): icmp_seq=1. time=0.843 ms
64 bytes from xyz (4::114:a00:20ff:ab3d:83ed): icmp_seq=2. time=0.516 ms
64 bytes from xyz (fec0::114:a00:20ff:ab3d:83ed): icmp_seq=3. time=4.94 ms
64 bytes from xyz (4::114:a00:20ff:ab3d:83ed): icmp_seq=4. time=0.485 ms
64 bytes from xyz (fec0::114:a00:20ff:ab3d:83ed): icmp_seq=5. time=2.20 ms
^C
----xyz PING Statistics----
6 packets transmitted, 6 packets received, 0% packet loss
round-trip (ms)  min/avg/stddev = 0.479/1.58/4.94/1.8
```

EXAMPLE 2 Using `ndd` to Set the `icmp_ipv6_hoplimit`

This example shows the `ndd` module, `/dev/icmp`, used to set the `icmp_ipv6_hoplimit`.

```
# ndd -set /dev/icmp icmp_ipv6_hoplimit 100
```

Exit Status The following exit values are returned:

`0`	Successful operation; the machine is alive.
non-zero	An error has occurred. Either a malformed argument has been specified, or the machine was not alive.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbip

See Also ifconfig(1M), in.routed(1M), ndd(1M), netstat(1M), rpcinfo(1M), traceroute(1M), attributes(5), icmp(7P), icmp6(7P)

Name pkg2du – convert driver packages to Driver Update format

Synopsis

```
/usr/bin/pkg2du [-f] [-v] [-d dir] [-o iso] [-l label]
     [-r release] pkg [pkg ...]
```

Description The `/usr/bin/pkg2du` utility takes one or more packages as input and converts them to Driver Update (DU) format. If the `-d` option is specified, the resulting DU directory tree is placed in the directory *dir*. If the `-o` option is specified, a Solaris ISO image of the DU directory tree is written in the file *iso*. The ISO image can be burned onto CD/DVD using `cdrw(1)` or `cdrecord(1)` (not a SunOS man page) and used during Solaris installation.

At least one of the `-d` and `-o` options must be specified. If both are specified, then both an ISO image and a directory tree are generated.

Options The following options are supported:

`-d` *dir*
Directory where the DU directory should be created.

`-o` *iso*
Create a Solaris ISO image of the DU directory.

`-f`
If *dir*`/DU` or *iso* exists, remove it without asking first.

`-l` *label*
Label/volume name of the ISO image (if `-o` option is specified).

`-r` *release*
Solaris release number to use. It takes the form of the return from `uname -r` command, for example, `5.10`. If unspecified, the release number of the currently running Solaris is used.

`-v`
Verbose. Multiple `-v` options increase verbosity.

Operands The following operands are supported:

pkg [*pkg*...]
One or more packages to be converted to DU format.

Examples **EXAMPLE 1** Creating a DU CD/DVD

The following commands create a DU CD or DVD containing packages `SUNWfoo` and `SUNWbar`.

```
# /usr/bin/pkg2du -r 5.10 -o my.iso SUNWfoo SUNWbar
# /usr/bin/cdrw -i my.iso
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed

See Also cdrw(1), mkbootmedia(1M), attributes(5)

mkisofs(8), (/usr/share/man/man8/mkisofs.8), in the SUNWfsman package (not a SunOS man page)

Name pkgadd – transfer software packages to the system

Synopsis

```
pkgadd [-nv] [-a admin] [-G] [-x proxy]
     [ [-M] -R root_path] [-r response] [-k keystore]
     [-P passwd] [-V fs_file]
     [-d device | -d datastream pkginst | all]
     [pkginst | -Y category [,category]...]

pkgadd -s [-d device | -d datastream pkginst | all]
     [pkginst | -Y category [,category]...]
```

Description `pkgadd` transfers the contents of a software package from the distribution medium or directory to install it onto the system. Used without the `-d` *device* source specifier, `pkgadd` looks in the default spool directory (`/var/spool/pkg`) for the package. Used with the `-s` option, it writes the package to a spool directory instead of installing it.

The `pkgadd` utility requires an amount of temporary space the size of the package that is being installed. `pkgadd` determines which temporary directory to use by checking for the existance of the `$TMPDIR` environment variable. If `$TMPDIR` is not defined, `pkgadd` uses `P_tmpdir` from `stdio.h`. `P_tmpdir` has a default of `/var/tmp/`.

Certain unbundled and third-party packages are no longer entirely compatible with the latest version of `pkgadd`. These packages require user interaction throughout the installation and not just at the very beginning, or require that their request scripts be run as the root user.

To install these older packages (released prior to Solaris 2.4), set the following environment variable: `NONABI_SCRIPTS=TRUE`

As long as this environment variable is set, `pkgadd` permits keyboard interaction throughout the installation and package request scripts are run as `root`.

If you have package request scripts that require running as user `root` (instead of `noaccess` [the default] or user `install`), use the `rscript_alt` parameter in the `admin(4)` file to make an appropriate selection. See `admin(4)`.

Note that, in Solaris 8 and Solaris 9, the default user when running a request script was either `root` or `nobody`, depending on the operating system's patch level. In the current release, the default user is `noaccess`.

When running `pkgadd` in the global zone (see `zones(5)`), a package that contains a request script (see `pkgask(1M)`) is added only to the global zone. The package is not propagated to any current or yet-to-be-installed non-global zone. This behavior mimics the effect of the `-G` option, described below.

Package commands are `largefile(5)`-aware. They handle files larger than 2 GB in the same way they handle smaller files. In their current implementations, `pkgadd`, `pkgtrans(1)` and other package commands can process a datastream of up to 4 GB.

The `-d`, `-Y`, and *pkginst* arguments shown in the SYNOPSIS are described under OPERANDS, following OPTIONS.

Options The supported options are described as follows. The `-d` *device* source specifier is described under OPERANDS, below.

`-a` *admin*
Define an installation administration file, *admin*, to be used in place of the default administration file. The token `none` overrides the use of any *admin* file, and thus forces interaction with the user. Unless a full path name is given, `pkgadd` first looks in the current working directory for the administration file. If the specified administration file is not in the current working directory, `pkgadd` looks in the `/var/sadm/install/admin` directory for the administration file.

`-G`
Add package(s) in the current zone only. When used in the global zone, the package is added to the global zone only and is not propagated to any existing or yet-to-be-created non-global zone. When used in a non-global zone, the package(s) are added to the non-global zone only.

This option causes package installation to fail if, in the `pkginfo` file for a package, `SUNW_PKG_ALLZONES` is set to true. See `pkginfo(4)`.

`-k` *keystore*
Use *keystore* as the location from which to get trusted certificate authority certificates when verifying digital signatures found in packages. If no keystore is specified, then the default keystore locations are searched for valid trusted certificates. See `KEYSTORE LOCATIONS` for more information.

`-M`
Instruct `pkgadd` not to use the `$`*root_path*`/etc/vfstab` file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

`-n`
Installation occurs in non-interactive mode. Suppress output of the list of installed files. The default mode is interactive.

`-P` *passwd*
Password to use to decrypt keystore specified with `-k`, if required. See `PASS PHRASE ARGUMENTS` for more information about the format of this option's argument.

`-r` *response*
Identify a file or directory which contains output from a previous `pkgask(1M)` session. This file supplies the interaction responses that would be requested by the package in interactive mode. *response* must be a full pathname.

`-R` *root_path*
Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* may be specified when installing to a client from a server (for example, `/export/root/client1`).

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-s` *spool*
Write the package into the directory *spool* instead of installing it.

`-v`
Trace all of the scripts that get executed by `pkgadd`, located in the *pkginst*`/install` directory. This option is used for debugging the procedural and non-procedural scripts.

`-V` *fs_file*
Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the $*root_path*`/etc/vfstab` file is non-existent or unreliable.

`-x` *proxy*
Specify a HTTP[S] proxy to use when downloading packages The format of proxy is *host*:*port*, where *host* is the hostname of the HTTP[S] proxy, and *port* is the port number associated with the proxy. This switch overrides all other methods of specifying a proxy. See ENVIRONMENT VARIABLES for more information on alternate methods of specifying a default proxy.

When executed without options or operands, `pkgadd` uses `/var/spool/pkg` (the default spool directory).

Operands The following operands are supported:

Sources By default, pkgadd looks in the `/var/spool/pkg` directory when searching for instances of a package to install or spool. Optionally, the source for the package instances to be installed or spooled can be specified using:

`-d` *device*
`-d` *datastream pkgname*,... | `all`
Install or copy a package from *device*. *device* can be any of the following:

- A full path name to a directory or the identifiers for tape, floppy disk, or removable disk (for example, `/var/tmp` or `/floppy/`*floppy_name*).
- A device alias (for example, `/floppy/floppy0`).
- A datastream created by `pkgtrans` (see `pkgtrans(1)`).
- A URL pointing to a datastream created by `pkgtrans`. The supported Universal Resource Identifiers (URIs) are `http:` and `https:`.

The second form of the `-d` specifier, above, indicates the syntax you use when specifying a datastream. In this case you must specify either a comma-separated list of package names or the keyword `all`.

Instances By default, `pkgadd` searches the specified source, and presents an interactive menu allowing the user to select which package instances found on the source are to be installed. As an alternative, the package instances to be installed can be specified using:

pkginst
The package instance or list of instances to be installed. The token `all` may be used to refer to all packages available on the source medium. The format *pkginst*`.*` can be used to indicate all instances of a package.

The asterisk character (*) is a special character to some shells and may need to be escaped. In the C-Shell, the asterisk must be surrounded by single quotes (`'`) or preceded by a backslash (\).

`-Y` *category*[,*category*...]
Install packages based on the value of the `CATEGORY` parameter stored in the package's `pkginfo(4)` file. All packages on the source medium whose `CATEGORY` matches one of the specified categories will be selected for installation or spooling.

Keystore Locations Package and patch tools such as `pkgadd` or `patchadd` use a set of trusted certificates to perform signature validation on any signatures found within the packages or patches. If there are no signatures included in the packages or patches then signature validation is skipped. The certificates can come from a variety of locations. If `-k` *keystore* is specified, and *keystore* is a directory, then *keystore* is assumed to be the base directory of the certificates to be used. If *keystore* is a file, then the file itself is assumed to have all required keys and certificates. When `-k` is not specified, then `/var/sadm/security` is used as the base directory.

Within the specified base directory, the store locations to be searched are different based on the application doing the searching and the type of store being searched for. The following directories are searched in the specified order:

1. <*store_dir*>/<*app_name*>/<*store_type*>
2. <*store_dir*>/<*store_type*>

Where <*store_dir*> is the directory specified by `-k`, <*app_name*> is the name of the application doing the searching, and <*store_type*> is one of `keystore` (for private keys), `certstore` (for untrusted public key certificates), or `truststore` (for trusted certificate authority certificates).

For example, when `pkgadd` is run with `-k /export/certs`, then the following locations are successively searched to find the trust store:

1. /export/certs/pkgadd/truststore
2. /export/certs/truststore

This searching order enables administrators to have a single location for most applications, and special certificate locations for certain applications.

Keystore And Certificate Formats

The packaging and patching utilities, such as `pkgtrans` and `patchadd`, require access to a set of keys and certificates in order to sign, and optionally verify, packages and patches.

The keystore files found by following the search pattern specified in `KEYSTORE LOCATIONS` must each be a self-contained PKCS#12-format file.

When signing a package with `pkgtrans`, if a `certstore` has more than one public key certificate, then each public key must have a `friendlyName` attribute in order to be identifiable and selectable with the `-a` option when signing packages or patches. In addition, the public key certificate selected with `-a` and found in the `certstore` must have an associated private key in the keystore.

Several browsers and utilities can be used to export and import certificates and keys into a PKCS#12 keystore. For example, a trusted certificate can be exported from Mozilla, and then imported into a PKCS#12 keystore for use with `pkgadd` with the OpenSSL Toolkit.

Pass Phrase Arguments

`pkgtrans` and `pkgadd` accept password arguments, typically using `-p` to specify the password. These allow the password to be obtained from a variety of sources. Both of these options take a single argument whose format is described below. If no password argument is given and a password is required then the user is prompted to enter one: this will typically be read from the current terminal with echoing turned off.

`pass:`*password*
: The actual password is *password*. Because the password is visible to utilities such as `ps` this form should only be used where security is not important.

`env:`*var*
: Obtain the password from the environment variable *var*. Because the environment of other processes is visible on certain platforms this option should be used with caution.

`file:`*pathname*
: The first line contained within *pathname* is the password. *pathname* need not refer to a regular file: it could, for example, refer to a device or named pipe. For example, to read the password from standard input, use `file:/dev/stdin`.

`console`
: Read the password from `/dev/tty`.

Examples

EXAMPLE 1 Installing a Package from a Solaris DVD

The following example installs a package from a Solaris DVD. You are prompted for the name of the package you want to install.

```
example# pkgadd -d /cdrom/cdrom0/s0/Solaris_10/Product
```

EXAMPLE 2 Installing a Set of Packages from a Datastream

The example command shown below installs all of the packages in the datastream specified by the `-d` source specifier. Prior to this command, this datastream must have been created with the `pkgtrans(1)` command.

EXAMPLE 2 Installing a Set of Packages from a Datastream *(Continued)*

```
example# pkgadd -d /var/tmp/datastream all
```

The keyword `all` specifies that all of the packages found in the designated datastream will be installed.

Exit Status

`0`
Successful completion

`1`
Fatal error.

`2`
Warning.

`3`
Interruption.

`4`
Administration.

`5`
Administration. Interaction is required. Do not use `pkgadd -n`.

`10`
Reboot after installation of all packages.

`20`
Reboot after installation of this package.

Environment Variables

`HTTPPROXY`
Specifies an HTTP proxy host. Overrides administration file setting, and `http_proxy` environment variable.

`HTTPPROXYPORT`
Specifies the port to use when contacting the host specified by `HTTPPROXY`. Ignored if `HTTPPROXY` is not set.

`http_proxy`
URL format for specifying proxy host and port. Overrides administration file setting.

Files

`/var/sadm/install/logs/`
Location where `pkgadd` logs an instance of software installation.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpkgcmdsu

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also pkginfo(1), pkgmk(1), pkgparam(1), pkgproto(1), pkgtrans(1), installf(1M), pkgadm(1M), pkgask(1M), pkgchk(1M), pkgrm(1M), removef(1M), admin(4), pkginfo(4), attributes(5), largefile(5), zones(5)

Application Packaging Developer's Guide

`http://www.openssl.org`

Notes When transferring a package to a spool directory, the `-r`, `-n`, and `-a` options cannot be used.

The `-r` option can be used to indicate a directory name as well as a filename. The directory can contain numerous response files, each sharing the name of the package with which it should be associated. This would be used, for example, when adding multiple interactive packages with one invocation of `pkgadd`. In this situation, each package would need a response file. If you create response files with the same name as the package (for example, `pkinst1` and `pkinst2`), then name the directory in which these files reside after the `-r`.

The `-n` option causes the installation to halt if any interaction is needed to complete it.

If the default *admin* file is too restrictive, the administration file may need to be modified to allow for total non-interaction during a package installation. See admin(4) for details.

If a package stream is specified with `-d`, and a digital signature is found in that stream, the default behavior is to attempt to validate the certificate and signature found. This behavior can be overridden with `admin` file settings. See admin(4) for more information.

Name pkgadm – manage packaging and patching system

Synopsis

```
pkgadm addcert [-ty] [-a app] [-k keystore] [-e keyfile]
     [-f format] [-n name] [-P passarg]
     [-p import_passarg] [-R rootpath] certfile

pkgadm removecert [-a app] [-k keystore] -n name
     [-P passarg] [-R rootpath]

pkgadm listcert [-a app] [-f format] [-k keystore] -n name
     [-P passarg] [-o outfile] [-R rootpath]

pkgadm dbstatus [-R rootpath]

pkgadm sync [-R rootpath] [-q]

pkgadm -V

pkgadm -?
```

Description The `pkgadm` utility is used for managing the packaging and patching system. It has several subcommands that perform various operations relating to packaging. The `pkgadm` command includes subcommands for managing certificates and keys used.

Managing Keys and Certificates

`pkgadm` maintains the packaging-system-wide keystore in `/var/sadm/security`, and individual user's certificates in `~/.pkg/security`. The following subcommands operate on the package keystore database:

`addcert`
: Add (import) a certificate into the database, with optional trust. Once added, trusted certificates can be used to verify signed packages and patches. Non-trusted user certificates and their associated keys can be used to sign packages and patches. Added user certificates are *not* used to build certificate chains during certificate verification.

`removecert`
: Removes a user certificate/private key pair, or a trusted certificate authority certificate from the keystore. Once removed, the certificate and keys cannot be used.

`listcert`
: Print details of one or more certificates in the keystore.

`sync`
: Writes the contents file and rolls the contents log file. With use of the `-q` option, forces the contents file server to quit.

Internal Install Database

The Solaris operating system relies upon enhanced System V revision 4 (SVr4) packages as the basis for its software installation and revision management. The package maintenance software stores information about installed packages in an internal database. The `pkgadm` subcomand `dbstatus` is used to determine how the package internal database is implemented. The `dbstatus` command returns a string that indicates the type of internal database in use. In

the current implementation, the `dbstatus` command always returns the string `text`, which indicates that the `contents(4)` package database is inuse. Future releases of Solaris might supply alternative database implementations.

Options The following options are supported:

`-a` *app*
If this option is used, then the command only affects the keystore associated with a particular application. Otherwise, the global keystore is affected.

`-e` *keyfile*
When adding a non-trusted certificate/key combination, this option can be used to specify the file that contains the private key. If this option is not used, the private key must be in the same file as the certificate being added.

`-f` *format*
When adding certificates, this specifies the format to expect certificates and private keys in. Possible values when adding are:

`pem`
Certificate and any private key uses PEM encoding.

`der`
Certificate and any private key uses DER encoding.

When printing certificates, this specifies the output format used when printing. Acceptable values for format are:

`pem`
Output each certificate using PEM encoding.

`der`
Output each certificate using DER encoding.

`text`
Output each certificate in human-readable format.

`-k` *keystore*
Overrides the default location used when accessing the keystore.

`-n` *name*
Identifies the entity in the store on which you want to operate. When adding a user certificate, or removing certificates, this name is required. The name is associated with the certificate/key combination, and when adding, can be used later to reference the entity. When printing certificates, if no alias is supplied, then all keystore entities are printed.

`-o` *outfile*
Output the result of the command to *outfile*. Only used when examining (printing) certificates from the key store. Standard out is the default.

`-P` *passarg*
Password retrieval method to use to decrypt keystore specified with `-k`, if required. See `PASS PHRASE ARGUMENTS` in `pkgadd(1M)` for more information about the format of this option's argument. `console` is the default.

`-p` *import_passarg*
This option's argument is identical to `-P`, but is used for supplying the password used to decrypt the certificate and/or private key being added. `console` is the default.

`-q`
(Applies to `sync` subcommand.) Shuts down the contents file cache daemon.

`-R` *rootpath*
Defines the full name of a directory to use as the root (`/`) path. The default user location of the certificate operations is `${HOME}/.pkg`. If the `-R` option is supplied, the certificates and keys will be stored under <*altroot*>`/var/sadm/security`. Note that this operation fails if the user does not have sufficient permissions to access this directory. The `listcert` command requires read permission, while `addcert` and `removecert` require both read and write permission.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-t`
Indicates the certificate being added is a trusted CA certificate. The details of the certificate (including the Subject Name, Validity Dates, and Fingerprints) are printed and the user is asked to verify the data. This verification step can be skipped with `-y`. When importing a trusted certificate, a private key should not be supplied, and will be rejected if supplied. Once a certificate is trusted, it can be used as a trust anchor when verifying future untrusted certificates.

`-V`
Print version associated with packaging tools.

`-y`
When adding a trusted certificate, the details of the certificate (Subject name, Issuer name, Validity dates, Fingerprints) are shown to the user and the user is asked to verify the correctness before proceeding. With `-y`, this additional verification step is skipped.

`-?`
Print help message.

Operands The following operand is supported:

`certfile`
File containing the certificate and optional private key, used when adding a trust anchor or certificate/key combination. Certificates must be encoded using PEM or binary DER.

Keystore Aliases All keystore entries (user cert/key and trusted certificate entries) are accessed via unique aliases. Aliases are case-sensitive.

An alias is specified when you add an entity to a keystore using the `addcert` or `trustcert` subcommand. If an alias is not supplied for a trust anchor, the trust anchor's Common Name is used as the alias. An alias is required when adding a signing certificate or chain certificate. Subsequent `pkgcert` or other package tool commands must use this same alias to refer to the entity.

Keystore Passwords See the `pkgadd(1M)` man page for a description of the passwords supplied to the `pkgadm` utility.

Examples

EXAMPLE 1 Adding a Trust Anchor

The following example adds a well-known and trusted certificate to be used when verifying signatures on packages.

```
example% pkgadm addcert -t /tmp/certfile.pem
```

EXAMPLE 2 Adding a Signing Certificate

The following example adds a signing certificate and associated private key, each of which is in a separate file, which can then be used to sign packages.

```
example% pkgadm addcert -a pkgtrans -e /tmp/keyfile.pem \
/tmp/certfile.pem
```

EXAMPLE 3 Printing Certificates

The following example prints all certificates in the root keystore.

```
example% pkgadm listcert
```

Exit Status

`0`
successful completion

`non-zero`
fatal error

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpkgcmdsu
Interface Stability	Evolving

See Also `pkginfo(1)`, `pkgmk(1)`, `pkgparam(1)`, `pkgproto(1)`, `pkgtrans(1)`, `svcs(1)`, `installf(1M)`, `pkgadd(1M)`, `pkgask(1M)`, `pkgrm(1M)`, `removef(1M)`, `svcadm(1M)`, `admin(4)`, `contents(4)`, `exec_attr(4)`, `pkginfo(4)`, `attributes(5)`, `rbac(5)`, `smf(5)`

Application Packaging Developer's Guide

Notes The service for `pkgadm` is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/pkgserv
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name pkgask – stores answers to a request script

Synopsis

```
pkgask [-d device] [-R root_path] -r response pkginst...
```

Description `pkgask` allows the administrator to store answers to an interactive package (one with a `request` script, that is, a user-created file that must be named `request`). Invoking this command generates a response file that is then used as input at installation time. The use of this response file prevents any interaction from occurring during installation since the file already contains all of the information the package needs.

Options The following options are supported

`-d` *device* Run the request script for a package on *device*. *device* can be a directory pathname or the identifiers for tape, floppy disk or removable disk (for example, `/var/tmp`, `/dev/diskette`, and `/dev/dsk/c1d0s0`). The default device is the installation spool directory.

`-R` *root_path* Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-r` *response* Identify a file or directory which should be created to contain the responses to interaction with the package. The name must be a full pathname. The file, or directory of files, can later be used as input to the `pkgadd(1M)` command.

Operands The following operands are supported:

pkginst Specify the package instance, or list of instances for which `request` scripts will be created. The token `all` may be used to refer to all packages available on the source medium.

Exit Status

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `pkginfo(1)`, `pkgmk(1)`, `pkgparam(1)`, `pkgproto(1)`, `pkgtrans(1)`, `installf(1M)`, `pkgadd(1M)`, `pkgchk(1M)`, `pkgrm(1M)`, `removef(1M)`, `admin(4)`,`attributes(5)`

Application Packaging Developer's Guide

Notes The `-r` option can be used to indicate a directory name as well as a filename. The directory name is used to create numerous response files, each sharing the name of the package with which it should be associated. This would be used, for example, when you will be adding multiple interactive packages with one invocation of `pkgadd(1M)`. Each package would need a response file. To create multiple response files with the same name as the package instance, name the directory in which the files should be created and supply multiple instance names with the `pkgask` command. When installing the packages, you will be able to identify this directory to the `pkgadd(1M)` command.

If the default `admin` file is too restrictive, the administration file may need to be modified to allow for total non-interaction during a package installation. See`admin(4)` for details.

Name pkgchk – check package installation accuracy

Synopsis

```
pkgchk [-l | -acfnqvx] [-i file | -]
     [-p path... | -P partial-path...] [-R root_path]
     [ [-m pkgmap [-e envfile]] | pkginst... | -Y category,category...]

pkgchk -d device [-l | -fv] [-i file | -] [-M] [-p path]...
     [-V fs_file]
     [pkginst... | -Y category[,category...]]
```

Description `pkgchk` checks the accuracy of installed files or, by using the `-l` option, displays information about package files. `pkgchk` checks the integrity of directory structures and files. Discrepancies are written to standard error along with a detailed explanation of the problem.

The first synopsis defined above is used to list or check the contents and/or attributes of objects that are currently installed on the system, or in the indicated `pkgmap`. Package names may be listed on the command line, or by default, the entire contents of a machine will be checked.

The second synopsis is used to list or check the contents of a package which has been spooled on the specified device, but not installed. Note that attributes cannot be checked for spooled packages.

Options The following options are supported:

`-a`
Audit the file attributes only and do not check file contents. Default is to check both.

`-c`
Audit the file contents only and do not check file attributes. Default is to check both.

`-d` *device*
Specify the device on which a spooled package resides. *device* can be a directory path name or the identifiers for tape, floppy disk, or removable disk (for example, `/var/tmp` or `/dev/diskette`).

`-e` *envfile*
Request that the package information file named as *envfile* be used to resolve parameters noted in the specified `pkgmap` file.

`-f`
Correct file attributes if possible. If used with the `-x` option, this option removes hidden files. When `pkgchk` is invoked with this option, it creates directories, named pipes, links, and special devices if they do not already exist. If the `-d` option calls out an uninstalled package, the `-f` option will only take effect if the package is in directory (not stream) format. All file attributes will be set to agree with the entries in the `pkgmap` file except that setuid, setgid, and sticky bits will not be set in the mode.

`-i` *file* | –
Read a list of path names from *file* or from stdin (–) and compare this list against the installation software database or the indicated `pkgmap` file. Path names that are not contained in *file* or stdin are not checked.

`-l`
List information on the selected files that make up a package. This option is not compatible with the `-a`, `-c`, `-f`, `-g`, and `-v` options.

`-m pkgmap`
Check the package against the package map file, `pkgmap`.

`-M`
Instruct `pkgchk` not to use the `$`*root_path*`/etc/vfstab` file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

`-n`
Do not check volatile or editable files' contents. This should be used for most post-installation checking.

`-p` *path*
Check the accuracy only of the path name or path names listed. *path* can be one or more path names separated by commas (or by whitespace, if the list is quoted).

To specify a *path* that includes a comma, you must use the `-i` option, described above. See EXAMPLES.

`-P` *partial-path*
Check the accuracy of only the partial path name or path names listed. *partial-path* can be one or more partial path names separated by commas (or by whitespace, if the list is quoted). This option can be used instead of `-p` and is not compatible with the other option. This option matches any path name that contains the string contained in the partial path. See the note about paths that contain commas in the description of `-p`.

`-q`
Quiet mode. Do not give messages about missing files.

`-R` *root_path*
Define the full name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* may be specified when installing to a client from a server (for example, `/export/root/client1`).

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-v`
Verbose mode. Files are listed as processed.

`-V` *fs_file*
Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the `$`*root_path*`/etc/vfstab` file is non-existent or unreliable.

`-x`
Search exclusive directories, looking for files which exist that are not in the installation software database or the indicated `pkgmap` file.

`-Y` *category*
Check packages based on the value of the `CATEGORY` parameter stored in the installed or spooled package's `pkginfo(4)` file.

Operands *pkginst*
The package instance or instances to be checked. The format *pkginst*`.*` can be used to check all instances of a package. The default is to display all information about all installed packages.

The asterisk character (`*`) is a special character to some shells and may need to be escaped. In the C-Shell, an asterisk must be surrounded by single quotes (`'`) or preceded by a backslash (\);

partial-path
A portion of a path, such as a file or directory name.

Examples **EXAMPLE 1** Using `pkgchk` for Displaying Package Installation Information

The following example displays package installation information for `/usr/bin/ls`:

```
example% pkgchk -l -p /usr/bin/ls
```

EXAMPLE 2 Checking on Java Font Properties

The following example displays package installation information for all Java font properties installed on the system.

```
example% pkgchk -l -P font.properties
```

EXAMPLE 3 Specifying a Path That Contains a Comma

Assume you want to specify the path:

```
/platform/SUNW,Netra-T12/lib
```

List this path in a file. Here is one way in which you can do that:

```
example% echo "/platform/SUNW,Netra-T12/lib" > /tmp/p
```

You can then enter:

EXAMPLE 3 Specifying a Path That Contains a Comma *(Continued)*

```
example% pkgchk -i /tmp/p -l
Pathname: /platform/SUNW,Netra-T12/lib
Type: directory
Expected mode: 0755
Expected owner: root
Expected group: bin
Referenced by the following packages:
        SUNWcar
Current status: installed
```

Exit Status

`0`
Successful completion.

`>0`
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`pkginfo(1)`, `pkgtrans(1)`, `pkgadd(1M)`, `pkgask(1M)`, `pkgrm(1M)`, `pkginfo(4)`, `attributes(5)`, `largefile(5)`

Application Packaging Developer's Guide

Notes

Package commands are `largefile(5)`-aware. They handle files larger than 2 GB in the same way they handle smaller files. In their current implementations, `pkgadd(1M)`, `pkgtrans(1)` and other package commands can process a datastream of up to 4 GB.

Name pkgcond – determine type and capability of target

Synopsis `/usr/bin/pkgcond [-nv]` *condition*

Description The `pkgcond` command allows you to determine the type of target being operated on (global zone, non-global zone, diskless client, and so forth) and the capabilities available for that type of client (can add a driver, path is writable, and so forth). The `pkgcond` command is intended to be invoked from package and patch scripts, but can also be used in situations that mimic the context of these scripts. See `NOTES` for further guidance.

`pkgcond` has one mandatory argument, a condition. The command tests whether the condition is true for the specified path. The condition can be one of the following:

- `can_add_driver` [*path*]
- `can_remove_driver` [*path*]
- `can_update_driver` [*path*]
- `is_alternative_root` [*path*]
- `is_boot_environment` [*path*]
- `is_diskless_client` [*path*]
- `is_global_zone` [*path*]
- `is_mounted_miniroot` [*path*]
- `is_netinstall_image` [*path*]
- `is_nonglobal_zone` [*path*]
- `is_path_writable` *path*
- `is_running_system` [*path*]
- `is_sparse_root_nonglobal_zone` [*path*]
- `is_what` [*path*]
- `is_whole_root_nonglobal_zone` [*path*]

The *path* argument usually denotes the root of the global zone or non-global zone, or alternate root. If path is optional and not specified, the default is `/`.

The behavior of the `is_what` condition is somewhat special, because it displays results of all other conditions to standard output.

Options The following options are supported:

`-n`
Negate return status (`0` becomes `1` and `1` becomes `0`). It negates results in the case of `is_what` condition.

`-v`
Verbose mode. Displays detailed data about intermediate checks performed.

Examples **EXAMPLE 1** Listing All Available Information

The following command lists all available information about the current running system, in a user-friendly way.

```
example# pkgcond -n is_what
```

EXAMPLE 2 Determining if Target is an Alternate Root

The following command determines whether an alternate boot environment exists under `/altroot_mount`.

```
example# pkgcond is_alternative_root /altroot_mount
```

Exit Status

`0`
Condition is true unless `-n` was specified.

`1`
Condition is false unless `-n` was specified.

`2`
Command line usage error.

`3`
Command failed to perform the test due to a fatal error.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs
Interface Stability	Uncommitted

See Also

`pkgtrans(1)`, `pkgadd(1M)`, `pkgask(1M)`, `pkgchk(1M)`, `pkgrm(1M)`, `pkginfo(4)`, `attributes(5)`

Notes

Supported usage of `pkgcond` is subject to the following constraints:

1. Do not use `pkgcond` outside of the Solaris marketing release in which it is provided (for example, do not use Solaris 10 `pkgcond` against a Solaris 9 target).
2. Restrict use of the optional path argument according to the following rules:
   - The command `pkgcond` *condition* `$ROOTDIR` can be used in patch level scripts.
   - The command `pkgcond` *condition* `$PKG_INSTALL_ROOT` can be used in package level scripts.
   - A command of the form `pkgcond` *condition without* the optional path argument can be used in any context.

Use of `pkgcond` with an arbitrary path argument is *not* recommended or supported, as the results returned might not be accurate.

Name pkgrm – remove a package from the system

Synopsis

```
pkgrm [-nv] [-a admin] [ [-A | -M] -R root_path]
     [-V fs_file]
     [pkginst... | -Y category[,category...]]

pkgrm -s spool
     [pkginst... | -Y category[,category...]]
```

Description `pkgrm` will remove a previously installed or partially installed package from the system. A check is made to determine if any other packages depend on the one being removed. If a dependency exists, the action taken is defined in the `admin` file.

The default state for the command is in interactive mode, meaning that prompt messages are given during processing to allow the administrator to confirm the actions being taken. Non-interactive mode can be requested with the `-n` option.

The `-s` option can be used to specify the directory from which spooled packages should be removed.

Certain unbundled and third-party packages are no longer entirely compatible with the latest version of `pkgrm`. These packages require user interaction throughout the removal and not just at the very beginning.

To remove these older packages (released prior to Solaris 2.4), set the following environment variable:`NONABI_SCRIPTS=TRUE` `pkgrm` permits keyboard interaction throughout the removal as long as this environment variable is set.

Options The following options are supported:

`-a` *admin*
: Use the installation administration file, `admin`, in place of the default `admin` file. `pkgrm` first looks in the current working directory for the administration file. If the specified administration file is not in the current working directory, `pkgrm` looks in the `/var/sadm/install/admin` directory for the administration file.

`-A`
: Remove the package files from the client's file system, absolutely. If a file is shared with other packages, the default behavior is to not remove the file from the client's file system.

`-M`
: Instruct `pkgrm` not to use the $*root_path*`/etc/vfstab` file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

`-n`
: Non-interactive mode. If there is a need for interaction, the command will exit.

Use of this option requires that at least one package instance be named upon invocation of the command. Certain conditions must exist for a package to be removed non-interactively or a non-restrictive `admin` file needs to be used.

`-R` *root_path*
Defines the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-s` *spool*
Remove the specified package(s) from the directory *spool*. The default directory for spooled packages is `/var/sadm/pkg`.

`-v`
Trace all of the scripts that get executed by `pkgrm`, located in the *pkginst*`/install` directory. This option is used for debugging the procedural and non-procedural scripts.

`-V` *fs_file*
Specify an alternative *fs_file* to map the client's file systems. Used in situations where the $*root_path*`/etc/vfstab` file is non-existent or unreliable.

`-Y` *category*
Remove packages based on the value of the `CATEGORY` parameter stored in the installed or spooled package's `pkginfo(4)` file. No package with the `CATEGORY` value of `system` can removed from the file system with this option.

Operands The following operand is supported:

pkginst
Specifies the package to be removed. The format *pkginst*`.*` can be used to remove all instances of a package.

The asterisk character (*) is a special character to some shells and may need to be escaped. In the C-Shell, "*" must be surrounded by single quotes (') or preceded by a backslash (\).

Examples **EXAMPLE 1** Removing All Instances of `SUNWjunk` from `client1`

The following example removes all instances of `SUNWjunk` from `client1`:

```
example% pkgrm -R /export/root/client1 SUNWjunk*
```

Note the caveat on the use of the `-R` option in the description of that option, above.

Exit Status The following exit values are returned:

`0`
Successful completion.

`1`
Fatal error.

`2`
Warning.

`3`
Interruption.

`4`
Administration.

`10`
Reboot after removal of all packages.

`20`
Reboot after removal of this package.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `pkginfo(1)`, `pkgmk(1)`, `pkgparam(1)`, `pkgproto(1)`, `pkgtrans(1)`, `installf(1M)`, `pkgadd(1M)`, `pkgask(1M)`, `pkgchk(1M)`, `removef(1M)`, `admin(4)`, `pkginfo(4)`, `attributes(5)`, `largefile(5)`

Application Packaging Developer's Guide

Notes Package commands are `largefile(5)`-aware. They handle files larger than 2 GB in the same way they handle smaller files. In their current implementations, `pkgadd(1M)`, `pkgtrans(1)` and other package commands can process a datastream of up to 4 GB.

Name plockstat – report user-level lock statistics

Synopsis

```
plockstat [-vACHV] [-n count] [-s depth] [-e secs]
     [-x arg [=val]] command [arg]...

plockstat [-vACHV] [-n count] [-s depth] [-e secs]
     [-x arg [=val]] -p pid
```

Description The `plockstat` utility gathers and displays user-level locking statistics. By default, `plockstat` monitors all lock contention events, gathers frequency and timing data about those events, and displays the data in decreasing frequency order, so that the most common events appear first.

`plockstat` gathers data until the specified command completes or the process specified with the `-p` option completes.

`plockstat` relies on DTrace to instrument a running process or a command it invokes to trace events of interest. This imposes a small but measurable performance overhead on the processes being observed. Users must have the `dtrace_proc` privilege and have permission to observe a particular process with `plockstat`. Refer to the *Solaris Dynamic Tracing Guide* for more information about DTrace security features.

Options The following options are supported:

`-A`	Watch all lock events. This option is equivalent to `-CH`.
`-C`	Watch contention events.
`-H`	Watch hold events.
`-e` *secs*	Exit after the number of seconds specified have elapsed.
`-n` *count*	Display only the specified number of entries for each output category.
`-s` *depth*	Record a stack trace rather than just the calling function.
`-p` *pid*	Specify a process ID from which `plockstat` is to gather data.
`-v`	Print out a message to indicate that tracing has started.
`-x` *arg*[=*val*]	Enable or modify a DTrace runtime option or D compiler option. The list of options is found in the *Solaris Dynamic Tracing Guide*. Boolean options are enabled by specifying their name. Options with values are set by separating the option name and value with an equals sign (=).
`-V`	Print the Dtrace commands used to gather the data. The output can then be used directly with the `dtrace(1M)` command.

Operands The following operands are supported:

arg	A string to be passed as an argument to *command*.
command	The name of a utility to be invoked.

count A positive integer value.

pid A process identifier for a process to be monitored.

secs Duration specified as a positive integer number of seconds.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdtrc
Interface Stability	See below.

The command-line syntax is Evolving. The human-readable output is Unstable.

See Also `dtrace(1M)`, `lockstat(1M)`, `mutex_init(3C)`, `pthread_mutex_lock(3C)`, `pthread_rwlock_rdlock(3C)`, `pthread_rwlock_wrlock(3C)`, `pthread_rwlock_unlock(3C)`, `rwlock(3C)`, `attributes(5)`, `fasttrap(7D)`

Solaris Dynamic Tracing Guide

Name pmadm – port monitor administration

Synopsis

```
pmadm -a [-p pmtag | -t type] -s svctag -i id -m pmspecific -v ver
     [-f xu]
     [-y comment] [-z  script]

pmadm -r -p pmtag -s svctag

pmadm -e -p pmtag -s svctag

pmadm -d -p pmtag -s svctag

pmadm -l [-t type | -p pmtag] [-s svctag]

pmadm -L [-t type | -p pmtag] [-s svctag]

pmadm -g -p pmtag -s svctag [-z script]

pmadm -g -s svctag -t type -z script
```

Description `pmadm` is the administrative command for the lower level of the Service Access Facility hierarchy, that is, for service administration. A port may have only one service associated with it although the same service may be available through more than one port. In order to uniquely identify an instance of a service, the `pmadm` command must identify both the port monitor or port monitors through which the service is available (`-p` or `-t`) and the service (`-s`). See Options.

`pmadm` performs the following functions:

- adds or removes a service
- enables or disables a service
- installs or replaces a per-service configuration script
- prints requested service information

Any user on the system may invoke `pmadm` to request service status (`-l` or `-L`) or to print per-service configuration scripts (`-g` without the `-z` option). `pmadm` with other options may be executed only by a privileged user.

Options The following options are supported:

`-a` Add a service. `pmadm` adds an entry for the new service to the port monitor's administrative file. Because of the complexity of the options and arguments that follow the `-a` option, it may be convenient to use a command script or the menu system to add services.

`-d` Disable a service. Add `x` to the flag field in the entry for the service *svctag* in the port monitor's administrative file. This is the entry used by port monitor *pmtag*. See the `-f` option, below, for a description of the flags available.

`-e` Enable a service. Remove `x` from the flag field in the entry for the service *svctag* in the port monitor administrative file. This is the entry used by port monitor *pmtag*. See the `-f` option, below, for a description of the flags available.

`-f xu` The `-f` option specifies one or both of the following two flags which are then included in the flag field of the entry for the new service in the port monitor's administrative file. If the `-f` option is not included, no flags are set and the default conditions prevail. By default, a new service is enabled and no `utmpx` entry is created for it. An `-f` option without a following argument is illegal.

- `x` Do not enable the service *svctag* available through port monitor *pmtag*.
- `u` Create a `utmpx` entry for service *svctag* available through port monitor *pmtag*.

`-g` Print, install, or replace a per-service configuration script. The `-g` option with a `-p` option and a `-s` option prints the per-service configuration script for service *svctag* available through port monitor *pmtag*. The `-g` option with a `-p` option, a `-s` option, and a `-z` option installs the per-service configuration script contained in the file `script` as the per-service configuration script for service *svctag* available through port monitor *pmtag*. The `-g` option with a `- s` option, a `-t` option, and a `-z` option installs the file `script` as the per-service configuration script for service *svctag* available through any port monitor of type `type`. Other combinations of options with `-g` are invalid.

`-i` *id* `id` is the identity that is to be assigned to service *svctag* when it is started. `id` must be an entry in `/etc/passwd`.

`-l` The `-l` option requests service information. Used by itself and with the options described below, it provides a filter for extracting information in several different groupings.

`-l`	By itself, the `-l` option lists all services on the system.
`-l -p` *pmtag*	Lists all services available through port monitor *pmtag*.
`-l -s` *svctag*	Lists all services with tag *svctag*.
`-l -p` *pmtag*`-s`*svctag*	Lists service *svctag*.
`-l -t` *type*	Lists all services available through port monitors of type `type`.
`-l -t` *type*`-s`*svctag*	Lists all services with tag *svctag* available through a port monitor of type `type`.

Other combinations of options with `-l` are invalid.

`-L` The `-L` option is identical to the `-l` option except that output is printed in a condensed format.

`-m` *pmspecific* *pmspecific* is the port monitor-specific portion of the port monitor administrative file entry for the service.

`-p` *pmtag* Specifies the tag associated with the port monitor through which a service (specified as `-s` *svctag*) is available.

`-r` Remove a service. When `pmadm` removes a service, the entry for the service is removed from the port monitor's administrative file.

`-s` *svctag* Specifies the service tag associated with a given service. The service tag is assigned by the system administrator and is part of the entry for the service in the port monitor's administrative file.

`-t` *type* Specifies the port monitor type.

`-v` *ver* Specifies the version number of the port monitor administrative file. The version number may be given as

`-v 'pmspec -V`

where *pmspec* is the special administrative command for port monitor *pmtag*. This special command is `ttyadm` for `ttymon` and `nlsadmin` for `listen`. The version stamp of the port monitor is known by the command and is returned when *pmspec* is invoked with a `-V` option.

`-y` *comment* Associate *comment* with the service entry in the port monitor administrative file.

`-z` *script* Used with the `-g` option to specify the name of the file that contains the per-service configuration script. Modifying a configuration script is a three-step procedure. First a copy of the existing script is made (`-g` alone). Then the copy is edited. Finally, the copy is put in place over the existing script (`-g` with `-z`).

Options that request information write the requested information to the standard output. A request for information using the `-l` option prints column headers and aligns the information under the appropriate headings. In this format, a missing field is indicated by a hyphen. A request for information in the condensed format using the `-L` option prints the information in colon-separated fields; missing fields are indicated by two successive colons. # is the comment character.

Examples

EXAMPLE 1 Adding a Service to a Port Monitor with the Tag `pmtag`

The following command adds a service to a port monitor with tag `pmtag` and gives the service the tag `svctag`. The port monitor-specific information is generated by `specpm`. The service defined by `svctag` will be invoked with identity `root`.

```
pmadm -a -p pmtag -s svctag -i root -m ‘specpm -a arg1 -b arg2‘-v ‘specpm -V‘
```

EXAMPLE 2 Adding a Service with Service Tab `svctag`

The following command adds a service with service tag `svctag`, identity `guest`, and port monitor-specific information generated by `specpm` to all port monitors of type `type`:

```
pmadm -a -s svctag -i guest -t type -m ‘specpm -a arg1 -b arg2‘-v ‘specpm -V‘
```

EXAMPLE 3 Removing a Service

The following command removes the service `svctag` from port monitor `pmtag`:

```
pmadm -r -p pmtag -s svctag
```

EXAMPLE 4 Enabling a Service

The following command enables the service `svctag` available through port monitor `pmtag`:

```
pmadm -e -p pmtag -s svctag
```

EXAMPLE 5 Disabling a Service

The following command disables the service `svctag` available through port monitor `pmtag`:

```
pmadm -d -p pmtag -s svctag
```

EXAMPLE 6 Listing Status Information

The following command lists status information for all services:

```
pmadm -l
```

EXAMPLE 7 Listing Status Information

The following command lists status information for all services available through the port monitor with tag `ports`:

```
pmadm -l -p ports
```

EXAMPLE 8 Listing Status Information in Condensed Format

The following command lists the status information for all services available through the port monitor with tag `ports` in condensed format:

```
pmadm -L -p ports
```

EXAMPLE 9 Listing Status Information for All Services

List status information for all services available through port monitors of type `listen`:

```
pmadm -l -t listen
```

EXAMPLE 10 Printing the per-service Configuration

The following command prints the per-service configuration script associated with the service `svctag` available through port monitor `pmtag`:

```
pmadm -g -p pmtag -s svctag
```

Exit Status

The following exit values are returned:

`0` Successful operation.

`>0` Operation failed.

Files

`/etc/saf/`*pmtag*`/_config`

`/etc/saf/`*pmtag*`/`*svctag*

`/var/saf/`*pmtag*`/*`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`sac(1M)`, `sacadm(1M)`, `doconfig(3NSL)`, `attributes(5)`

Name pmconfig – Configure the Power Management system

Synopsis

```
/usr/sbin/pmconfig [-r]
```

Description The `pmconfig` utility sets the Power Management and suspend-resume configuration. User has permission to change Power Management configuration using `pmconfig` only if he is allowed to do so according to `PMCHANGEPERM` keyword of `/etc/default/power`. User has permission to change the suspend-resume configuration using `pmconfig` only if he is allowed to do so according to the `CPRCHANGEPERM` keyword of `/etc/default/power`. See FILES section below for a description of the `PMCHANGEPERM` and `CPRCHANGEPERM` keywords of `/etc/default/power`.

Based on user permissions, `pmconfig` first resets the Power Management and/or suspend-resume state back to its default and then reads the new Power Management and/or suspend-resume configuration from `/etc/power.conf` and issues the commands to activiate the new configuration. The `pmconfig` utility is run at system boot. This utility can also be run from the command line after manual changes have been made to the `/etc/power.conf` file. For editing changes made to the `/etc/power.conf` file to take effect, users must run `pmconfig`.

The preferred interface for changing Power Management and suspend-resume configuration is `dtpower`(1M).

Options The following options are supported:

`-r` Reset Power Management and suspend-resume state to default and exit. User must have both Power Management and suspend-resume configuration permission for this option.

Exit Status The following exit values are returned:

`0` Upon successful completion

`>0` An error occurred

Files

`/etc/power.conf`	System Power Management configuration file
`/etc/default/power`	File that controls permissions for system's Power Management and suspend-resume features. The `PMCHANGEPERM` keyboard controls the Power Management configuration permissions, while the `CPRCHANGEPERM` keyword controls the suspend-resume configuration permissions.

Allowed values are:

`all`	Any user can change the configuration.
`-`	No one except super-user can change the configuration.

`<user1, user2,...>`	A user in this user list or a super-user can change the configuration. The user list is a space and/or comma (,) separated list. You must enclose the list in < and > characters.
`console-owner`	A user who owns the system console device node or a super-user can change the configuration.

The default values are `PMCHANGEPERM=console-owner` and `CPRCHANGEPERM=console-owner`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpmu
Interface stability	Unstable

See Also `svcs(1)`, `powerd(1M)`, `power.conf(4)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`, `cpr(7)`, `pm(7D)`

Using Power Management

Notes The `pmconfig` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/power:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Diagnostics If the program cannot open the configuration file, it prints an error message to standard error. If the program encounters a syntax error in the configuration file, it prints an error message and the line number of the error in the configuration file. It then skips the rest of the information on that line and processes the next line. Any configuration information already processed on the line containing the error is used. If user does not have permission to change Power Management and/or suspend-resume configuration, and configuration file has entries for which user doesn't have permission, it process the entries for which user has permissions and prints error on rest.

Name pntadm – DHCP network table management utility

Synopsis

```
pntadm -C [-r resource] [-p path] [-u uninterpreted] network

pntadm -A name_IP_address [-c comment] [-e mm/dd/yyyy]
     [-f num | keywords] [-h client_hostname]
     [-i [-a] client_ID] [-m [-y] macro] [-s server]
     [-r resource] [-p path] [-u uninterpreted] network

pntadm -M name_IP_address [-c comment] [-e mm/dd/yyyy]
     [-f num | keywords] [-h client_hostname]
     [-i [-a] client ID] [-m [-y] macro]
     [-n new_client_IP_address] [-s server] [-r resource]
     [-p path] [-u uninterpreted] network

pntadm -D name_IP_address [-y] [-r resource] [-p path]
     [-u uninterpreted] network

pntadm -P [-v] [-x] [-r resource] [-p path]
     [-u uninterpreted] network

pntadm -R [-r resource] [-p path] [-u uninterpreted] network

pntadm -L [-r resource] [-p path] [-u uninterpreted]

pntadm -B [-v] [batchfile]
```

Description The `pntadm` command is used to manage the Dynamic Host Configuration Protocol (DHCP) network tables. It is used to add and remove networks under DHCP management, and add, delete, or modify IP address records within network tables, or to view tables. For a description of the format of DHCP network tables, see `dhcp_network(4)`.

`pntadm` can be run as root or by other users assigned to the DHCP Management profile. See `rbac(5)` and `user_attr(4)`.

If the networks you want to add are subnetted, you need to update the `netmasks(4)` table.

One of the following options (function flags) must be specified with the `pntadm` command: `-A`, `-B`, `-C`, `-D`, `-L`, `-M`, `-P`, or `-R`.

Options The following options are supported:

`-A` *name_IP_address* Add a client entry with hostname or client IP address, *name_IP_address*, to the named DHCP network table.

The following sub-options are optional:

`-c` *comment*	Comment text. The default is `NULL`.
`-e` *mm/dd/yyyy*	Absolute lease. The default is `0`.
`-f` *num* | *keywords*	Flag value. The default is `00`.

The flag (`-f`) option can be specified either as a single number denoting the intended flag value, or as a series of the following keywords, combined using the plus (+) symbol:

`DYNAMIC` or `00`	Server manager's assignment.
`PERMANENT` or `01`	Lease on entry is permanent.
`MANUAL` or `02`	Administrator managed assignment.
`UNUSABLE` or `04`	Entry is not valid.
`BOOTP` or `08`	Entry reserved for `BOOTP` clients.

For a more detailed description of the flag values, see `dhcp_network(4)`.

`-h` *client_hostname*

Client hostname. The default is NULL.

When the `-h` option is used in this mode, the *client_hostname* is added to the hosts table within the resource used for storing host names (files, NIS+ or DNS). The command will fail if this *client_hostname* is already present in the hosts table.

`-i` *client_ID* [`-a`]

Client identifier [`-a`]. The default is `00`.

The `-i` option modified with `-a` specifies that the client identifier is in ASCII format, and thus needs to be converted to hexadecimal format before insertion into the table.

`-m` *macro* [`-y`]

Macro name. Default is UNKNOWN.

The `-m` option modified with `-y` verifies the existence of the named macro in the `dhcptab` table before adding the entry.

`-s` *server*

Server IP or name. Default is system name (`uname -n`).

`-B`

Activate batch mode. `pntadm` will read from the specified file or from standard input a series of `pntadm` commands and execute them within the same process. Processing many `pntadm` commands using this method is much faster than running an executable batchfile itself. Batch mode is recommended for using `pntadm` in scripts.

The following sub-option is optional:

`-v` Display commands to standard output as they are processed.

`-C`

Create the DHCP network table for the network specified by *network*. See Operands. For details, see `dhcp_network(4)` and `networks(4)`.

`-D` *name_IP_address*

Delete the specified client entry with hostname or client IP address, *name_IP_address*, in the named DHCP network table. (See `dhcp_network(4)`.)

The following sub-option is optional:

`-y` Remove associated host table entry. The `-y` option requests that all hostnames associated with the IP address in the hosts table in the resource be removed.

`-L`

List the DHCP network tables presently configured, one per line, on standard output. If none are found, no output is printed and an exit status of `0` is returned.

`-M` *name_IP_address*

Modify the specified client entry with hostname or client IP address, *name_IP_address*, in the named DHCP network table. See `dhcp_network(4)`. The default for the sub-options is what they currently are set to.

The following sub-options are optional.

`-c` *comment*	New comment text.
`-e` *mm/dd/yy*	New absolute lease expiration date. Time defaults to 12:00 AM of the day specified.
`-f` *num* | *keyboard*	New flag value, see explanation following the description of the `-A` option.
`-h` *host_name*	New client hostname.

The `-h` option allows you to change the current *hostname* associated with the IP address or to add a new *hostname* to the hosts table if an entry associated with this IP address does not exist.

`-i` *client_ID* — New client identifier [`-a`].

`-m` *macro* [`-y`] — Macro name defined in `dhcptab`.

`-n` *new_client_IP_address* — New IP address.

`-s` *server* — New server IP or name.

For more detailed description of the sub-options and flag values, see `dhcp_network(4)`.

`-P` — Display the named DHCP network table.

The following sub-options are optional:

`-v` Display lease time in full verbose format and resolve IP addresses for the clients and server to hostnames.

`-x` Display lease time in raw format.

These flag codes are used with the `-P` sub-options:

`-v`	`-x`	Description
D	00	DYNAMIC
P	01	PERMANENT
M	02	MANUAL
U	04	UNUSABLE
B	08	BOOTP

See `dhcp_network(4)` for information on these sub-options and associated flag codes.

`-p` *path* — Override the `dhcpsvc.conf(4)` configuration value for data store resource path, *path* See `dhcpsvc.conf(4)`

`-R` — Remove the named DHCP network table. See `dhcp_network(4)`.

`-r` *data_store_resource* — Override the `/etc/inet/dhcpsvc.conf` configuration value for `RESOURCE=` with the *data_store_resource* specified. See the

dhcpsvc.conf(4) man page for more details on resource type, and the *Solaris DHCP Service Developer's Guide* for more information about adding support for other data stores.

`-u` uninterpreted — Data which will be ignored by `pntadm`, but passed to the currently configured public module to be interpreted by the data store. This might be used for a database account name or other authentication or authorization parameters required by a particular data store.

Operands The following operand is supported:

network — The network address or network name which corresponds to the `dhcp network` table. See dhcp_network(4).

Examples

EXAMPLE 1 Creating a Table for the `10.0.0.0` DHCP Network

The following command creates a table for the `10.0.0.0` (subnetted to class C) DHCP network table. Note that if you have an alias for this network in your networks(4) table, you can use that value rather than the dotted Internet Address notation.

```
example# pntadm -C 10.0.0.0
```

EXAMPLE 2 Adding an Entry to the `10.0.0.0` Table

The following command adds an entry to the `10.0.0.0` table in the `files` resource in the `/var/mydhcp` directory:

```
example# pntadm -r SUNWfiles -p /var/mydhcp -A 10.0.0.1 10.0.0.0
```

EXAMPLE 3 Modifying the `10.0.0.1` Entry of the `10.0.0.0` Table

The following command modifies the `10.0.0.1` entry of the `10.0.0.0` table, changing the macro name to `Green`, setting the flags field to `MANUAL` and `PERMANENT`:

```
example# pntadm -M 10.0.0.1 -m Green -f 'PERMANENT+MANUAL' 10.0.0.0
```

EXAMPLE 4 Changing the `10.0.0.1` Entry to `10.0.0.2`

The following command changes the `10.0.0.1` entry to `10.0.0.2`, making an entry in the hosts(4) table called `myclient`:

```
example# pntadm -M 10.0.0.1 -n 10.0.0.2 -h myclient 10.0.0.0
```

EXAMPLE 5 Setting the Client ID as ASCII

The following command sets the client ID as ASCII `aruba.foo.com` for the `myclient` entry:

```
example# pntadm -M myclient -i 'aruba.foo.com' -a 10.0.0.0
```

EXAMPLE 6 Deleting the `myclient` Entry from the `10.0.0.0` Table

The following command deletes the `myclient` (`10.0.0.2`) entry from the `10.0.0.0` table:

```
example# pntadm -D myclient 10.0.0.0
```

EXAMPLE 7 Removing the Named DHCP Network Table

The following command removes the named DHCP network table in the NIS+ directory specified:

```
example# pntadm -r SUNWnisplus -p Test.Nis.Plus. -R 10.0.0.0
```

EXAMPLE 8 Listing the Configured DHCP Network Tables

The following command lists the configured DHCP network tables:

```
example# pntadm -L
192.168.0.0
10.0.0.0
```

EXAMPLE 9 Executing `pntadm` Commands in Batch Mode

The following command runs a series of `pntadm` commands contained in a batch file:

```
example# pntadm -B addclients
```

Exit Status

`0` Successful completion.

1 Object already exists.

2 Object does not exist.

3 Non-critical error.

4 Critical error.

Files

`/etc/inet/dhcpsvc.conf`

`/etc/inet/hosts`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdhcsu
Interface Stability	Evolving

See Also

`dhcpconfig(1M)`, `dhcpmgr(1M)`, `dhcp_network(4)`, `dhcpsvc.conf(4)`, `dhcptab(4)`, `hosts(4)`, `netmasks(4)`, `networks(4)`, `user_attr(4)`, `attributes(5)`, `dhcp(5)`, `dhcp_modules(5)`, `rbac(5)`

Solaris DHCP Service Developer's Guide

System Administration Guide: IP Services

Alexander, S., and R. Droms, *DHCP Options and BOOTP Vendor Extensions*, RFC 1533, Lachman Technology, Inc., Bucknell University, October 1993.

Droms, R., *Interoperation Between DHCP and BOOTP*, RFC 1534, Bucknell University, October 1993.

Droms, R., *Dynamic Host Configuration Protocol*, RFC 1541, Bucknell University, October 1993.

Wimer, W., *Clarifications and Extensions for the Bootstrap Protocol*, RFC 1542, Carnegie Mellon University, October 1993.

Name pooladm – activate and deactivate the resource pools facility

Synopsis

```
/usr/sbin/pooladm [-n] [-s] [-c] [filename] | -x

/usr/sbin/pooladm [-d | -e]
```

Description The `pooladm` command provides administrative operations on pools and sets. `pooladm` reads the specified filename and attempts to activate the pool configuration contained in it.

Before updating the current pool run-time configuration, `pooladm` validates the configuration for correctness.

Without options, `pooladm` prints out the current running pools configuration.

Options The following options are supported:

`-c` Instantiate the configuration at the given location. If a filename is not specified, it defaults to `/etc/pooladm.conf`.

`-d` Disable the pools facility so that pools can no longer be manipulated.

`-e` Enable the pools facility so that pools can be manipulated.

`-n` Validate the configuration without actually updating the current active configuration. Checks that there are no syntactic errors and that the configuration can be instantiated on the current system. No validation of application specific properties is performed.

`-s` Update the specified location with the details of the current dynamic configuration.

This option requires update permission for the configuration that you are going to update. If you use this option with the `-c` option, the dynamic configuration is updated before the static location.

`-x` Remove the currently active pool configuration. Destroy all defined resources, and return all formerly partitioned components to their default resources.

Operands The following operands are supported:

filename Use the configuration contained within this file.

Examples **EXAMPLE 1** Instantiating a Configuration

The following command instantiates the configuration contained at `/home/admin/newconfig`:

```
example# /usr/sbin/pooladm -c /home/admin/newconfig
```

EXAMPLE 2 Validating the Configuration Without Instantiating It

The following command attempts to instantiate the configuration contained at `/home/admin/newconfig`. It displays any error conditions that it encounters, but does not actually modify the active configuration.

EXAMPLE 2 Validating the Configuration Without Instantiating It *(Continued)*

```
example# /usr/sbin/pooladm -n -c /home/admin/newconfig
```

EXAMPLE 3 Removing the Current Configuration

The following command removes the current pool configuration:

```
example# /usr/sbin/pooladm -x
```

EXAMPLE 4 Enabling the Pools Facility

The following command enables the pool facility:

```
example# /usr/sbin/pooladm -e
```

EXAMPLE 5 Enabling the Pools Facility Using SMF

The following command enables the pool facility through use of the Service Management Facility. See `smf(5)`.

```
example# /usr/sbin/svcadm enable svc:/system/pools:default
```

EXAMPLE 6 Saving the Active Configuration to a Specified Location

The following command saves the active configuration to `/tmp/state.backup`:

```
example# /usr/sbin/pooladm -s /tmp/state.backup
```

Files `/etc/pooladm.conf` Configuration file for `pooladm`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpool
Interface Stability	See below.

The invocation is Evolving. The output is Unstable.

See Also `poolcfg(1M)`, `poolbind(1M)`, `psrset(1M)`, `svcadm(1M)`, `pset_destroy(2)`, `libpool(3LIB)`, `attributes(5)`, `smf(5)`

System Administration Guide: Solaris Containers-Resource Management and Solaris Zones

Notes Resource bindings that are not presented in the form of a binding to a partitionable resource, such as the scheduling class, are not necessarily modified in a `pooladm -x` operation.

The pools facility is not active by default when Solaris starts. `pooladm -e` explicitly activates the pools facility. The behavior of certain APIs related to processor partitioning and process binding are modified when pools is active. See `libpool(3LIB)`.

You cannot enable the pools facility on a system where processor sets have been created. Use the `psrset(1M)` command or `pset_destroy(2)` to destroy processor sets manually before you enable the pools facility.

Because the Resource Pools facility is an `smf(5)` service, it can also be enabled and disabled using the standard SMF interfaces.

Name poolbind – bind processes, tasks, or projects or query binding of processes to resource pools

Synopsis

```
/usr/sbin/poolbind  -p poolname [-i idtype] id...

/usr/sbin/poolbind  -q pid...

/usr/sbin/poolbind  -Q pid...
```

Description The `poolbind` command allows an authorized user to bind zones, projects, tasks, and processes to pools. It can also allow a user to query a process to determine which pool the process is bound to.

Options The following options are supported:

`-i` *idtype*
: This option, together with the *idlist* arguments, specifies one or more processes to which the poolbind command is to apply. The interpretation of *idlist* depends on the value of *idtype*. The valid *idtype* arguments and corresponding interpretations of *idlist* are as follows:

  pid
  : *idlist* is a list of process IDs. Binds the specified processes to the specified pool. This is the default behavior if no idtype is specified.

  taskid
  : *idlist* is a list of task IDs. Bind all processes within the list of task IDs to the specified pool.

  projid
  : *idlist* is a list of project IDs. Bind all processes within the list of projects to the specified pool. Each project ID can be specified as either a project name or a numerical project ID. See `project(4)`.

  zoneid
  : *idlist* is a list of zone IDs. Bind all processes within the list of zones to the specified pool. Each zone ID can be specified as either a zone name or a numerical zone ID. See `zones(5)`.

`-p` *poolname*
: Specifies the name of a pool to which the specified zone, project, tasks, or processes are to be bound.

`-q` *pid ...*
: Queries the pool bindings for a given list of process IDs. If the collection of resources associated with the process does not correspond to any currently existing pool, or if there are multiple pools with the set of resources that the process is bound to, the query fails for that particular process ID.

`-Q` *pid ...*
: Queries the resource bindings for a given list of process IDs. The resource bindings are each reported on a separate line.

Examples

EXAMPLE 1 Binding All Processes

The following command binds all processes in projects 5 and 7 to the pool `web_app`:

```
example# /usr/sbin/poolbind -p web_app -i projid 5 7
```

EXAMPLE 2 Binding the Running Shell

The following command binds the running shell to the pool `web_app`:

```
 example# /usr/sbin/poolbind -p web_app $$
```

EXAMPLE 3 Querying the Pool Bindings

The following command queries the bindings to verify that the shell is bound to the given pool:

```
example# /usr/sbin/poolbind -q $$
```

EXAMPLE 4 Querying the Resource Bindings

The following command queries the bindings to verify that the shell is bound to the given resources:

```
example# /usr/sbin/poolbind -Q $$
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` Requested operation could not be completed.

`2` Invalid command line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpool
Interface Stability	See below.

The invocation is Evolving. The output is Unstable.

See Also

`pooladm(1M)`, `poolcfg(1M)`, `libpool(3LIB)`, `project(4)`, `attributes(5)`, `zones(5)`

System Administration Guide: Solaris Containers-Resource Management and Solaris Zones

Name poolcfg – create and modify resource pool configuration files

Synopsis

```
/usr/sbin/poolcfg -c command [-d | [filename]]
/usr/sbin/poolcfg -f command_file [-d | [filename]]
/usr/sbin/poolcfg -h
```

Description The `poolcfg` utility provides configuration operations on pools and sets. These operations are performed upon an existing configuration and take the form of modifications to the specified configuration file. If you use the `-d` option, the modifications occur to the kernel state. Actual activation of the resulting configuration is achieved by way of the `pooladm(1M)` utility.

Pools configuration files are structured files that must have been constructed using `poolcfg` itself or `libpool(3LIB)` directly.

An invocation of `poolcfg` with the pool dynamic location and write permission will hang if the dynamic location has already been opened for writing.

The configurations which are created by this utility can be used by `pooladm` to instantiate the configuration upon a target host.

Options The following options are supported:

`-c` *command*	Specify *command* as an editing command. See `USAGE`.
`-d`	Operate directly on the kernel state. No *filename* is allowed.
`-f` *command_file*	Take the commands from *command_file*. *command_file* consists of editing commands, one per line.
`-h`	Display extended information about the syntax of editing commands.

Usage

Scripts A script consists of editing commands, one per line, of the following:

info [*entity-name*]
Display configuration (or specified portion) in human readable form to standard output. If no entity is specified, system information is displayed. Therefore, `poolcfg -c 'info' afile` is an equivalent invocation to `poolcfg -c 'info system name' afile`.

create *entity-name* [*property-list*]
Make an entity of the specified type and name.

destroy *entity-name*
Remove the specified entity.

modify *entity-name* [*property-list*]
Change the listed properties on the named entity.

associate *pool-name* [*resource-list*]
Connect one or more resources to a pool, or replace one or more existing connections.

transfer to [*resourcetype*] *name*[*component-list*]
Transfer one or more discrete components to a resource .

transfer [*quantity*] from [*resourcetype*] [*src*] to [*tgt*]
Transfer a resource quantity from *src* to *tgt*.

transfer [*quantity*] to [*resourcetype*] [*tgt*] from [*src*]
Transfer a resource quantity to *tgt* from *src*.

discover
Create a system entity, with one pool entity and resources to match current system configuration. All discovered resources of each resource type are recorded in the file, with the single pool referring to the default resource for each resource type.

This command is a NO-OP when `poolcfg` operates directly on the kernel. See the `-d` option.

You should avoid use of this command. The preferred method for creating a configuration is to export the dynamic configuration using `pooladm(1M)` with the `-s` option.

rename *entity-name* to *new-name*
Change the name of an entity on the system to its new name.

Property Lists The property list is specified by:

```
( proptype name = value [ ; proptype name = value ]* )
```

where the last definition in the sequence for a given proptype, name pair is the one that holds. For property deletion, use ~ `proptype name`.

Resource Lists A resource list is specified by:

```
( resourcetype name [ ; resourcetype name ]* )
```

where the last specification in the sequence for a resource is the one that holds. There is no deletion syntax for resource lists.

Component Lists A component list is specified by:

```
( componenttype name [ ; componenttype name ]* )
```

where the last specification in the sequence for a component is the one that holds. There is no deletion syntax for component lists.

Recognized Entities

system	Machine level entity
pool	Named collection of resource associations

Resource Types

pset	Processor set resource

Property Types

boolean	Takes one of two values true or false.
int	A 64–bit signed integer value.
uint	A 64–bit unsigned integer value.
string	Strings are delimited by quotes ("), and support the character escape sequences defined in `formats(5)`.
float	Scientific notation is not supported.

Examples

EXAMPLE 1 Writing a poolcfg Script

The following `poolcfg` script creates a pool named `Accounting`, and a processor set, `small-1`. The processor set is created first, then the pool is created and associated with the set.

```
create pset small-1 ( uint pset.min = 1 ; uint pset.max = 4)
create pool Accounting
associate pool Accounting ( pset small-1 )
```

EXAMPLE 2 Reporting on `pool_0`

The following command reports on `pool_0` to standard output in human readable form:

```
 # poolcfg -c 'info pool pool_0' /etc/pooladm.conf
```

EXAMPLE 3 Destroying pool_0 and Its Associations

The following command destroys `pool_0` and associations, but not the formerly associated resources:

```
# poolcfg -c 'destroy pool pool_0' /etc/pooladm.conf
```

EXAMPLE 4 Displaying the Current Configuration

The following command displays the current configuration:

```
$ poolcfg -c 'info' /etc/pooladm.conf
system example_system
               int system.version 1
               boolean system.bind-default true
               string system.comment Discovered by libpool

               pool pool_default
                    boolean pool.default true
                    boolean pool.active true
                    int pool.importance 5
                    string pool.comment
                    string.pool.scheduler FSS
                    pset pset_default

               pset pset_default
```

EXAMPLE 4 Displaying the Current Configuration *(Continued)*

```
int pset.sys_id -1
string pset.units population
boolean pset.default true
uint pset.max 4294967295
uint pset.min 1
string pset.comment
boolean pset.escapable false
uint pset.load 0
uint pset.size 2

cpu
     int cpu.sys_id 0
     string cpu.comment

cpu
     int cpu.sys_id 2
     string cpu.comment
```

EXAMPLE 5 Moving cpu with ID 2 to Processor Set pset1 in the Kernel

The following command moves `cpu` with ID `2` to processor set `pset1` in the kernel:

```
# poolcfg -dc 'transfer to pset pset1 ( cpu 2 )'
```

EXAMPLE 6 Moving 2 cpus from Processor Set pset1 to Processor Set pset2 in the Kernel

The following command moves 2 `cpus` from processor set `pset1` to processor set `pset2` in the kernel:

```
# poolcfg -dc 'transfer 2 from pset pset1 to pset2'
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpool
Interface Stability	See below.

The invocation is Committed. The output is Uncommitted.

See Also

`pooladm(1M)`, `poolbind(1M)`, `libpool(3LIB)`, `attributes(5)`, `formats(5)`

System Administration Guide: Solaris Containers-Resource Management and Solaris Zones

Name poold – automated resource pools partitioning daemon

Synopsis `poold [-l level]`

Description `poold` provides automated resource partitioning facilities. Normally, `poold` is active on the system whenever the pools facility is active. `poold` starts and stops when the `pool_set_status(3POOL)` function activates or deactivates the pools facility. `poold` starts when you activate pools and stops when you deactivate pools. If you manually stop `poold` by using a utility such as `kill(1)`, you can invoke it manually.

`poold`'s configuration details are held in a `libpool(3LIB)` configuration and you can access all customizable behavior from this configuration.

`poold` periodically examines the load on the system and decides whether intervention is required to maintain optimal system performance with respect to resource consumption. `poold` also responds to externally initiated (with respect to `poold`) changes of either resource configuration or objectives.

If intervention is required, `poold` attempts to reallocate the available resources to ensure that performance objectives are satisfied. If it is not possible for `poold` to meet performance objectives with the available resources, then a message is written to the log. `poold` allocates scarce resources according to the objectives configured by the administrator. The system administrator must determine which resource pools are most deserving of scarce resource and indicate this through the importance of resource pools and objectives.

Options The following options are supported:

`-l` *level* Specify the vebosity level for logging information.

Specify *level* as `ALERT`, `CRIT`, `ERR`, `WARNING`, `NOTICE`, `INFO`, and `DEBUG`. If *level* is not supplied, then the default logging level is `INFO`.

`ALERT`	A condition that should be corrected immediately, such as a corrupted system database.
`CRIT`	Critical conditions, such as hard device errors.
`ERR`	Errors.
`WARNING`	Warning messages.
`NOTICE`	Conditions that are not error conditions, but that may require special handling.
`INFO`	Informational messages.
`DEBUG`	Messages that contain information normally of use only when debugging a program.

When invoked manually, with the `-l` option, all log output is directed to standard error.

Examples

EXAMPLE 1 Modifying the Default Logging Level

The following command modifies the default logging level to ERR:

```
# /usr/lib/pool/poold -l ERR
```

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpool
Interface Stability	See below.

The invocation is Evolving. The output is Unstable.

See Also

pooladm(1M), poolbind(1M), poolcfg(1M), poolstat(1M), pool_set_status(3POOL), libpool(3LIB), attributes(5)

System Administration Guide: Solaris Containers-Resource Management and Solaris Zones

Name poolstat – report active pool statistics

Synopsis

```
poolstat [-p pool-list] [-r rset-list] [interval [count]]

poolstat [-p pool-list] [-o format -r rset-list]
     [interval [count]]
```

Description The `poolstat` utility iteratively examines all active pools on the system. It reports statistics based on the selected output mode. `poolstat` provides options to examine only specified pools and report resource set-specific statistics.

Without options, `poolstat` examines all pools, reports basic statistics for their resource sets, and exits.

DISPLAY FORMATS In default output format, `poolstat` outputs a header line and a line for each pool. The line begins with the pool ID and its name, followed by a column of statistical data for the processor set attached to the pool.

The columns are defined as follows:

`id` Pool ID.

`pool` Pool name.

`rid` Resource set id.

`rset` Resource set name.

`type` Resource set type.

`min` Minimum resource set size.

`max` Maximum resource set size.

`size` Current resource set size.

`used` The measure of how much of the resource set is currently is use. This is calculated as the percentage utilization of the resource set multiplied by its size. If resource set has been reconfigured during last sampling interval, this value might be not reported (`-`).

`load` The absolute representation of the load that is put on the resource set. For the definition of this property see `libpool(3LIB)`.

Options The following options are supported:

`-o` *format* Report statistics according to the format specification given in format. See `DISPLAY FORMATS`.

The `-o` option accepts lists as arguments. Items in a list can be either separated by commas or enclosed in quotes and separated by commas or spaces.

You can specify multiple `-o` options. The format specification is interpreted as the whitespace separated concatenation of all the format option arguments.

The `-o` option must be used in conjunction with the `-r` option.

`-p` *pool-list* Report only pools whose names are in the given list. If the -r option is also used, this option selects only resource sets which belong to pools in the given list. Statistics for pools or resource sets are reported in the same order in which pool names are listed on the pool-list. Pool can be specified by name or by ID.

The `-p` option accepts lists as arguments. Items in a *pool-list* can only be separated by spaces.

`-r` *rset-list* Report resource set statistics. If the *rset-list* argument is "`all`", then all possible resource set types are selected.

The `-r` option accepts lists as arguments. Items in a list can be either separated by commas or enclosed in quotes and separated by commas or spaces.

The following resource set types are supported:

`all` All resource set types

`pset` Processor set

Operands The following operands are supported:

count The number of times that the statistics are repeated. By default, poolstat reports statistics only once.

If neither interval nor count are specified, statistics are reported once. If interval is specified and count is not, statistics are reported indefinitely.

interval The sampling interval in seconds.

If neither interval nor count are specified, statistics are reported once. If interval is specified and count is not, statistics are reported indefinitely.

Examples **EXAMPLE 1** Using `poolstat`

The following example shows the default output from the `poolstat` utility:

```
% poolstat
                          pset
  id pool           size used load
   0 pool_default      4  3.6  6.2
   1 pool_admin        4  3.3  8.4
```

EXAMPLE 2 Reporting Resource Set Statistics

The following example reports resource set statistics.

```
% poolstat -r pset
```

EXAMPLE 2 Reporting Resource Set Statistics *(Continued)*

```
  id pool           type rid rset          min  max size used load
  0 pool_default    pset  -1 pset_default    1  65K    2  1.2  8.3
  1 pool_admin      pset   1 pset_admin      1    1    1  0.4  5.2
  2 pool_other      pset  -1 pset_default    1  65K    2  1.2  8.3
```

Resource sets attached to multiple pools, as `pset_default` in the example above, are listed multiple times, once for each pool.

EXAMPLE 3 Restricting the Output to the List of Pools

The following example restricts the output to the list of pools

```
% poolstat -p pool_default
                     pset
 id pool         size used load
 0 pool_default    8  5.3 10.3

% poolstat -p 'pool_admin pool_default'
                       pset
  id pool         size used load
  1 pool_admin      6  4.3  5.3
  0 pool_default    2  1.9  2.0

% poolstat -r all -p 'pool_admin pool_default'
  id pool            type rid rset          min  max size used load
  1 pool_admin      pset   1 pset_admin      1    1    1  0.9  2.3
  2 pool_default    pset  -1 pset_default    1  65K    2  2.0  2.0
```

EXAMPLE 4 Customizing Output

The following example customizes output:

```
% poolstat -r -o pool,rset,size,load
  pool            rset          size load
  pool_default    pset_default     4  4.5
  pool_admin      pset_admin       4  2.1
```

Exit Status

The following exit values are returned:

`0` Successful completion.

1 An error occurred.

2 Invalid command line options were specified.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpool
Stability	Evolving

See Also libpool(3LIB), attributes(5)

System Administration Guide: Solaris Containers-Resource Management and Solaris Zones

Notes The system ids associated with resources can change after the system reboots or the resource configuration is altered.

Name ports – creates /dev entries and inittab entries for serial lines

Synopsis `/usr/sbin/ports [-r` *rootdir*`]`

Description `devfsadm(1M)` is now the preferred command for `/dev` and `/devices` and should be used instead of `ports`.

The `ports` command creates symbolic links in the `/dev/term` and `/dev/cua` directories to the serial-port character device files in `/devices` and adds new entries in `/etc/inittab` for non-system ports found. System-board ports are given single lower-case letters for names (such as `a` and `b`) while other ports are named numerically.

`ports` searches the kernel device tree to find the serial devices attached to the system. It also checks `/dev/term` and `/dev/cua` to see what symbolic links to serial devices already exist. `ports` then performs the following:

1. Assigns new numbers (or letters for system-board ports) to ports that are attached to the system but do not have `/dev/term` and `/dev/cua` entries. The numbers or letters assigned are the lowest-unused numbers or letters.
2. Removes dangling links: links from `/dev/term` and `/dev/cua` pointing to no-longer-existing ports.
3. Creates new `/dev/term` and `/dev/cua` links for new serial devices.
4. Invokes `sacadm(1M)` to make new port monitor entries for the new devices. This is not done automatically for on-board ports; on workstations these ports are often not used for dial-in sessions, so a port-monitor for one of these ports must be created explicitly.

If the configuration has not changed, ports exits without doing anything.

Notice to Driver Writers

`ports` considers devices with a node type of `DDI_NT_SERIAL`, `DDI_NT_SERIAL_MB`, `DDI_NT_SERIAL_DO`, or `DDI_NT_SERIAL_MB_DO` to be serial port devices. Devices with one of these node types must create minor device names that obey the following conventions when calling `ddi_create_minor_node(9F)`.

- The minor name for non-system port devices (`DDI_NT_SERIAL`) consists of an ASCII numeric string, where the first port on the device is named `0`, the second named `1`, the third named `2`, up to the number of ports provided by the device.
- The minor name for non-system dialout devices (`DDI_NT_SERIAL_DO`) is the ASCII numeric port name, concatenated with `,cu`. For example, the minor name for the first dialout port on the serial board is `0,cu`.
- The minor name for system-board port devices (`DDI_NT_SERIAL_MB`) consists of a string containing a single ASCII lowercase character, where the first port on the device is named `a`, the second is named `b`, the third is named `c`, for all ports on the device (or up through port `z`).
- The minor name for system-board dialout devices (`DDI_NT_SERIAL_MB_DO`) consists of the lowercase character port name, concatenated with `,cu`. For example, the minor name for the first dialout port on the on-board serial device is `a,cu`.

To prevent `disks` from attempting to automatically generate links for a device, drivers must specify a private node type and refrain from using one of the above node types when calling `ddi_create_minor_node(9F)`.

Options The following options are supported:

`-r` *rootdir* Causes `ports` to presume that the `/dev/term`, `/dev/cua`, and `/devices` directories are found under *rootdir*, not directly under `/`. If this argument is specified, `sacadm(1M)` is not invoked, since it would update terminal administration files under `/etc` without regard to the *rootdir*.

Examples **EXAMPLE 1** Creating the Serial and Dialout Minor Device Nodes

The following example creates the serial and dialout minor device nodes from the `xkserial` driver's `attach(9E)` function:

```
/*
* Create the minor number by combining the instance number
* with the port number.
*/ #define XKNUMPORTS           8
#define XKMINORNUM(i, p)     ((i) << 4 | (p))
#define XKMINORNUM_DO(i, p)  ((i) << 4 | (p) | 0x80)
int
xkserialattach(dev_info_t *dip, ddi_attach_cmd_t cmd)
{
   int instance, portnum;
   char name[8];
       /* other stuff in attach... */
   instance = ddi_get_instance(dip);
   for (portnum = 0; portnum < XKNUMPORTS; portnum++) {
       /*
        * create the serial port device
        */
        sprintf(name, "%d", portnum);
        ddi_create_minor_node(dip, name, S_IFCHR,
           XKMINORNUM(instance, portnum), DDI_NT_SERIAL, 0);

        /*
         * create the dialout device
         */
         sprintf(name,"%d,cu", portnum);
         ddi_create_minor_node(dip, name, S_IFCHR,
             XKMINORNUM_DO(instance, portnum), DDI_NT_SERIAL_DO, 0);
     }
}
```

EXAMPLE 2 Installing the `xkserial` Port Driver on a Sun Fire 4800

The following example installs the `xkserial` port driver on a Sun Fire 4800 (with the driver controlling the fictional XKSerial 8 port serial board), with these special files in `/devices`:

```
# ls -l /devices/ssm@0,0/pci@18,700000/pci@1/xkserial@f,800000/
crw-r-----   1 root sys   32,  16 Aug 29 00:02 xkserial@2000:0
crw-r-----   1 root sys   32, 144 Aug 29 00:02 xkserial@2000:0,cu
crw-r-----   1 root sys   32,  17 Aug 29 00:02 xkserial@2000:1
crw-r-----   1 root sys   32, 145 Aug 29 00:02 xkserial@2000:1,cu
crw-r-----   1 root sys   32,  18 Aug 29 00:02 xkserial@2000:2
crw-r-----   1 root sys   32, 146 Aug 29 00:02 xkserial@2000:2,cu
crw-r-----   1 root sys   32,  19 Aug 29 00:02 xkserial@2000:3
crw-r-----   1 root sys   32, 147 Aug 29 00:02 xkserial@2000:3,cu
crw-r-----   1 root sys   32,  20 Aug 29 00:02 xkserial@2000:4
crw-r-----   1 root sys   32, 148 Aug 29 00:02 xkserial@2000:4,cu
crw-r-----   1 root sys   32,  21 Aug 29 00:02 xkserial@2000:5
crw-r-----   1 root sys   32, 149 Aug 29 00:02 xkserial@2000:5,cu
crw-r-----   1 root sys   32,  22 Aug 29 00:02 xkserial@2000:6
crw-r-----   1 root sys   32, 150 Aug 29 00:02 xkserial@2000:6,cu
crw-r-----   1 root sys   32,  23 Aug 29 00:02 xkserial@2000:7
crw-r-----   1 root sys   32, 151 Aug 29 00:02 xkserial@2000:7,cu
```

`/dev/term` contain symbolic links to the serial port device nodes in `/devices`

```
# ls -l /dev/term
/dev/term/0 -> ../../devices/[....]/xkserial@2000:0
/dev/term/1 -> ../../devices/[....]/xkserial@2000:1
/dev/term/2 -> ../../devices/[....]/xkserial@2000:2
/dev/term/3 -> ../../devices/[....]/xkserial@2000:3
/dev/term/4 -> ../../devices/[....]/xkserial@2000:4
/dev/term/5 -> ../../devices/[....]/xkserial@2000:5
/dev/term/6 -> ../../devices/[....]/xkserial@2000:6
/dev/term/7 -> ../../devices/[....]/xkserial@2000:7
```

and `/dev/cua` contain symbolic links to the dialout port device nodes in `/devices`

```
# ls -l /dev/cua

/dev/cua/0 -> ../../devices/[....]/xkserial@2000:0,cu
/dev/cua/1 -> ../../devices/[....]/xkserial@2000:1,cu
/dev/cua/2 -> ../../devices/[....]/xkserial@2000:2,cu
/dev/cua/3 -> ../../devices/[....]/xkserial@2000:3,cu
/dev/cua/4 -> ../../devices/[....]/xkserial@2000:4,cu
/dev/cua/5 -> ../../devices/[....]/xkserial@2000:5,cu
/dev/cua/6 -> ../../devices/[....]/xkserial@2000:6,cu
/dev/cua/7 -> ../../devices/[....]/xkserial@2000:7,cu
```

Files

`/dev/term/`*n*	Logical serial port devices
`/dev/cua/`*n*	Logical dialout port devices
`/etc/inittab`	
`/etc/saf/*`	

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `add_drv(1M)`, `devfsadm(1M)`, `drvconfig(1M)`, `pmadm(1M)`, `sacadm(1M)`, `attributes(5)`, `devfs(7FS)`, `attach(9E)`, `ddi_create_minor_node(9F)`

Writing Device Drivers

Name powerd – power manager daemon

Synopsis

```
/usr/lib/power/powerd [-n]
```

Description The `powerd` daemon is started by `pmconfig(1M)` to monitor system activity and perform an automatic shutdown using the suspend-resume feature. When the system is suspended, complete current state is saved on the disk before power is removed. On reboot, the system automatically starts a resume operation and the system is restored to the same state it was in immediately prior to suspend.

Immediately prior to system shutdown, the daemon notifies `syslogd(1M)` of the shutdown, which broadcasts a notification.

Options The following option is supported:

`-n` No broadcast mode. The daemon silently shuts down the system without notifying `syslogd(1M)`.

Files `/etc/power.conf` Power Management configuration information file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpmu
Interface Stability	Unstable

See Also `pmconfig(1M)`, `dtpower(1M)`, `syslogd(1M)`, `power.conf(4)`, `attributes(5)`, `cpr(7)`, `pm(7D)`

Using Power Management

Name ppdmgr – utility for managing PPD files

Synopsis

```
ppdmgr -a ppd_filename_path [-L label] [-R ppd_repository_name] [-w]

ppdmgr -r [-L label] [-R ppd_repository_name]

ppdmgr -u [-L label] [-R ppd_repository_name]
```

Description The PPD File Manager adds PPD files to the print system's PPD file repositories. When changes in the PPD file repositories are made by the utility, they are reflected in the Solaris Print Manager (see `printmgr(1M)`) GUI printer driver field when adding or modifying local attached or network attached printers.

Multiple PPD file repositories may be available. PPD files that are supplied with the system are available in the "`system`" repository. PPD files that are supplied by third party vendors may be available in the "`vendor`" repository. PPD files that are supplied by system administrators may be available in the "`admin`" repository, and PPD files that are added using this utility are available in the "user" repository. When this utility is used to update or rebuild printer information reflected in the `printmgr(1M)`, the special reserved repository name "`all`" can be used to specify all of the available PPD file repositories.

PPD files are grouped under a user-provided "label" within the print system's PPD file repository to allow easier selection of a printer driver through the `printmgr` GUI. The special reserved label name "`all`" can be used to specify all of the available labels in a PPD file repository. The results are undefined if a label name contains characters that are not in the portable filename character set or if a label name starts with a hyphen (-). A label name may not contain a semi-colon (:).

Options The following subcommand are supported:

`-a` *ppd_file_path*
: Adds a copy of *ppd_file_path* to the system under the specified label in the specified repository, where *ppd_file_path* is the full path and file name to the PPD file. The utility accepts a valid PPD file with either a `.ppd` or `.ppd.gz` (gzipped) extension.

`-L` *label*
: Specifies a label to group PPD files under. When specified with the `-a` option, the default label is "`user`". When specified with the `-r` or `-u` option, the default label is "`all`". The following are reserved labels:

`caches`	may never be specified
`ppdcache`	may never be specified
`manufaliases`	may never be specified
`all`	applies the specified action to all labels in a repository, and may only be specified with the `-r` or `-u` option.
`SUNW*`	anything starting with `SUNW` is reserved and should not be specified with the `-a` option.

`-R` *repository_name*
Specifies the name of a PPD file repository representing one of the PPD file installation locations. Possible repository names include: "`user`", "`admin`", "`vendor`", "`system`", and "`all`".

The repository name "`all`" signifies all of the possible repository names. The default repository name is "`user`". Only the "`user`" or "`admin`" repository may be specified with the `-a` option.

`-r`
Rebuilds the cache information for the specified label in the specified repository.

`-u`
Updates the cache information for the specified label in the specified repository.

`-w`
Write the full path of the added PPD file. This option is only valid with the `-a` option. Otherwise, this option is ignored.

Examples

EXAMPLE 1 Adding a copy of a PPD file to the repository

The following commands add a copy of a PPD file to the "`user`" repository under the "`user`" label:

```
# ppdmgr -a /net/somesystem/ppdfiles/ppdfile.ppd
```

or

```
# ppdmgr -a /net/somesystem/ppdfiles/ppdfile.ppd -L user
```

The following command adds a copy of a PPD file to the "`user`" repository under the "`Photo`" label and write the full path of the added copy to standard output:

```
# ppdmgr -a /net/somesystem/ppdfiles/ppdfile.ppd -L "Photo" -w
```

EXAMPLE 2 Updating the cache for the PPD files

The following commands update the cache for the PPD files under the "`all`" labels in the "`user`" repository:

```
 # ppdmgr -u
```

or

```
 # ppdmgr -u -R user -L all
```

The following command updates the cache for the PPD files under the "photo" label in the "`user`" repository:

```
 # ppdmgr -u -R user -L Photo
```

EXAMPLE 3 Rebuilding the cache for the PPD files

The following command rebuilds the cache for the PPD files under the "Photo" label in the "user" repository:

```
# ppdmgr -r -R user -L Photo
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpsr
Interface Stability	Committed

See Also

printmgr(1M), attributes(5)

Name pppd – point to point protocol daemon

Synopsis `pppd [`*tty_name*`] [`*speed*`] [`*options*`]`

Description The point-to-point protocol (PPP) provides a method for transmitting datagrams over serial point-to-point links. PPP is composed of three components: a facility for encapsulating datagrams over serial links, an extensible link control protocol (LCP), and a family of network control protocols (NCP) for establishing and configuring different network-layer protocols.

The encapsulation scheme is provided by driver code in the kernel. `pppd` provides the basic LCP authentication support and several NCPs for establishing and configuring the Internet Protocol (referred to as the IP Control Protocol or "IPCP") and IPv6 (IPV6CP).

Options The following sections discuss the `pppd` options:

Options Files Options are taken from files and the command line. `pppd` reads options from the files `/etc/ppp/options`, `$HOME/.ppprc` and `/etc/ppp/options.`*ttyname* (in that order) before processing the options on the command line. (Command-line options are scanned for the terminal name before the `options.`*ttyname* file is read.) To form the name of the `options.`*ttyname* file, the initial `/dev/` is removed from the terminal name, and any remaining forward slash characters (/) are replaced with dots. For example, with serial device `/dev/cua/a`, option file `/etc/ppp/options.cua.a` is read.

An options file is parsed into a series of words that are delimited by whitespace. Whitespace can be included in a word by enclosing the word in double-quotes ("). A backslash (\) quotes the succeeding character. A hash (#) starts a comment, which continues until the end of the line. There is no restriction on using the `file` or `call` options within an options file.

Frequently Used Options

`<tty_name>` Communicate over the named device. The string `/dev/` is prepended if necessary. If no device name is given, or if the name of the terminal connected to the standard input is given, `pppd` uses that terminal and does not fork to put itself in the background. A value for this option from a privileged source cannot be overridden by a non-privileged user.

`<speed>` Set the baud rate to `<speed>` (a decimal number). The default is to leave the baud rate unchanged. This option is normally needed for dial-out only.

`asyncmap` <*map*> Set the `async` character map to <*map*>. The map describes which control characters cannot be successfully received over the serial line. `pppd` asks the peer to send these characters as a 2-byte escape sequence. The argument is a 32 bit hex number, with each bit representing a character to escape. Bit 0 (00000001) represents the character 0x00; bit 31 (80000000) represents the character 0x1f or ^_. If multiple `asyncmap` options are given, the values are `ORed` together. If no `asyncmap` option is given, `pppd` attempts to negotiate a value of 0. If the

peer agrees, this disables escaping of the standard control characters. Use the `default-asyncmap` option to disable negotiation and escape all control characters.

`auth` — Require the peer to authenticate itself before allowing network packets to be sent or received. This option is the default if the system has a default route. If the `auth` or the `noauth` option is not specified, `pppd` allows the peer to use only those IP addresses to which the system does not already have a route.

`call` *name* — Read options from the file `/etc/ppp/peers/`*name*. This file may contain privileged options, including `noauth`, even if `pppd` is not being run by root. The *name* string may not begin with a slash ("/") or include consecutive periods (`".."`) as a pathname component.

`callback` *number* — Request a callback to the given telephone number using Microsoft CBCP.

`connect` *script* — Use the executable or shell command specified by *script* to set up the serial line. This script would typically use the `chat(1M)` program to dial the modem and start the remote `PPP` session. A value for this option originating from a privileged source cannot be overridden by a non-privileged user.

`crtscts` — Use hardware flow control, that is, RTS/CTS, to control the flow of data on the serial port. If the `crtscts`, `nocrtscts`, `cdtrcts` or `nocdtrcts` option is not provided, the hardware flow control setting for the serial port is left unchanged. Some serial ports lack a true RTS output and use this mode to implement unidirectional flow control. The serial port suspends transmission when requested by the modem by means of CTS but cannot request the modem to stop sending to the computer. This mode allows the use of DTR as a modem control line.

`defaultroute` — Add a default route to the system routing tables when IPCP negotiation successfully completes, using the peer as the gateway. This entry is removed when the `PPP` connection is broken. This option is privileged if the `nodefaultroute` option is specified.

`disconnect` *script* — Run the executable or shell command specified by *script* after `pppd` terminates the link. Typically, this script is used to command the modem to hang up if hardware modem control signals are not available. `disconnect` is not run if the modem has already hung up. A value for this option originating from a privileged source cannot be overridden by a non-privileged user.

`escape` *xx,yy,...* — Specifies that certain characters be escaped on transmission regardless of whether the peer requests them to be escaped with its `async` control

character map. The characters to be escaped are specified as a list of hex numbers separated by commas. Note that almost any character can be specified for the `escape` option, unlike the `asyncmap` option which allows only control characters to be specified. Characters that cannot be escaped are those containing hex values 0x20 through 0x3f and 0x5e.

`file` *name* — Read options from file *name*. If this option is used on the command line or in `$HOME/.pppprc`, the file must be readable by the user invoking `pppd`. See Options Files for a list of files that `pppd` always reads, regardless of the use of this option.

`init` *script* — Run the executable or shell command specified by *script* to initialize the serial line. This script would typically use the `chat(1M)` program to configure the modem to enable auto-answer. A value for this option from a privileged source cannot be overridden by a non-privileged user.

`lock` — Directs `pppd` to create a UUCP-style lock file for the serial device to ensure exclusive access to the device.

`mru` *n* — Set the Maximum Receive Unit (MRU) value to *n*. `pppd` asks the peer to send packets of no more than *n* bytes. Minimum MRU value is 128. Default MRU value is 1500. A value of 296 is recommended for slow links (40 bytes for TCP/IP header + 256 bytes of data). For IPv6, MRU must be at least 1280.

`mtu` *n* — Set the Maximum Transmit Unit (MTU) value to *n*. Unless the peer requests a smaller value via MRU negotiation, `pppd` requests the kernel networking code to send data packets of no more than *n* bytes through the PPP network interface. For IPv6, MTU must be at least 1280.

`passive` — Enables the "passive" option in the LCP. With this option, `pppd` attempts to initiate a connection; if no reply is received from the peer, `pppd` waits passively for a valid LCP packet instead of exiting, as it would without this option.

Options `<local_IP_address>:<remote_IP_address>`

Set the local and/or remote interface IP addresses. Either one may be omitted, but the colon is required. The IP addresses are specified with a host name or in decimal dot notation, for example: `:10.1.2.3`. The default local address is the first IP address of the system unless the `noipdefault` option is provided. The remote address is obtained from the peer if not specified in any option. Thus, in simple cases, this option is not required. If a local and/or remote IP address is specified with this option, `pppd` will not accept a different value from the peer in the IPCP negotiation unless the `ipcp-accept-local` and/or `ipcp-accept-remote` options are given, respectively.

`allow-fcs` *fcs-type*
Set allowable FCS type(s) for data sent to the peer. The *fcs-type* is a comma-separated list of "crc16", "crc32", "null", or integers. By default, all known types are allowed. If this option is specified and the peer requests a type not listed, a LCP Configure-Nak is sent to request only the listed types.

`allow-ip` *address(es)*
Allow peers to use the given IP address or subnet without authenticating themselves. The parameter is parsed in the same manner as each element of the list of allowed IP addresses is parsed in the secrets files. See the Authentication section more more details.

`bsdcomp` *nr,nt*
Request that the peer compress packets that it sends using the BSD-Compress scheme, with a maximum code size of *nr* bits, and agree to compress packets sent to the peer with a maximum code size of *nt* bits. If *nt* is not specified, it defaults to the value given for *nr*. Values in the range 9 to 15 may be used for *nr* and *nt*; larger values provide better compression but consume more kernel memory for compression dictionaries. Alternatively, a value of 0 for *nr* or *nt* disables compression in the corresponding direction. Use `nobsdcomp` or `bsdcomp 0` to disable BSD-Compress compression entirely. If this option is read from a privileged source, a nonprivileged user may not specify a code size larger than the value from the privileged source.

`cdtrcts`
Use a non-standard hardware flow control such as DTR/CTS to control the flow of data on the serial port. If the `crtscts`, `nocrtscts`, `cdtrcts` or `nocdtrcts` option is not specified, the hardware flow control setting for the serial port is left unchanged. Some serial ports lack a true RTS output. Such serial ports use this mode to implement true bi-directional flow control. Note that this flow control mode does not permit using DTR as a modem control line.

`chap-interval` *n*
If this option is given, `pppd` will rechallenge the peer every *n* seconds.

`chap-max-challenge` *n*
Set the maximum number of CHAP challenge transmissions to *n* (default 10).

`chap-restart` *n*
Set the CHAP restart interval (retransmission timeout for challenges) to *n* seconds. The default is 3.

`connect-delay` *n*
Wait for up to *n* milliseconds after the connect script finishes for a valid PPP packet from the peer. When the wait period elapses or when a valid PPP packet is received from the peer, `pppd` begins negotiation by sending its first LCP packet. The default value is 1000 (1 second). A wait period applies only if the `connect` or `pty` option is used.

`datarate` *n*
Set maximum data rate to *n* (in bytes per second) when using the `pty`, `notty`, `record`, or `socket` options.

`debug`
Enables connection debugging facilities. If this option is given, `pppd` logs the contents of all control packets sent or received in a readable form. The packets are logged through syslog with facility `daemon` and level `debug`. This information can be directed to a file by configuring `/etc/syslog.conf` appropriately.

`default-asyncmap`
Disable `asyncmap` negotiation, forcing all control characters to be escaped for both the transmit and the receive direction.

`default-fcs`
Disable FCS Alternatives negotiation entirely. By default, no FCS Alternatives option is sent to the peer, but the option is accepted. If this option is specified by the peer, then LCP Configure-Reject is sent.

`default-mru`
Disable MRU [Maximum Receive Unit] negotiation. With this option, `pppd` uses the default MRU value of 1500 bytes for the transmit and receive directions.

`deflate` *nr,nt,e*
Request that the peer compress packets that it sends, using the `deflate` scheme, with a maximum window size of *2**nr* bytes, and agree to compress packets sent to the peer with a maximum window size of *2**nt* bytes and effort level of *e* (1 to 9). If *nt* is not specified, it defaults to the value given for *nr*. If *e* is not specified, it defaults to 6. Values in the range 9 to 15 may be used for *nr* and *nt*; larger values provide better compression but consume more kernel memory for compression dictionaries. (Value 8 is not permitted due to a zlib bug.) Alternatively, a value of 0 for *nr* or *nt* disables compression in the corresponding direction. Use `nodeflate` or `deflate 0` to disable `deflate` compression entirely. (Note: `pppd` requests deflate compression in preference to BSD-Compress if the peer can do either.) If this option is read from a privileged source, a nonprivileged user may not specify a code size larger than the value from the privileged source.

`demand`
Initiate the link only on demand, that is, when data traffic is present. With this option, the remote IP address must be specified by the user on the command line or in an options file. `pppd` initially configures and enables the interface for IP traffic without connecting to the peer. When traffic is available, `pppd` connects to the peer and performs negotiation, authentication and other actions. When completed, `pppd` passes data packets across the link. The `demand` option implies the `persist` option. If this behavior is not desired, use the `nopersist` option after the `demand` option. The `idle` and `holdoff` options can be used in conjunction with the `demand` option.

`domain` *d*
Append the domain name *d* to the local host name for authentication purposes. For example, if `gethostname()` returns the name `porsche`, but the fully qualified domain name is `porsche.Quotron.COM`, you could specify `domain Quotron.COM`. With this configuration, `pppd` uses the name `porsche.Quotron.COM` for accessing secrets in the secrets file and as the default name when authenticating to the peer. This option is privileged.

`endpoint` *endpoint-value*
Set the endpoint discriminator (normally used for RFC 1990 Multilink PPP operation). The *endpoint-value* consists of a class identifier and a class-dependent value. The class identifier is one of "null," "local," "IP," "MAC," "magic," "phone," or a decimal integer. If present, the class-dependent value is separated from the identifier by a colon (":") or period (".") . This value may be a standard dotted-decimal IP address for class "IP," an optionally colon-or-dot separated hex Ethernet address for class "MAC" (must have 6 numbers), or an arbitrary string of bytes specified in hex with optional colon or dot separators between bytes. Although this option is available, this implementation does not support multilink.

`fcs` *fcs-type*
Set FCS type(s) desired for data sent by the peer. The *fcs-type* is a comma-separated list of `crc16`, `crc32`, `null`, or integers. By default, an FCS Alternatives option is not specified, and the medium-dependent FCS type is used. If this option is specified and the peer sends an LCP Configure-Nak, only the listed types are used. If none are in common, the FCS Alternatives option is omitted from the next LCP Configure-Request to drop back to the default.

`hide-password`
When logging the contents of PAP packets, this option causes `pppd` to exclude the password string from the log. This is the default.

`holdoff` *n*
Specifies how many seconds to wait before re-initiating the link after it terminates. This option is effective only if the `persist` or `demand` option is used. The holdoff period is not applied if the link is terminated because it was idle.

`ident` *string*
Set the LCP Identification string. The default value is a version string similar to that displayed by the `--version` option.

`idle` *n*
Specifies that `pppd` must disconnect if the link is idle for *n* seconds. The link is idle when no data packets (i.e. IP packets) are being sent or received. Do not use this option with the `persist` option but without the `demand` option.

`ipcp-accept-local`
With this option, `pppd` accepts the peer's idea of the local IP address, even if the local IP address is specified in an option.

`ipcp-accept-remote`
With this option, `pppd` accepts the peer's idea of its remote IP address, even if the remote IP address is specified in an option.

`ipcp-max-configure` *n*
Set the maximum number of IPCP Configure-Request transmissions to *n* (default 10).

`ipcp-max-failure` *n*
Set the maximum number of IPCP Configure-NAKs sent before sending Configure-Rejects instead to *n* (default 10).

`ipcp-max-terminate` *n*
Set the maximum number of IPCP terminate-request transmissions to *n* (default 3).

`ipcp-restart` *n*
Set the IPCP restart interval (retransmission timeout) to *n* seconds (default 3).

`ipparam` *string*
Provides an extra parameter to the ip-up and ip-down scripts. When this option is given, the *string* supplied is given as the sixth parameter to those scripts. See the Scripts section.

`ipv6` <*local_interface_identifier*>,<*remote_interface_identifier*>
Set the local and/or remote 64-bit interface identifier. Either one may be omitted. The identifier must be specified in standard ASCII notation of IPv6 addresses (for example: `::dead:beef`). If the `ipv6cp-use-ipaddr` option is given, the local and remote identifiers are derived from the respective IPv4 addresses (see above). The `ipv6cp-use-persistent` option can be used instead of the `ipv6 <local>,<remote>` option.

`ipv6cp-accept-local`
Accept peer's interface identifier for the local link identifier.

`ipv6cp-max-configure` *n*
Set the maximum number of IPv6CP Configure-Request transmissions to *n* (default 10).

`ipv6cp-max-failure` *n*
Set the maximum number of IPv6CP Configure-NAKs sent before sending Configure-Rejects instead to *n* (default 10).

`ipv6cp-max-terminate` *n*
Set the maximum number of IPv6CP terminate-request transmissions to *n* (default 3).

`ipv6cp-restart` *n*
Set the IPv6CP restart interval (retransmission timeout) to *n* seconds (default 3).

`ipv6cp-use-ipaddr`
If either the local or remote IPv6 address is unspecified, use the corresponding configured IPv4 address as a default interface identifier. (This option uses the configured addresses, not the negotiated addresses. Do not use it with `ipcp-accept-local` if the local IPv6 identifier is unspecified or with `ipcp-accept-remote` if the remote IPv6 identifier is unspecified.)

`ipv6cp-use-persistent`
Use uniquely-available persistent value for link local address.

`kdebug` *n*
Enable debugging code in the kernel-level PPP driver. Argument *n* is the sum of the following values: `1` to enable general debug messages, `2` to request that contents of received

packets be printed, and 4 to request contents of transmitted packets be printed. Messages printed by the kernel are logged by `syslogd(1M)` to a file directed in the `/etc/syslog.conf` configuration file. Do not use the `kdebug` option to debug failed links. Use the `debug` option instead.

`lcp-echo-failure` *n*
If this option is given, `pppd` presumes the peer to be dead if *n* LCP Echo-Requests are sent without receiving a valid LCP Echo-Reply. If this happens, `pppd` terminates the connection. This option requires a non-zero value for the `lcp-echo-interval` parameter. This option enables `pppd` to terminate after the physical connection is broken (for example, if the modem has hung up) in situations where no hardware modem control lines are available.

`lcp-echo-interval` *n*
If this option is given, `pppd` sends an LCP Echo-Request frame to the peer every *n* seconds. Normally the peer responds to the Echo-Request by sending an Echo-Reply. This option can be used with the `lcp-echo-failure` option to detect that the peer is no longer connected.

`lcp-max-configure` *n*
Set the maximum number of LCP Configure-Request transmissions to *n* (default 10).

`lcp-max-failure` *n*
Set the maximum number of LCP Configure-NAKs sent before starting to send Configure-Rejects instead to *n* (default 10).

`lcp-max-terminate` *n*
Set the maximum number of LCP Terminate-Request transmissions to *n* (default 3).

`lcp-restart` *n*
Set the LCP restart interval (retransmission timeout) to *n* seconds (default 3).

`linkname` *name*
Sets the logical name of the link to *name*. `pppd` creates a file named `ppp-`*name*`.pid` in `/var/run` containing its process ID. This is useful in determining which instance of `pppd` is responsible for the link to a given peer system. This is a privileged option.

`local`
Do not use modem control lines. With this option, `pppd` ignores the state of the CD (Carrier Detect) signal from the modem and does not change the state of the DTR (Data Terminal Ready) signal.

`logfd` *n*
Send log messages to file descriptor *n*. `pppd` sends log messages to (at most) one file or file descriptor (as well as sending the log messages to syslog), so this option and the `logfile` option are mutually exclusive. By default `pppd` sends log messages to `stdout` (file descriptor 1) unless the serial port is open on stdout.

`logfile` *filename*
Append log messages to the file *filename* (and send the log messages to syslog). The file is opened in append mode with the privileges of the user who invoked `pppd`.

`login`
Use the system password database for authenticating the peer using PAP, and record the user in the system `wtmp` file. Note that the peer must have an entry in the `/etc/ppp/pap-secrets` file and the system password database to be allowed access.

`maxconnect` *n*
Terminate the connection after it has been available for network traffic for *n* seconds (that is, *n* seconds after the first network control protocol starts). An LCP Time-Remaining message is sent when the first NCP starts, and again when 5, 2, and 0.5 minutes are remaining.

`maxfail` *n*
Terminate after *n* consecutive failed connection attempts. A value of 0 means no limit. The default value is 10.

`modem`
Use the modem control lines. This option is the default. With this option, `pppd` waits for the CD (Carrier Detect) signal from the modem to be asserted when opening the serial device (unless a connect script is specified), and drops the DTR (Data Terminal Ready) signal briefly when the connection is terminated and before executing the connect script.

`ms-dns` <*addr*>
If `pppd` is acting as a server for Microsoft Windows clients, this option allows `pppd` to supply one or two DNS (Domain Name Server) addresses to the clients. The first instance of this option specifies the primary DNS address; the second instance (if given) specifies the secondary DNS address. If the first instance specifies a name that resolves to multiple IP addresses, then the first two addresses are used. (This option is present in some older versions of `pppd` under the name `dns-addr`.)

`ms-lanman`
If `pppd` connects as a client to a Microsoft server and uses MS-CHAPv1 for authentication, this option selects the LAN Manager password style instead of Microsoft NT.

`ms-wins` <*addr*>
If `pppd` acts as a server for Microsoft Windows or Samba clients, this option allows `pppd` to supply one or two WINS (Windows Internet Name Services) server addresses to the clients. The first instance of this option specifies the primary WINS address; the second instance (if given) specifies the secondary WINS address. As with `ms-dns`, if the name specified resolves to multiple IP addresses, then the first two will be taken as primary and secondary.

`name` *name*
Set the name of the local system for authentication purposes to *name*. This is a privileged option. With this option, `pppd` uses lines in the secrets files that have *name* as the second field to look for a secret to use in authenticating the peer. In addition, unless overridden with the `user` option, *name* is used as the name to send to the peer when authenticating the local system. (Note that `pppd` does not append the domain name to *name*.)

`no-accm-test`
Disable use of `asyncmap` (ACCM) checking using LCP Echo-Request messages. If the `lcp-echo-failure` is used on an asynchronous line, `pppd` includes all control characters in the first *n* LCP Echo-Request messages. If the `asyncmap` is set incorrectly, the link drops rather than continue operation with random failures. This option disables that feature.

`noaccomp`
Disable HDLC Address/Control compression in both directions (send and receive).

`noauth`
Do not require the peer to authenticate itself. This option is privileged.

`nobsdcomp`
Disables BSD-Compress compression; `pppd` will not request or agree to compress packets using the BSD-Compress scheme. This option is not necessary if `noccp` is specified.

`noccp`
Disable CCP (Compression Control Protocol) negotiation. This option should only be required if the peer has bugs or becomes confused by requests from `pppd` for CCP negotiation. If CCP is disabled, then BSD and deflate compression do not need to be separately disabled.

`nocrtscts`
Disable hardware flow control (i.e. RTS/CTS) on the serial port. If the `crtscts`, `nocrtscts`, `cdtrcts` or `nocdtrcts` options are not given, the hardware flow control setting for the serial port is left unchanged.

`nocdtrcts`
This option is a synonym for `nocrtscts`. Either option will disable both forms of hardware flow control.

`nodefaultroute`
Disable the `defaultroute` option. You can prevent non-root users from creating default routes with `pppd` by placing this option in the `/etc/ppp/options` file.

`nodeflate`
Disables deflate compression; `pppd` will not request or agree to compress packets using the deflate scheme. This option is not necessary if `noccp` is specified.

`nodeflatedraft`
Do not use Internet Draft (incorrectly assigned) algorithm number for deflate compression. This option is not necessary if `noccp` is specified.

`nodetach`
Do not detach from the controlling terminal. Without this option, `pppd` forks to become a background process if a serial device other than the terminal on the standard input is specified.

`noendpoint`
Do not send or accept the Multilink Endpoint Discriminator option.

`noident`
Disable use of LCP Identification. LCP Identification messages will not be sent to the peer, but received messages will be logged. (Specify this option twice to completely disable LCP Identification. In this case, `pppd` sends LCP Code-Reject in response to received LCP Identification messages.)

`noip`
Disable IPCP negotiation and IP communication. Use this option only if the peer has bugs or becomes confused by requests from `pppd` for IPCP negotiation.

`noipv6`
Disable IPv6CP negotiation and IPv6 communication. IPv6 is not enabled by default.

`noipdefault`
Disables the default behavior when no local IP address is specified, which is to determine (if possible) the local IP address from the hostname. With this option, the peer must supply the local IP address during IPCP negotiation (unless it specified explicitly on the command line or in an options file).

`nolog`
Do not send log messages to a file or file descriptor. This option cancels the `logfd` and `logfile` options. `nologfd` acts as an alias for this option.

`nomagic`
Disable magic number negotiation. With this option, `pppd` cannot detect a looped-back line. Use this option only if the peer has bugs. Do not use this option to work around the "Serial line is looped back" error message.

`nopam`
This privileged option disables use of pluggable authentication modules. If this option is specified, `pppd` reverts to standard authentication mechanisms. The default is not to use PAM.

`nopcomp`
Disable protocol field compression negotiation in the receive and the transmit direction.

`nopersist`
Exit once a connection has been made and terminated. This is the default unless the `persist` or `demand` option is specified.

`noplink`
Cause `pppd` to use I_LINK instead of I_PLINK. This is the default. When I_LINK is used, the system cleans up terminated interfaces (even when SIGKILL is used) but does not allow `ifconfig(1M)` to unplumb PPP streams or insert or remove modules dynamically. Use the `plink` option if `ifconfig(1M)` modinsert, modremove or unplumb support is needed.

`nopredictor1`
Do not accept or agree to Predictor-1 compression. (This option is accepted for compatibility. The implementation does not support Predictor-1 compression.)

`noproxyarp`
Disable the `proxyarp` option. If you want to prevent users from creating proxy ARP entries with `pppd`, place this option in the `/etc/ppp/options` file.

`notty`
Normally, `pppd` requires a terminal device. With this option, `pppd` allocates itself a pseudo-tty master/slave pair and uses the slave as its terminal device. `pppd` creates a child process to act as a character shunt to transfer characters between the pseudo-tty master and its standard input and output. Thus, `pppd` transmits characters on its standard output and receives characters on its standard input even if they are not terminal devices. This option increases the latency and CPU overhead of transferring data over the ppp interface as all of the characters sent and received must flow through the character shunt process. An explicit device name may not be given if this option is used.

`novj`
Disable Van Jacobson style TCP/IP header compression in both the transmit and the receive direction.

`novjccomp`
Disable the connection-ID compression option in Van Jacobson style TCP/IP header compression. With this option, `pppd` does not omit the connection-ID byte from Van Jacobson compressed TCP/IP headers, nor does it ask the peer to do so. This option is unnecessary if `novj` is specified.

`pam`
This privileged option enables use of PAM. If this is specified, `pppd` uses the `pam(3PAM)` framework for user authentication with a service name of "ppp" if the `login` option and PAP authentication are used. The default is not to use PAM.

`papcrypt`
Indicates that `pppd` should not accept a password which, before encryption, is identical to the secret from the `/etc/ppp/pap-secrets` file. Use this option if the secrets in the `pap-secrets` file are in `crypt(3C)` format.

`pap-max-authreq` *n*
Set the maximum number of PAP authenticate-request transmissions to *n* (default 10).

`pap-restart` *n*
Set the PAP restart interval (retransmission timeout) to *n* seconds (default 3).

`pap-timeout` *n*
Set the maximum time that `pppd` waits for the peer to authenticate itself with PAP to *n* seconds (0= no limit). The default is 30 seconds.

`password` *string*
Password string for authentication to the peer.

`persist`
Do not exit after a connection is terminated; instead try to reopen the connection.

`plink`
Cause `pppd` to use I_PLINK instead of I_LINK. The default is to use I_LINK, which cleans up terminated interface (even if SIGKILL is used), but does not allow `ifconfig(1M)` to unplumb PPP streams or insert or remove modules dynamically. Use this option if `ifconfig(1M)` modinsert/modremove/unplumb support is needed. See also the `plumbed` option.

`plugin` *filename*
Load the shared library object file *filename* as a plugin. This is a privileged option. Unless the filename specifies an explicit path, `/etc/ppp/plugins` and `/usr/lib/inet/ppp` will be searched for the object to load in that order.

`plumbed`
This option indicates that `pppd` should find a plumbed interface and use that for the session. If IPv4 addresses or IPv6 interface IDs or link MTU are otherwise unspecified, they are copied from the interface selected. This mode mimics some of the functionality of the older `aspppd` implementation and may be helpful when `pppd` is used with external applications that use `ifconfig(1M)`.

`pppmux` *timer*
Enable PPP Multiplexing option negotiation and set transmit multiplexing timeout to *timer* microseconds.

`privgroup` *group-name*
Allows members of group *group-name* to use privileged options. This is a privileged option. Because there is no guarantee that members of *group-name* cannot use `pppd` to become root themselves, you should be careful using this option. Consider it equivalent to putting the members of *group-name* in the `root` or `sys` group.

`proxyarp`
Add an entry to the system's Address Resolution Protocol (ARP) table with the IP address of the peer and the Ethernet address of this system. When you use this option, the peer appears to other systems to be on the local Ethernet. The remote address on the PPP link must be in the same subnet as assigned to an Ethernet interface.

`pty` *script*
Specifies that the command *script*, and not a specific terminal device is used for serial communication. `pppd` allocates itself a pseudo-tty master/slave pair and uses the slave as its terminal device. *script* runs in a child process with the pseudo-tty master as its standard input and output. An explicit device name may not be given if this option is used. (Note: if the `record` option is used in conjunction with the `pty` option, the child process will have pipes on its standard input and output.)

`receive-all`
With this option, `pppd` accepts all control characters from the peer, including those marked in the receive `asyncmap`. Without this option, `pppd` discards those characters as specified in *RFC 1662*. This option should be used only if the peer has bugs, as is often found with dial-back implementations.

`record` *filename*
Directs `pppd` to record all characters sent and received to a file named *filename*. *filename* is opened in append mode, using the user's user-ID and permissions. Because this option uses a pseudo-tty and a process to transfer characters between the pseudo-tty and the real serial device, it increases the latency and CPU overhead of transferring data over the PPP interface. Characters are stored in a tagged format with timestamps that can be displayed in readable form using the `pppdump`(1M) program. This option is generally used when debugging the kernel portion of `pppd` (especially CCP compression algorithms) and not for debugging link configuration problems. See the `debug` option.

`remotename` *name*
Set the assumed name of the remote system for authentication purposes to *name*. Microsoft WindowsNT does not provide a system name in its CHAP Challenge messages, and this option is often used to work around this problem.

`refuse-chap`
With this option, `pppd` will not agree to authenticate itself to the peer using standard Challenge Handshake Authentication Protocol (CHAP). (MS-CHAP is not affected.)

`refuse-mschap`
Do not agree to authenticate to peer with MS-CHAPv1. If this option is specified, requests for MS-CHAPv1 authentication from the peer are declined with LCP Configure-Nak. That option does not disable any other form of CHAP.

`refuse-mschapv2`
Do not agree to authenticate to peer with MS-CHAPv2. If specified, this option requests that MS-CHAPv2 authentication from the peer be declined with LCP Configure-Nak. That option does not disable any other form of CHAP.

`refuse-pap`
With this option, `pppd` will not agree to authenticate itself to the peer using Password Authentication Protocol (PAP).

`require-chap`
Require the peer to authenticate itself using standard CHAP authentication. MS-CHAP is not affected.

`require-mschap`
Require the peer to authenticate itself using MS-CHAPv1 authentication.

`require-mschapv2`
Require the peer to authenticate itself using MS-CHAPv2 authentication.

`require-pap`
Require the peer to authenticate itself using PAP authentication.

`show-password`
When logging contents of PAP packets, this option causes `pppd` to show the password string in the log message.

`silent`
With this option, `pppd` will not transmit LCP packets to initiate a connection until a valid LCP packet is received from the peer. This is like the "passive" option with older versions of `pppd` and is retained for compatibility, but the current `passive` option is preferred.

`small-accm-test`
When checking the `asyncmap` (ACCM) setting, `pppd` uses all 256 possible values by default. See `no-accm-test`. This option restricts the test so that only the 32 values affected by standard ACCM negotiation are tested. This option is useful on very slow links.

`socket` *host*:*port*
Connect to given host and port using TCP and run PPP over this connection.

`sync`
Use synchronous HDLC serial encoding instead of asynchronous. The device used by `pppd` with this option must have sync support. Currently supports `zs`, `se`, and `hsi` drivers.

`unit` *n*
Set PPP interface unit number to *n*, if possible.

`updetach`
With this option, `pppd` detaches from its controlling terminal after establishing the PPP connection. When this is specified, messages sent to `stderr` by the connect script, usually `chat(1M)`, and debugging messages from the debug option are directed to `pppd`'s standard output.

`usehostname`
Enforce the use of the hostname with domain name appended, if given, as the name of the local system for authentication purposes. This overrides the `name` option. Because the `name` option is privileged, this option is normally not needed.

`usepeerdns`
Ask the peer for up to two DNS server addresses. Addresses supplied by the peer, if any, are passed to the `/etc/ppp/ip-up` script in the environment variables DNS1 and DNS2. In addition, `pppd` creates an `/etc/ppp/resolv.conf` file containing one or two nameserver lines with the address(es) supplied by the peer.

`user` *name*
Sets the name used for authenticating the local system to the peer to *name*.

`vj-max-slots` *n*
Sets the number of connection slots to be used by the Van Jacobson TCP/IP header compression and decompression code to *n*, which must be between 2 and 16 (inclusive).

`welcome` *script*
Run the executable or shell command specified by *script* before initiating PPP negotiation, after the connect script, if any, has completed. A value for this option from a privileged source cannot be overridden by a non-privileged user.

`xonxoff`
Use software flow control, that is, XON/XOFF, to control the flow of data on the serial port.

Obsolete Options

The following options are obsolete:

`+ua` *name* Read a PAP user name and password from the file *name*. This file must have two lines for name and password. Name and password are sent to the peer when the peer requests PAP authentication.

`+ipv6` Enable IPv6 and IPv6CP without specifying interface identifiers.

`--version` Show version number and exit.

`--help` Show brief help message and exit.

Extended Description

The following sections discuss miscellaneous features of `pppd`:

Security

`pppd` allows system administrators to provide legitimate users with PPP access to a server machine without fear of compromising the security of the server or the network it runs on. Access control is provided by restricting IP addresses the peer may use based on its authenticated identity (if any), and through restrictions on options a non-privileged user may use. Options that permit potentially insecure configurations are privileged. Privileged options are accepted only in files that are under the control of the system administrator or when `pppd` is being run by root.

By default, `pppd` allows an unauthenticated peer to use a given IP address only if the system does not already have a route to that IP address. For example, a system with a permanent connection to the wider Internet will normally have a default route, meaning all peers must authenticate themselves to set up a connection. On such a system, the `auth` option is the default. Conversely, a system with a PPP link that comprises the only connection to the Internet probably does not possess a default route, so the peer can use virtually any IP address without authenticating itself.

Security-sensitive options are privileged and cannot be accessed by a non-privileged user running `pppd`, either on the command line, in the user's `$HOME/.pprc` file, or in an options file read using the `file` option. Privileged options may be used in `/etc/ppp/options` file or in an options file read using the `call` option. If `pppd` is run by the root user, privileged options can be used without restriction. If the `/etc/ppp/options` file does not exist, then only root may invoke `pppd`. The `/etc/ppp/options` file must be created (but may be empty) to allow ordinary non-root users to access `pppd`.

When opening the device, `pppd` uses the invoking user's user ID or the root UID (that is, 0), depending if the device name was specified by the user or the system administrator. If the device name comes from a privileged source, that is, `/etc/ppp/options` or an options file read using the `call` option, `pppd` uses full root privileges when opening the device. Thus, by creating an appropriate file under `/etc/ppp/peers`, the system administrator can allow users

to establish a PPP connection via a device that they would not normally have access to. Otherwise pppd uses the invoking user's real UID when opening the device.

Authentication

During the authentication process, one peer convinces the other of its identity by sending its name and some secret information to the other. During authentication, the first peer becomes the "client" and the second becomes the "server." Authentication names can (but are not required to) correspond to the peer's Internet hostnames.

pppd supports four authentication protocols: the Password Authentication Protocol (PAP) and three forms of the Challenge Handshake Authentication Protocol (CHAP). With the PAP protocol, the client sends its name and a cleartext password to the server to authenticate itself. With CHAP, the server initiates the authentication exchange by sending a challenge to the client who must respond with its name and a hash value derived from the shared secret and the challenge.

The PPP protocol is symmetrical, meaning that each peer may be required to authenticate itself to the other. Different authentication protocols and names can be used for each exchange.

By default, pppd authenticates if requested and does not require authentication from the peer. However, pppd does not authenticate itself with a specific protocol if it has no secrets that can do so.

pppd stores authentication secrets in the /etc/ppp/pap-secrets (for PAP), and /etc/ppp/chap-secrets (for CHAP) files. Both files use the same format. pppd uses secrets files to authenticate itself to other systems and to authenticate other systems to itself.

Secrets files contain one secret per line. Secrets are specific to a particular combination of client and server and can only be used by that client to authenticate itself to that server. Each line in a secrets file has a minimum of three fields that contain the client and server names followed by the secret. Often, these three fields are followed by IP addresses that are used by clients to connect to a server.

A secrets file is parsed into words, with client name, server name and secrets fields allocated one word each. Embedded spaces or other special characters within a word must be quoted or escaped. Case is significant in all three fields.

A secret beginning with an at sign (“@”) is followed by the name of a file containing the secret. An asterisk (*) as the client or server name matches any name. When choosing a match, pppd selects the one with the fewest wildcards. Succeeding words on a line are interpreted by pppd as acceptable IP addresses for that client. IP Addresses are disallowed if they appear in lines that contain only three words or lines whose first word begins with a hyphen (“-”). To allow any address, use "*". An address starting with an exclamation point (”!”) indicates that the specified address is not acceptable. An address may be followed by "/" and a number *n* to indicate a whole subnet (all addresses that have the same value in the most significant *n* bits).

In this form, the address may be followed by a plus sign ("+") to indicate that one address from the subnet is authorized, based on the ppp network interface unit number in use. In this case, the host part of the address is set to the unit number, plus one.

When authenticating the peer, `pppd` chooses a secret with the peer's name in the first field of the secrets file and the name of the local system in the second field. The local system name defaults to the hostname, with the domain name appended if the `domain` option is used. The default can be overridden with the `name` option unless the `usehostname` option is used.

When authenticating to the peer, `pppd` first determines the name it will use to identify itself to the peer. This name is specified with the `user` option. If the `user` option is not used, the name defaults to the host name of the local system. `pppd` then selects a secret from the secrets file by searching for an entry with a local name in the first field and the peer's name in the second field. `pppd` will know the name of the peer if standard CHAP authentication is used because the peer will have sent it in the Challenge packet. However, if MS-CHAP or PAP is being used, `pppd` must determine the peer's name from the options specified by the user. The user can specify the peer's name directly with the `remotename` option. Otherwise, if the remote IP address was specified by a name, rather than in numeric form, that name will be used as the peer's name. If that fails, `pppd` uses the null string as the peer's name.

When authenticating the peer with PAP, the supplied password is compared with data in the secrets file. If the password and secret do not match, the password is encrypted using `crypt()` and checked against the secret again. If the `papcrypt` option is given, the first unencrypted comparison is omitted for better security, and entries must thus be in encrypted `crypt(3C)` form.

If the `login` option is specified, the username and password are also checked against the system password database. This allows you to set up the `pap-secrets` file to enable PPP access only to certain users, and to restrict the set of IP addresses available to users. Typically, when using the `login` option, the secret in `/etc/ppp/pap-secrets` would be "", which matches any password supplied by the peer. This makes having the same secret in two places unnecessary. When `login` is used, the `pam` option enables access control through `pam(3PAM)`.

Authentication must be completed before IPCP (or other network protocol) can be started. If the peer is required to authenticate itself and fails, `pppd` closes LCP and terminates the link. If IPCP negotiates an unacceptable IP address for the remote host, IPCP is closed. IP packets are sent or received only when IPCP is open.

To allow hosts that cannot authenticate themselves to connect and use one of a restricted set of IP addresses, add a line to the `pap-secrets` file specifying the empty string for the client name and secret.

Additional `pppd` options for a given peer may be specified by placing them at the end of the secrets entry, separated by two dashes (––). For example

```
peername servername secret ip-address -- novj
```

Routing When IPCP negotiation is complete, pppd informs the kernel of the local and remote IP addresses for the PPP interface and creates a host route to the remote end of the link that enables peers to exchange IP packets. Communication with other machines generally requires further modification to routing tables and/or Address Resolution Protocol (ARP) tables. In most cases the defaultroute and/or proxyarp options are sufficient for this, but further intervention may be necessary. If further intervention is required, use the /etc/ppp/ip-up script or a routing protocol daemon.

To add a default route through the remote host, use the defaultroute option. This option is typically used for "client" systems; that is, end-nodes that use the PPP link for access to the general Internet.

In some cases it is desirable to use proxy ARP, for example on a server machine connected to a LAN, to allow other hosts to communicate with the remote host. proxyarp instructs pppd to look for a network interface on the same subnet as the remote host. That is, an interface supporting broadcast and ARP that is not a point-to-point or loopback interface and that is currently up. If found, pppd creates a permanent, published ARP entry with the IP address of the remote host and the hardware address of the network interface.

When the demand option is used, the interface IP addresses are already set at the time when IPCP comes up. If pppd cannot negotiate the same addresses it used to configure the interface, it changes the interface IP addresses to the negotiated addresses. This may disrupt existing connections. Using demand dialing with peers that perform dynamic IP address assignment is not recommended.

Scripts pppd invokes scripts at various stages during processing that are used to perform site-specific ancillary processing. These scripts may be shell scripts or executable programs. pppd does not wait for the scripts to finish. The scripts are executed as root (with the real and effective user-id set to 0), enabling them to update routing tables, run privileged daemons, or perform other tasks. Be sure that the contents of these scripts do not compromise your system's security. pppd runs the scripts with standard input, output and error redirected to /dev/null, and with an environment that is empty except for some environment variables that give information about the link. The pppd environment variables are:

DEVICE	Name of the serial tty device.
IFNAME	Name of the network interface.
IPLOCAL	IP address for the link's local end. This is set only when IPCP has started.
IPREMOTE	IP address for the link's remote end. This is set only when IPCP has started.
PEERNAME	Authenticated name of the peer. This is set only if the peer authenticates itself.
SPEED	Baud rate of the tty device.
ORIG_UID	Real user-id of user who invoked pppd.

PPPLOGNAME Username of the real user-id who invoked pppd. This is always set.

pppd also sets the following variables for the ip-down and auth-down scripts:

CONNECT_TIME Number of seconds between the start of PPP negotiation and connection termination.

BYTES_SENT Number of bytes sent at the level of the serial port during the connection.

BYTES_RCVD Number of bytes received at the level of the serial port during the connection.

LINKNAME Logical name of the link, set with the linkname option.

If they exist, pppd invokes the following scripts. It is not an error if they do not exist.

/etc/ppp/auth-up Program or script executed after the remote system successfully authenticates itself. It is executed with five command-line arguments: interface-name peer-name user-name tty-device speed. Note that this script is not executed if the peer does not authenticate itself, for example, when the noauth option is used.

/etc/ppp/auth-down Program or script executed when the link goes down if /etc/ppp/auth-up was previously executed. It is executed in the same manner with the same parameters as /etc/ppp/auth-up.

/etc/ppp/ip-up A program or script that is executed when the link is available for sending and receiving IP packets (that is, IPCP has come up). It is executed with six command-line arguments: interface-name tty-device speed local-IP-address remote-IP-address ipparam.

/etc/ppp/ip-down A program or script which is executed when the link is no longer available for sending and receiving IP packets. This script can be used for undoing the effects of the /etc/ppp/ip-up script. It is invoked in the same manner and with the same parameters as the ip-up script.

/etc/ppp/ipv6-up Similar to /etc/ppp/ip-up, except that it is executed when the link is available for sending and receiving IPv6 packets. Executed with six command-line arguments: interface-name tty-device speed local-link-local-address remote-link-local-address ipparam.

/etc/ppp/ipv6-down Similar to /etc/ppp/ip-down, but executed when IPv6 packets can no longer be transmitted on the link. Executed with the same parameters as the ipv6-up script.

Examples

EXAMPLE 1 Using the auth Option

The following examples assume that the /etc/ppp/options file contains the auth option.

EXAMPLE 1 Using the auth Option *(Continued)*

pppd is commonly used to dial out to an ISP. You can do this using the "pppd call isp" command where the /etc/ppp/peers/isp file is set up to contain a line similar to the following:

```
cua/a 19200 crtscts connect '/usr/bin/chat -f /etc/ppp/chat-isp' noauth
```

For this example, chat(1M) is used to dial the ISP's modem and process any login sequence required. The /etc/ppp/chat-isp file is used by chat and could contain the following:

```
ABORT "NO CARRIER"
ABORT "NO DIALTONE"
ABORT "ERROR"
ABORT "NO ANSWER"
ABORT "BUSY"
ABORT "Username/Password Incorrect"
"" "at"
OK "at&f&d2&c1"
OK "atdt2468135"
"name:" "^Umyuserid"
"word:" "\qmypassword"
"ispts" "\q^Uppp"
"~-^Uppp-~"
```

See the chat(1M) man page for details of chat scripts.

EXAMPLE 2 Using pppd with proxyarp

pppd can also provide a dial-in ppp service for users. If the users already have login accounts, the simplest way to set up the ppp service is to let the users log in to their accounts and run pppd as shown in the following example:

```
example% pppd proxyarp
```

EXAMPLE 3 Providing a User with Access to PPP Facilities

To provide a user with access to the PPP facilities, allocate an IP address for the user's machine, create an entry in /etc/ppp/pap-secrets or /etc/ppp/chap-secrets. This enables the user's machine to authenticate itself. For example, to enable user "Joe" using machine "joespc" to dial in to machine "server" and use the IP address "joespc.my.net," add the following entry to the /etc/ppp/pap-secrets or /etc/ppp/chap-secrets files:

```
joespc    server    "joe's secret"    joespc.my.net
```

Alternatively, you can create another username, for example "ppp," whose login shell is /usr/bin/pppd and whose home directory is /etc/ppp. If you run pppd this way, add the options to the /etc/ppp/.ppprc file.

EXAMPLE 3 Providing a User with Access to PPP Facilities *(Continued)*

If your serial connection is complex, it may be useful to escape such control characters as XON (^Q) and XOFF (^S), using `asyncmap a0000`. If the path includes a telnet, escape ^] (`asyncmap 200a0000`). If the path includes a `rlogin` command, add `escape ff` option to the options, because `rlogin` removes the window-size-change sequence [0xff, 0xff, 0x73, 0x73, followed by any 8 bytes] from the stream.

Exit Status The `pppd` exit status indicates errors or specifies why a link was terminated. Exit status values are:

`0` `pppd` has detached or the connection was successfully established and terminated at the peer's request.

`1` An immediately fatal error occurred. For example, an essential system call failed.

`2` An error was detected in the options given. For example, two mutually exclusive options were used, or `/etc/ppp/options` is missing and the user is not root.

`3` `pppd` is not `setuid-root` and the invoking user is not root.

`4` The kernel does not support PPP. For example, the PPP kernel driver is not included or cannot be loaded.

`5` `pppd` terminated because it was sent a SIGINT, SIGTERM or SIGHUP signal.

`6` The serial port could not be locked.

`7` The serial port could not be opened.

`8` The connect script failed and returned a non-zero exit status.

`9` The command specified as the argument to the `pty` option could not be run.

`10` The PPP negotiation failed because no network protocols were able to run.

`11` The peer system failed or refused to authenticate itself.

`12` The link was established successfully, but terminated because it was idle.

`13` The link was established successfully, but terminated because the connect time limit was reached.

`14` Callback was negotiated and an incoming call should arrive shortly.

`15` The link was terminated because the peer is not responding to echo requests.

`16` The link was terminated by the modem hanging up.

`17` The PPP negotiation failed because serial loopback was detected.

`18` The init script failed because a non-zero exit status was returned.

`19` Authentication to the peer failed.

Files

`/var/run/spppn.pid`	Process-ID for `pppd` process on PPP interface unit *n*.
`/var/run/ppp-`*name*`.pid`	Process-ID for `pppd` process for logical link name (see the `linkname` option).
`/etc/ppp/pap-secrets`	Usernames, passwords and IP addresses for PAP authentication. This file should be owned by root and not readable or writable by any other user, otherwise `pppd` will log a warning.
`/etc/ppp/chap-secrets`	Names, secrets and IP addresses for all forms of CHAP authentication. The `/etc/ppp/pap-secrets` file should be owned by `root` should not readable or writable by any other user, otherwise, `pppd` will log a warning.
`/etc/ppp/options`	System default options for `pppd`, read before user default options or command-line options.
`$HOME/.ppprc`	User default options, read before `/etc/ppp/options.`*ttyname*.
`/etc/ppp/options.`*ttyname*	System default options for the serial port in use; read after `$HOME/.ppprc`. The *ttyname* component of this filename is formed when the initial `/dev/` is stripped from the port name (if present), and slashes (if any) are converted to dots.
`/etc/ppp/peers`	Directory with options files that may contain privileged options, even if `pppd` was invoked by a user other than `root`. The system administrator can create options files in this directory to permit non-privileged users to dial out without requiring the peer to authenticate, but only to certain trusted peers.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpppdu
Interface Stability	Evolving

See Also `chat(1M)`, `ifconfig(1M)`, `crypt(3C)`, `pam(3PAM)`, `attributes(5)`

Haskin, D., Allen, E. *RFC 2472 – IP Version 6 Over PPP*. Network Working Group. December 1998.

Jacobson, V. *RFC 1144, Compressing TCP/IP Headers for Low-Speed Serial Links*. Network Working Group. February, 1990

Lloyd, B., Simpson, W. *RFC 1334, PPP Authentication Protocols*. Network Working Group. October 1992.

McGregor, G. *RFC 1332, The PPP Internet Protocol Control Protocol (IPCP)*. Network Working Group. May 1992.

Rivest, R. *RFC 1321, The MD5 Message-Digest Algorithm*. Network Working Group. April 1992

Simpson, W. *RFC 1661, The Point-to-Point Protocol (PPP)*. Network Working Group. July 1994.

Simpson, W. *RFC 1662, HDLC-like Framing* . Network Working Group. July 1994.

Notes These signals affect `pppd` behavior:

`SIGINT, SIGTERM`	Terminate the link, restore the serial device settings and exit.
`SIGHUP`	Terminate the link, restore the serial device settings and close the serial device. If the `persist` or `demand` option is specified, `pppd` attempts to reopen the serial device and start another connection after the holdoff period. Otherwise `pppd` exits. If received during the holdoff period, `SIGHUP` causes `pppd` to end the holdoff period immediately.
`SIGUSR1`	Toggles the state of the `debug` option and prints link status information to the log.
`SIGUSR2`	Causes `pppd` to renegotiate compression. This is useful to re-enable compression after it has been disabled as a result of a fatal decompression error. (Fatal decompression errors generally indicate a bug in an implementation.)

Diagnostics Messages are sent to the syslog daemon using facility `LOG_DAEMON`. To see error and debug messages, edit the `/etc/syslog.conf` file to direct the messages to the desired output device or file, or use the `updetach` or `logfile` options.

The `debug` option causes the contents of all LCP, PAP, CHAP or IPCP control packets sent or received to be logged. This is useful if PPP negotiation does not succeed or if authentication fails.

Debugging can also be enabled or disabled by sending a `SIGUSR1` signal, which acts as a toggle to the `pppd` process.

Name pppoec – PPPoE chat utility

Synopsis

```
pppoec [-omillisecs] [-smillisecs] [-v] device
     [service [ [except]server... [only]]]

pppoec [-omillisecs] [-v] -i [device]
```

Description The `pppoec` utility implements the client-side negotiation of PPPoE. It is intended to be used with the `pppd(1M)` `connect` option, in the same manner as the `chat(1M)` utility is used for asynchronous dial-up PPP.

When given with the `-i` flag, `pppoec` sends out a broadcast query on the given interface named by the *device* parameter. You can specify no other arguments in this mode. All responding PPPoE servers and the offered services are displayed on standard output.

Otherwise, when given without the `-i` flag, `pppoec` does the full PPPoE client-side negotiation. The *device* parameter is the intended Ethernet interface, and must already be plumbed with `sppptun(1M)`. The optional *service* parameter specifies a particular service desired; other offered services will be ignored. The optional *server* parameter specifies a specific server desired. You can specify *server* as an Ethernet address in the usual x:x:x:x:x:x format (with "*" in any of the six byte positions interpreted to mean "any"), or as a symbolic name resolved through `/etc/ethers` (or NIS), or as a PPPoE access concentrator name. The sense of the match (true or false) can be inverted by specifying the keyword `except` before this string. This parameter can be specified more than once, and the first match is taken.

If you specify the *server* parameter, then the selected servers become "preferred." If no preferred server responds, then the first responding server is used instead. To exclude non-matching servers entirely, append the keyword `only`.

Options The following options are supported:

`-i` Sends out broadcast query over interface specified by *device*.

`-o` Sets the initial wait time in milliseconds for PADO from the server before PADI is retried. The default is 500 milliseconds for normal operation, or 3000 milliseconds (3 seconds) for inquiry (`-i`) mode.

`-s` Sets the initial wait time in milliseconds for PADS from the server before PADR is retried. The default is 2000 milliseconds (2 seconds).

`-v` Displays verbose progress messages, including all PPPoE messages sent, and all state machine transitions.

You normally do not need to adjust the parameters set with `-o` and `-s`. They are provided for coping with unusually slow servers.

Operands The following operands are supported:

device plumbed Ethernet interface

server preferred server or, if you specify `only`, the specified server

service desired service; other available services are ignored

Examples

EXAMPLE 1 Connecting to Any Service on `hme0`

The following command enables you to connect to any PPPoE service on `hme0`:

```
# /usr/bin/pppd sppptun plugin pppoe.so \
connect "/usr/lib/inet/pppoec hme0" debug
```

Often, a command such as the preceding is specified in an `/etc/ppp/peers` file instead. For example, enter the following in `/etc/ppp/peers/myisp`:

```
sppptun
plugin pppoe.so
connect "/usr/lib/inet/pppoec hme0"
debug
```

To invoke the PPP connection described in the file, enter:

```
% /usr/bin/pppd call myisp
```

Note that, because the `/etc/ppp/peers` files are considered privileged by `pppd`, you need not be root to invoke the preceding command.

EXAMPLE 2 Connecting to a Particular Service

A more complex example: on `hme0`, connect to only the `internet` service offered by PPPoE servers with access concentrator name `isp`, but not to any Ethernet addresses starting with `40:0:1a`.

```
# /usr/lib/inet/pppoec hme0 internet except 40:0:1a:*:*:* isp only
```

Note that the `except 40:0:1a:*:*:*` filter must come before `isp`, because the filters are first-match.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files

`/usr/lib/inet/pppoec`	executable command
`/dev/sppptun`	Solaris PPP tunneling device driver.
`/etc/ppp/connect-errors`	usual location of error output (see DIAGNOSTICS, below)

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpppdt

See Also pppd(1M), sppptun(1M), pppoed(1M), sppptun(7M)

RFC 2516, Method for Transmitting PPP Over Ethernet (PPPoE), Mamakos et al, February 1999

Diagnostics Error messages are written to standard error, which is normally redirected by pppd to /etc/ppp/connect-errors. The errors can also be redirected to pppd's standard output by using the updetach option.

If you specify the -v, verbose progress messages are displayed, including all PPPoE messages sent, and all state machine transitions. Specifying the updetach or nodetach pppd option is helpful when using verbose mode.

Name pppoed – PPPoE server daemon

Synopsis `ppoed [`*options*`]`

Description The `pppoed` daemon implements the server-side negotiation of PPPoE. When a client requests service from this daemon, a copy of `pppd(1M)` is invoked to handle the actual PPP communication.

At startup, options are read from the command line and the `/etc/ppp/pppoe` file. After these options have been read, options in the per-device `/etc/ppp/pppoe.`*device* files are read, using the device names specified on the command line or in `/etc/ppp/pppoe`. Device names are not permitted in the per-device files. It is not an error if any of these files are absent; missing files are ignored.

Options are reread in the same order on `SIGHUP`. Except for the possibility of short delays due to the processing time, `SIGHUP` does not interfere with any client operations. Current status, including options read, is dumped to `/tmp/pppoed.`*pid* on `SIGINT`.

The options are used to set up a list of services to be offered to PPPoE clients on the broadcast domains (Ethernet subnets) specified by the named devices. Option parsing is always in one of two modes, either global mode or service mode. The initial mode at the beginning of each file (and the command line) is global mode. Options specified in global mode serve as default values for subsequently defined services. Service mode is entered by the `service` *name* option. In this mode, the named option is defined. Options that appear in this mode override any global mode definitions for the current service.

The option parsing follows standard shell tokenizing rules, using whitespace to delimit tokens, quotes to enclose strings that can contain whitespace, and escape sequences for special characters. Environment variables are substituted using familiar `$VAR` and `${VAR}` syntax and set using `NEWVAR=`*string*. Variables are both usable in subsequent options and provided to the `pppd(1M)` processes spawned for each client, but they are interpreted as they are encountered during option processing. Thus, all set variables are seen by all processes spawned; position in the configuration files has no effect on this.

Options The `pppoed` daemon supports the following options:

`client [except]` *client-list* This option restricts the clients that may receive the service. If the `except` keyword is given, then the clients on the list cannot access the service, but others can. If this keyword is not given, then only the listed clients can access the service.

This option can be specified more than once for a given service. For a given client, first match among all listed options encountered specifies the handling. If it matches an option with `except` specified, then access is denied. Otherwise, it is granted. The `client` list within a service is prepended to any list specified in the global context.

If no `client` options are given or if all options are specified with `except`, then all clients are permitted by default. If any `client` options without `except` are specified, then no clients are permitted by default.

The *client-list* is a comma-separated list of client identifiers. The match is made if any client on the list matches; thus, these are logically "ORed" together. Each client identifier can be either a symbolic name (resolved through `/etc/ethers` or NIS, as defined by `/etc/nsswitch.conf`) or a hexadecimal Ethernet address in the format `x:x:x:x:x:x`. In the latter case, any byte of the address can be "*", which matches any value in that position. For example, `40:0:1a:*:*:*` matches Ethernet adapters from the manufacturer assigned block `40:0:1a`.

`debug`

Increase debug logging detail level by one. The detail levels are 0 (no logging), 1 (errors only; the default), 2 (warnings), 3 (informational messages), and 4 (debug messages). Log messages are written by default to `syslog(3C)` using facility *daemon* (see the `log` option below). When specified on the command line or in the global context of the `/etc/ppp/pppoe` file, this option also sets the daemon's default (non-service-related) detail level.

`device` *device-list*

Specify the devices on which the service is available. The *device-list* is a comma-separated list of logical device names (without the leading `/dev/`), such as `hme0`. This option is ignored if encountered in the per-device `/etc/ppp/pppoe.`*device* files.

`extra` *string*

Specifies extra options to `pppd(1M)`. It defaults to "`plugin pppoe.so directtty`" and usually does not need to be overridden.

`file` *path*

Suspends parsing of the current file, returns to global mode, and reads options from *path*. This file must be present and readable; if it is not, an error is logged. When the end of that file is reached, processing returns to the current file and the mode is reset to global again.

The global mode options specified in files read by this command use the options set in the current file's global mode; this condition extends to any file included by those files. All files read are parsed as though the command line had specified this option, and thus inherit the command line's global modes.

This option can be used to revert to global mode at any point in an option file by specifying `file /dev/null`.

`group` *name*
: Specifies the group ID (symbolic or numeric) under which `pppd` is executed. If `pppoed` is not run as root, this option is ignored.

`log` *path*
: Specifies an alternate debug logging file. Debug messages are sent to this file instead of `syslog`. The special name `syslog` is recognized to switch logging back to `syslog`. When specified on the command line or in the global context of the `/etc/ppp/pppoe` file, this option also sets the daemon's default (non-service-related) log file.

`nodebug`
: Set debug logging detail level to 0 (no logging). When specified on the command line or in the global context of the `/etc/ppp/pppoe` file, this option also sets the daemon's default (non-service-related) detail level.

`nowildcard`
: Specifies that the current service should not be included in response to clients requesting "any" service. The client must ask for this service by name. When specified on the command line or in the global context of the `/etc/ppp/pppoe` file, this option causes `pppoed` to ignore all wildcard service requests.

`path` *path*
: Specifies the path to the `pppd` executable. Defaults to `/usr/bin/pppd`.

`pppd` *string*
: Passes command-line arguments to `pppd`. It can be used to set the IP addresses or configure security for the session. The default value is the empty string.

`server` *string*
: Specifies the PPPoE Access Concentrator name to be sent to the client. It defaults to "Solaris PPPoE".

`service` *name*
: Closes any service being defined and begins definition of a new service. The same service name can be used without conflict on multiple devices. If the same service name is used on a single device, then the last definition encountered during parsing overrides all previous definitions.

`user` *name*
: Specifies the user ID, symbolic or numeric, under which `pppd` is executed. If `pppoed` is not run as root, this option is ignored.

`wildcard`
: Specifies that the service should be included in responses to client queries that request "any" service, which is done by requesting a service name of length zero. When specified on the command line or in the global context of the

/etc/ppp/pppoe file, this option causes pppoed to ignore all wildcard service requests. This is the default.

Examples

EXAMPLE 1 Configuring for Particular Services

In the /etc/ppp/pppoe file:

```
service internet
    device $DEV
    pppd "proxyarp 192.168.1.1:"
service debugging
    device hme0,$DEV
    pppd "debug proxyarp 192.168.1.1:"
```

You then invoke the daemon with:

```
example% /usr/lib/inet/pppoed DEV=eri0
```

The lines in /etc/ppp/pppoe and the preceding command result in offering services "internet" and "debugging" (and responding to wildcard queries) on interface eri0, and offering only service "debugging" on interface hme0.

Signals

The pppoed daemon responds to the following signals:

SIGHUP — Causes pppoed to reparse the original command line and all configuration files, and close and reopen any log files.

SIGINT — Causes a snapshot of the state of the pppoed daemon to be written to /tmp/pppoed.*pid* (where *pid* is the decimal process ID of the daemon).

Files

/usr/lib/inet/pppoed	executable command
/dev/sppptun	Solaris PPP tunneling device driver
/etc/ppp/pppoe	main configuration option file
/etc/ppp/pppoe.*device*	per–device configuration option file
/etc/ppp/pppoe-errors	location of output from pppd's stderr
/etc/ppp/pppoe.if	list of Ethernet interfaces to be plumbed at boot time
/tmp/pppoed.*pid*	ASCII text file containing dumped pppoed state information

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpppdt

See Also pppd(1M), pppoec(1M), sppptun(1M), sppptun(7M)

Mamakos, L., et al. *RFC 2516, A Method for Transmitting PPP Over Ethernet (PPPoE)*. Network Working Group. February 1999

Notes Because `pppd` is installed setuid root, this daemon need not be run as root. However, if it is not run as root, the `user` and `group` options are ignored.

The Ethernet interfaces to be used must be plumbed for PPPoE using the sppptun(1M) utility before services can be offered.

The daemon operate runs even if there are no services to offer. If you want to modify a configuration, it is not necessary to terminate the daemon. Simply use `pkill -HUP pppoed` after updating the configuration files.

The PPPoE protocol is far from perfect. Because it runs directly over Ethernet, there is no possibility of security and the MTU is limited to 1492 (violating RFC 1661's default value of 1500). It is also not possible to run the client and the server of a given session on a single machine with a single Ethernet interface for testing purposes. The client and server portions of a single session must be run on separate Ethernet interfaces with different MAC addresses.

Name pppstats – print PPP statistics

Synopsis

```
pppstats [-a] [-v] [-r] [-z] [-c <count>] [-w <secs>]
     [interface]
```

Description The `pppstats` utility reports PPP-related statistics at regular intervals for the specified PPP interface. If the interface is unspecified, `pppstats` defaults to `sppp0`. The display is split horizontally into input and output sections containing columns of statistics describing the properties and volume of packets received and transmitted by the interface.

Options The `pppstats` options are:

`-a` Display absolute values rather than deltas. With this option, all reports show statistics for the time elapsed since the link was initiated. Without this option, the second and subsequent reports show statistics for the time since the last report.

`-c` *count* Repeat the display *count* times. If this option is not specified, the default repeat count is 1 if the `-w` option is not specified, otherwise infinity.

`-r` Display additional statistics summarizing the compression ratio achieved by the packet compression algorithm in use.

`-v` Display additional statistics relating to the performance of the Van Jacobson TCP header compression algorithm.

`-w` *wait* Pause *wait* seconds between each display. If this option is not specified, the default interval is five seconds.

`-z` Instead of the standard display, show statistics indicating the performance of the packet compression algorithm in use.

Extended Description The following fields are printed on the input side when the `-z` option is not used:

`IN` Total number of bytes received by this interface.

`PACK` Total number of packets received by this interface.

`VJCOMP` Number of header-compressed TCP packets received by this interface.

`VJUNC` Number of header-uncompressed TCP packets received by this interface. Not reported when the `-r` option is specified.

`VJERR` Number of corrupted or bogus header-compressed TCP packets received by this interface. Not reported when the `-r` option is specified.

`VJTOSS` Number of VJ header-compressed TCP packets dropped on reception by this interface because of preceding errors. Only reported when the `-v` option is specified.

`NON-VJ` Total number of non-TCP packets received by this interface. Only reported when the `-v` option is specified.

RATIO Compression ratio achieved for received packets by the packet compression scheme in use, defined as the uncompressed size divided by the compressed size. Only reported when the -r option is specified.

UBYTE Total number of bytes received, after decompression of compressed packets. Only reported when the -r option is specified.

The following fields are printed on the output side:

OUT Total number of bytes transmitted from this interface.

PACK Total number of packets transmitted from this interface.

VJCOMP Number of TCP packets transmitted from this interface with VJ-compressed TCP headers.

VJUNC Number of TCP packets transmitted from this interface with VJ-uncompressed TCP headers. Not reported when the -r option is specified.

NON-VJ Total number of non-TCP packets transmitted from this interface. Not reported when the -r option is specified.

VJSRCH Number of searches for the cached header entry for a VJ header compressed TCP packet. Only reported when the -v option is specified.

VJMISS Number of failed searches for the cached header entry for a VJ header compressed TCP packet. Only reported when the -v option is specified.

RATIO Compression ratio achieved for transmitted packets by the packet compression scheme in use, defined as the size before compression divided by the compressed size. Only reported when the -r option is specified.

UBYTE Total number of bytes to be transmitted before packet compression is applied. Only reported when the -r option is specified.

When the -z option is specified, pppstats displays the following fields relating to the packet compression algorithm currently in use. If packet compression is not in use, these fields display zeroes. The fields displayed on the input side are:

COMPRESSED BYTE Number of bytes of compressed packets received.

COMPRESSED PACK Number of compressed packets received.

INCOMPRESSIBLE BYTE Number of bytes of incompressible packets (that is, those which were transmitted in uncompressed form) received.

INCOMPRESSIBLE PACK Number of incompressible packets received.

COMP RATIO Recent compression ratio for incoming packets, defined as the uncompressed size divided by the compressed size (including both compressible and incompressible packets).

The fields displayed on the output side are:

`COMPRESSED BYTE`	Number of bytes of compressed packets transmitted.
`COMPRESSED PACK`	Number of compressed packets transmitted.
`INCOMPRESSIBLE BYTE`	Number of bytes of incompressible packets received; that is, those that were transmitted by the peer in uncompressed form.
`INCOMPRESSIBLE PACK`	Number of incompressible packets transmitted.
`COMP RATIO`	Recent compression ratio for outgoing packets.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpppdu
Interface Stability	Evolving

See Also `pppd(1M)`, `attributes(5)`

Name pprosetup – setup program for Patch Manager

Synopsis

```
/usr/sbin/pprosetup [-a admin-email-addr] [-b backout-dir]
     [-c config-name] [-C] [-d patch-dir]
     [ [-D | -M day-of-month | -W day-of-week] [-s hh:mm]]
     [-h] [-H] [-i [none | patch-property-list]] [-L]
     [-p [none | standard]] [-P patch-source-url]
     [-q sequester-dir] [-u user-name]
     [-U proxy-user-name] [-x [host:port]]
```

Description **Note –** This command is deprecated. Use the `smpatch set`, `smpatch unset`, and `smpatch get` commands instead. See the `smpatch(1M)` man page.

Use the `pprosetup` command, as superuser, to configure your patch management environment by doing the following:

- Scheduling the patch operations
- Setting a patch policy
- Specifying patch directories
- Specifying the hardware on the system
- Specifying alternate configurations

Scheduling the Patch Operations

Schedule the automatic synchronization of patches with Sun's patch base. This scheduling makes the `pprosvc` command run in *automatic mode*. This mode is set up by using the `cron` interface. Use the `-C`, `-D`, `-M`, `-s`, and `-W` options to perform the scheduling tasks.

If you do not want to schedule patch operations, you can run the `pprosvc` and `smpatch` commands in *manual mode*, which means running the tool from the command line.

Note that midnight is represented as 00:00.

Note – The `smpatch` command does not directly support this mechanism for scheduling patch operations. You can set up a schedule by using `cron` to run `smpatch` in local mode. See the `smpatch(1M)` man page.

Setting a Patch Policy

Patches are classified as being standard or nonstandard. A *standard patch* can be applied by `pprosvc` in automatic mode. Such a patch is associated with the `standard` patch property. A *nonstandard patch* is one that has one of the following characteristics:

- A patch that is associated with the `rebootafter`, `rebootimmediate`, `reconfigafter`, `reconfigimmediate`, or `singleuser` properties. This nonstandard patch can be applied by running the `pprosvc` command or the `smpatch` command in manual mode.
- A patch that is associated with the `interactive` property. Such a patch cannot be applied by using the `smpatch` command.

Use `pprosetup` to schedule patch operations to run in *automatic mode*. Patches are applied based on the policy, which you can set by running `pprosetup`.

Use `pprosetup -p` to specify the types of patches to apply in automatic mode. You can set a policy to apply no patches (`none`) or standard patches (`standard`).

Use `pprosetup -i` to specify the types of patches to apply in *manual* mode. Such patches might include those that require a reboot and those that must be applied while the system is in single-user mode. Specify the types of patches that can be applied by using the following command:

```
# pprosetup -i patch-property-list
```

patch-property-list is a colon-separated list of one or more of the following patch properties:

`interactive`	A patch that cannot be applied by running the usual patch management tools (`pprosvc`, `smpatch`, or `patchadd`). Before this patch is applied, the user must perform special actions. Such actions might include checking the serial number of a disk drive, stopping a critical daemon, or reading the patch's README file.
`rebootafter`	The effects of this patch are not visible until after the system is rebooted.
`rebootimmediate`	When this patch is applied, the system becomes unstable until the system is rebooted. An unstable system is one in which the behavior is unpredictable and data might be lost.
`reconfigafter`	The effects of this patch are not visible until after a reconfiguration reboot (`boot -r`). See the `boot(1M)` man page.
`reconfigimmediate`	When this patch is applied, the system becomes unstable until the system gets a reconfiguration reboot (`boot -r`). An unstable system is one in which the behavior is unpredictable and data might be lost.
`singleuser`	Do not apply this patch while the system is in multiuser mode. You must apply this patch on a quiet system with no network traffic and with extremely restricted I/O activity.
`standard`	This patch can be applied while the system is in multiuser mode. The effects of the patch are visible as soon as it is applied unless the application being patched is running while the patch is applied. In this case, the effects of the patch are visible after the affected application is restarted.

Note – The `smpatch` command only supports the patch policy for manual mode.

Specifying Patch Directories

Use the following options to specify the directories in which to store patch-related data:

- Use the `-b` option to specify the directory in which to store backout data. During a patch backout operation, the data is retrieved from this directory to restore the system to its state prior to applying the patch.
- Use the `-d` option to specify the download directory in which to store patches that are downloaded from the Sun patch server. This directory is also the location from which patches are applied.

- Use the `-q` option to specify the directory in which to store patches that cannot be applied automatically. Such patches are called *sequestered patches*.

  Note – The sequester directory is not used by the `smpatch` command.

Specifying the Hardware on the System

Use the `-H` option to run a program that helps you determine the hardware that is attached to the host system, such as firmware, disk array systems, and tape storage systems.

Use this option to select the hardware that applies to this system. Select the sequence number of the specific hardware. A confirmation page lists the selections.

Save the specified hardware configuration information to a file. Then, the system responds by performing the appropriate actions.

Note – The `smpatch` command does not support this feature for specifying hardware on your system.

Specifying Alternate Configurations

The `pprosetup` command uses a configuration file to specify the collection of patches with which to perform patch operations. By default, all of the patches from the Sun patch server are available for patch operations.

The `-c` option enables you to specify an alternate configuration.

Sun currently provides one alternate configuration, which is called the `recommended` configuration. This configuration includes only those patches that have been declared significant. Such patches include security patches and patches that address known performance and availability problems.

You can use the `-c recommended` option when you schedule patch operations. For example, the following command schedules monthly patch operations that use the `recommended` configuration:

```
# pprosetup -c recommended -M 15 -s 23:30
```

To cancel a schedule that uses the `recommended` configuration, type:

```
# pprosetup -c recommended -C
```

You are permitted to modify the `recommended` configuration by using the `-c` option. See EXAMPLES.

Note – The `smpatch` command does not support this feature for specifying alternate configurations.

Options The following options are supported:

`-a` *admin-email-addr* Is the email address of the patch administrator. Email notification is sent to describe the patches downloaded, the

patches applied, and any error events that occurred when running the `pprosvc -i -n` command.

Note – This option does not affect the `smpatch` command.

`-b` *backout-dir*

Stores backout data in the specified directory.

The backout data is used whenever you use the `patchrm` command to remove a patch that has already been applied to your system. The data is used to restore a system to the state it was in before you applied a particular patch. Since backout data might be quite large, store the data in a large partition that holds large transitory data. Such a partition might be `/var`.

If you do not specify the `-b` option, the backout data is stored in the default locations used by `patchadd`. These locations are the `save` directories of the packages that were modified by the patch. For example, if a patch modifies the `SUNWcsr` package, the backout data for that package is stored in the `/var/sadm/pkg/SUNWcsr/save` directory.

To specify the backout directory, use the `smpatch set` command to set the `patchpro.backout.directory` parameter.

Note – The root file system of any non-global zones must not be referenced with the `-b` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-C`

Clears the existing patch service schedule.

Note – This feature is not supported by the `smpatch` command.

`-c` *config-name*

Uses the *config-name* configuration for patch operations. When this option is included in any `pprosetup` command, the entire command applies to the specified configuration.

Note – This feature is not supported by the `smpatch` command.

`-d` *patch-dir*

Is the directory in which to download the patches that are appropriate for this host system. This directory is also the

location from which patches are applied. By default, the download directory is `/var/sadm/spool`.

Note – To specify the download directory, use the `smpatch set` command to set the `patchpro.download.directory` parameter.

`-D` Schedules the automatic analysis, download, and optional application of patches on a daily basis. This option is equivalent to executing the `pprosvc -i -n` command on a daily basis. See the `crontab(1)` man page.

The policy defined by the `-p` option determines whether no patches (`pprosetup -p none`) are applied or whether standard patches (`pprosetup -p standard`) are applied. By default, no patches are applied.

This option is mutually exclusive with the `-M` option and the `-W` option.

Note – This feature is not supported by the `smpatch` command.

`-h` Displays information about command-line options.

`-H` Establishes a dialog with the user to determine what hardware is attached to the host system.

Note – This feature is not supported by the `smpatch` command.

`-i` [`none` | *patch-property-list*] Specifies the policy for applying patches in manual mode.

No patches are applied when `none` is specified. *patch-property-list* is a colon-separated list of one or more of the following patch properties: `interactive`, `rebootafter`, `rebootimmediate`, `reconfigafter`, `reconfigimmediate`, `singleuser`, and `standard`. See Setting a Patch Policy.

Note – To specify the patch policy, use the `smpatch set` command to set the `patchpro.install.types` parameter.

`-L` Displays the configuration parameter settings of your patch management environment.

This option is mutually exclusive with the other options.

Note – To view the configuration parameter settings, use the `smpatch get` command.

`-M` *day-of-month*

Schedules the automatic analysis, download, and optional application of patches on a monthly basis.

The policy defined by the `-p` option determines whether no patches (`pprosetup -p none`) are applied or whether standard patches (`pprosetup -p standard`) are applied. By default, no patches are applied.

day-of-month is a numerical value from 1-28, which represents the day of the month. Note that the values 29, 30, and 31 are invalid. See the `crontab(1)` man page.

This option is mutually exclusive with the `-D` option and the `-W` option.

Note – This feature is not supported by the `smpatch` command.

`-p` [`none` | `standard`]

Specifies the policy for applying patches in automatic mode.

No patches are applied when `none`, the default, is specified.

When `standard` is specified, only standard patches are applied.

Note – This feature is not supported by the `smpatch` command.

`-P` *patch-source-url*

Is the URL that points to the collection of patches. The default is the Sun patch server, which has the following URL:

`https://updateserver.sun.com/solaris/`

Note – To specify the URL that points to the collection of patches, use the `smpatch set` command to set the `patchpro.patch.source` parameter.

`-q` *sequester-dir*

Is the directory in which patches are moved if they cannot be automatically applied. By default, the sequester directory is `/var/sadm/spool/patchproSequester`.

Note – The sequester directory is not used by the `smpatch` command.

`-s` *hh*`:`*mm*

Optionally sets the time of day to perform patch operations, which by default, is midnight local time.

hh is a value from 00-23, which specifies the hour. *mm* is a value from 00-59, which specifies the minute.

Use this option with the `-D`, `-M`, and `-W` options.

Note – This feature is not supported by the `smpatch` command.

`-u` *user-name*

Is the user name with which to obtain contract patches from Sun.

Store the corresponding SunSpectrum user's password in the `lib/.sunsolvepw` file. If PatchPro is installed in the default location, this file is in the `/opt/SUNWppro` directory.

Keep the password safe by setting the owner, group, and permissions to `root`, `sys`, and `0600`, respectively.

Note – This file method of supplying passwords is no longer supported.

Note – To specify this user, use the `smpatch set` command to set the `patchpro.sun.user` parameter. Also, specify this user's password by setting the `patchpro.sun.passwd` parameter.

`-U` *proxy-user-name*

Is the user name required for authentication of the web proxy, if applicable.

Store the corresponding user's password in the `lib/.proxypw` file. If PatchPro is installed in the default location, this file is in the `/opt/SUNWppro` directory.

Keep the password safe by setting the owner, group, and permissions to `root`, `sys`, and `0600`, respectively.

Note – This file method of supplying passwords is no longer supported.

Note – To specify this user, use the `smpatch set` command to set the `patchpro.proxy.user` parameter. Also, specify this user's password by setting the `patchpro.proxy.passwd` parameter.

`-W` *day-of-week*

Schedules the automatic analysis, download, and optional application of patches on a weekly basis.

day-of-week is a numerical value from 0-6, which represents the day of the week. 0 represents Sunday. See the `crontab(1)` man page.

The policy defined by the `-p` option determines whether no patches (`pprosetup -p none`) are applied or whether standard patches (`pprosetup -p standard`) are applied. By default, no patches are applied.

This option is mutually exclusive with the `-D` option and the `-M` option.

Note – This feature is not supported by the `smpatch` command.

`-x` [*host*`:`*port*]

Specifies the web proxy. If your system is behind a firewall, use this option to specify your web proxy. Get the name of the web proxy and its port from your system administrator or network administrator.

Note – To specify the web proxy host name and port, use the `smpatch set` command to set the `patchpro.proxy.host` and `patchpro.proxy.port` parameters, respectively.

Examples

EXAMPLE 1 Scheduling Daily Patch Operations in Automatic Mode

```
# pprosetup -D
```

Schedules `smpatch update` to run in automatic mode daily at midnight local time.

EXAMPLE 2 Scheduling Weekly Patch Operations in Automatic Mode

```
# pprosetup -W 0 -s 00:45
```

Schedules `smpatch update` to run in automatic mode every Sunday at 12:45 a.m. local time.

EXAMPLE 3 Scheduling Monthly Patch Operations in Automatic Mode

```
# pprosetup -M 15 -s 02:30
```

Schedules `smpatch update` to run in automatic mode on the 15th day of every month at 2:30 a.m. local time.

EXAMPLE 4 Canceling Scheduled Jobs

```
# pprosetup -C
```

Cancels the scheduled jobs that use the default configuration.

EXAMPLE 5 Specifying the Patch Policy for Manual Mode

```
# pprosetup -i standard:singleuser:reconfigafter:rebootafter
```

Specifies the policy for applying patches in manual mode. This policy permits you to apply the following types of patches to your system in manual mode:

- Standard patches
- Patches that must be applied in single-user mode
- Patches that require that the system undergo a reconfiguration reboot after they have been applied
- Patches that require that the system undergo a reboot after they have been applied

EXAMPLE 6 Specifying the Patch Policy for Automatic Mode

```
# pprosetup -p none
```

Specifies that no patches are automatically applied.

```
# pprosetup -p standard
```

Specifies that *only* standard patches can be downloaded and applied.

EXAMPLE 7 Specifying an Alternate Download Directory

```
# pprosetup -d /export/home/patches
```

Specifies that patches are downloaded to the `/export/home/patches` directory.

EXAMPLE 8 Specifying an Alternate Sequester Directory

```
# pprosetup -q /export/home/patches/sequester
```

Specifies that sequestered patches are stored in the `/export/home/patches/sequester` directory.

EXAMPLE 9 Identifying the Hardware on Your System

```
# pprosetup -H
```

Enables a patch analysis to determine whether your system needs specific patches based on your hardware configuration. This command only helps you identify hardware products from Sun Network Storage.

EXAMPLE 10 Configuring Your System to Obtain Contract Patches

```
# pprosetup -u myuser
# echo mypasswd > /opt/SUNWppro/lib/.sunsolvepw
```

Enables your contract user, `myuser`, to obtain the contract patches.

Ensure that the contract user's password is safe by setting the owner, group, and permissions of the `.sunsolvepw` file to `root`, `sys`, and `0600`, respectively.

EXAMPLE 11 Specifying a Web Proxy

```
# pprosetup -x webaccess.corp.net.com:8080
```

Specifies the host name, `webaccess.corp.net.com`, and port, `8080`, of the web proxy to use.

EXAMPLE 12 Scheduling Daily Patch Operations to Use the `recommended` Configuration

```
# pprosetup -c recommended -D -s 23:00
```

Schedules a daily patch analysis that uses the `recommended` configuration. You can use the alternate configuration in conjunction with or in place of a full analysis.

```
# pprosetup -c recommended -C
```

Cancels this job that uses the `recommended` configuration.

EXAMPLE 13 Modifying the `recommended` Configuration

```
# pprosetup -c recommended -a recommended@local
```

Modifies the `recommended` configuration to send email notifications to the `recommended@local` email alias about each scheduled analysis that uses the `recommended` cluster. Any scheduled operation that uses the `recommended` configuration will send notification to the alias you specify.

EXAMPLE 14 Creating a New Configuration

```
# pprosetup -c export -d /export/patches
```

Creates a new configuration named `export` that downloads patches to the `/export/patches` directory. After executing this command, you can schedule patch operations or manually run patch operations that use the `export` configuration by running the `pprosetup` or `pprosvc` commands, respectively.

```
# pprosvc -c export -d
```

Downloads patches to the download directory specified by the `export` configuration.

Attributes

See the `attributes(5)` man page for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpprou
Interface Stability	Obsolete

See Also crontab(1), boot(1M), patchadd(1M), patchrm(1M), pprosvc(1M), smpatch(1M), attributes(5)

Name pprosvc – automation service program for Patch Manager

Synopsis

```
/usr/sbin/pprosvc [-c config-name]
     [-d [-p patch-id [,patch-id]...]] [-h]
     [-i [-n] [-p patch-id [,patch-id]...]] [-l]
```

Description **Note –** This command is deprecated. Use the `smpatch analyze`, `smpatch download`, and `smpatch update` commands instead. See the `smpatch(1M)` man page.

Use the `pprosvc` command to analyze a system to determine the list of appropriate patches, download the patches, and apply them. This command invokes patch operations in response to a user request or at a scheduled time. You must run this command as superuser.

The `pprosvc` command enables you to do the following:

- Analyze the host system for appropriate patches based on an established configuration
- Generate the list of appropriate patches
- Download the patches to your host system from the Sun patch server
- Apply the patches based on a patch policy

Use the `pprosvc -i` command to analyze a system, download the appropriate patches, and apply them. If analysis determines that patches are needed, the `pprosvc` command downloads them and applies them.

Specify other options to automate a subset of the patch management tasks. If you specify the `-d` option, your system is analyzed and the appropriate patches are downloaded to your system. If you specify the `-l` option, your system is analyzed and the appropriate patches are listed.

The list of patches that is generated by the analysis is based on all of the available patches from the Sun patch server. No explicit information about your host system or its network configuration is transmitted to Sun. Only a request for the Sun patch set is transmitted. The patch set is scanned for patches that are appropriate for this host system, the results are displayed, and those patches are optionally downloaded.

The `-d`, `-i`, and `-l` options are mutually exclusive.

Use the `-p` option to specify the patches on which to operate. You can use the `-p` option with the `-d` and `-i` options.

Options The following options are supported:

`-c` *config-name*
: Uses an alternate configuration for the current patch operation. Use the `pprosetup` command to create new configurations.

  A configuration named `recommended` is included. For more information, see "Specifying Alternate Configurations" in the `pprosetup(1M)` man page.

Note – This feature is not supported by the `smpatch` command.

`-d`

Downloads the patches that are appropriate for this host system. The patches are downloaded to the designated download directory.

This option generates a list of appropriate patches, as does the `-l` option. However, instead of just displaying the list of patches, the `-d` option displays and downloads the patches from the Sun patch server. The patches are downloaded using a secure connection, and all patches are authenticated using digital signature technology. Only patches that are signed with a Sun digital signature are stored in your download directory.

Note – Specifying this option is equivalent to running the `smpatch download` command.

`-h`

Displays information about the command-line options.

`-i`

Applies the patches based on the patch policy. This option analyzes your system to generate a list of appropriate patches. If analysis determines that patches are needed, the patches are downloaded and applied. If no patches are permitted to be applied in *automatic* mode (by running `pprosetup -p none`), this option is identical to specifying the `-d` option.

If only standard patches are permitted to be applied in automatic mode (by running `pprosetup -p standard`), all standard patches are applied.

Note – Specifying this option is equivalent to running the `smpatch update` command.

`-l`

Generates a list of the patches that are appropriate for this host system.

Note – Specifying this option is equivalent to running the `smpatch analyze` command.

`-n`

Runs `pprosvc` in *automatic mode*. The command that the `cron` job specifies is `pprosvc -i -n`. To schedule patch operations to run in automatic mode, see the `pprosetup(1M)` man page.

In automatic mode, the patch administrator (specified by the `-a` option) receives email notifications that describe the patches you downloaded and applied, and any error events that occurred.

Do not use this option on the command line.

Standard patches do not require any special actions on the part of the user. Such patches can be applied by using the `patchadd` command (see the `patchadd(1M)` man page) and do not need the host system to reboot for the patch to take effect.

All nonstandard patches are moved to the sequester directory if you use the `-n` option to run in automatic mode. If you run in manual mode, however, nonstandard patches that have properties that match the policy specified by `pprosetup -i` are applied. The rest of the nonstandard patches are moved to the sequester directory. You can apply patches from this directory at a later time. Patches, whether standard or nonstandard, that depend on sequestered patches are not applied under any circumstances. Such patches are placed in the sequester directory.

For any patch that is placed in the sequester directory, refer to the patch's README file to determine how to apply it to your system.

Note – This feature is not supported by the `smpatch` command.

`-p` [*patch-id*[,*patch-id*,...] Designates the specific patches on which to operate. Use this option with the `-d` option or the `-i` option. The list of patch IDs must be separated by commas.

The specified patches are adjusted to use the current versions based on the patch baseline. Patches that are required by the specified patches are added to the complete list of patches to be applied.

Note – Specifying this option is equivalent to specifying the `-i` option to the `smpatch analyze`, `smpatch download`, and `smpatch update` commands.

Examples

EXAMPLE 1 Applying Specific Patches in Manual Mode

```
# pprosvc -i -p 102893-01,106895-09,106527-05
```

Applies patches 102893-01, 106895-09, and 106527-05 to the local system in manual mode.

EXAMPLE 2 Analyzing a System and Downloading Appropriate Patches

```
# pprosvc -i
```

EXAMPLE 2 Analyzing a System and Downloading Appropriate Patches *(Continued)*

Performs an analysis of the current system and downloads the appropriate patches based on all the patches from the Sun patch server. The resulting list of patches can be very long.

EXAMPLE 3 Applying Patches From the Recommended Configuration

```
# pprosvc -c recommended -i
```

Uses the `recommended` configuration to perform an analysis of the current system and downloads the appropriate patches. Standard patches and those needed from the `recommended` configuration are applied to the system based on the established patch policy. For information about setting the patch policy for manual mode, see the description of the `-i` option on the `pprosetup(1M)` man page.

Attributes

See the `attributes(5)` man page for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpprou
Interface Stability	Obsolete

See Also

`patchadd(1M)`, `patchrm(1M)`, `pprosetup(1M)`, `smc(1M)`, `smpatch(1M)`, `attributes(5)`

Name praudit – print contents of an audit trail file

Synopsis

```
praudit [-lrsx] [-ddel] [filename]...
```

Description `praudit` reads the listed *filename*s (or standard input, if no *filename* is specified) and interprets the data as audit trail records as defined in `audit.log(4)`. By default, times, user and group IDs (UIDs and GIDs, respectively) are converted to their ASCII representation. Record type and event fields are converted to their ASCII representation. A maximum of 100 audit files can be specified on the command line.

Options The following options are supported:

`-d`*del*
: Use *del* as the field delimiter instead of the default delimiter, which is the comma. If *del* has special meaning for the shell, it must be quoted. The maximum size of a delimiter is three characters. The delimiter is not meaningful and is not used when the `-x` option is specified.

`-l`
: Print one line per record.

`-r`
: Print records in their raw form. Times, UIDs, GIDs, record types, and events are displayed as integers. This option is useful when naming services are offline. The `-r` option and the `-s` option are exclusive. If both are used, a format usage error message is output.

`-s`
: Display records in their short form. Numeric fields' ASCII equivalents are looked up by means of the sources specified in the `/etc/nsswitch.conf` file (see `nsswitch.conf(4)`). All numeric fields are converted to ASCII and then displayed. The short ASCII representations for the record type and event fields are used. This option and the `-r` option are exclusive. If both are used, a format usage error message is output.

`-x`
: Print records in XML form. Tags are included in the output to identify tokens and fields within tokens. Output begins with a valid XML prolog, which includes identification of the DTD which can be used to parse the XML.

Files

`/etc/security/audit_event`
: Audit event definition and class mappings.

`/etc/security/audit_class`
: Audit class definitions.

`/usr/share/lib/xml/dtd`
: Directory containing the verisioned DTD file referenced in XML output, for example, `adt_record.dtd.1`.

`/usr/share/lib/xml/style`
: Directory containing the versioned XSL file referenced in XML output, for example, `adt_record.xsl.1`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	See below

The command stability is evolving. The output format is unstable.

See Also bsmconv(1M), getent(1M), audit(2), getauditflags(3BSM), getpwuid(3C), gethostbyaddr(3NSL), ethers(3SOCKET), getipnodebyaddr(3SOCKET), audit.log(4), audit_class(4), audit_event(4), group(4), nsswitch.conf(4), passwd(4), attributes(5)

See the section on Solaris Auditing in *System Administration Guide: Security Services*.

Notes This functionality is available only if the Solaris Auditing feature has been enabled. See bsmconv(1M) for more information.

Name printmgr – Solaris Print Manager is a graphical user interface for managing printers in a network

Synopsis

```
/usr/sadm/admin/bin/printmgr
```

Description Solaris Print Manager is a Java-based graphical user interface that enables you to manage local and remote printer access. This tool can be used in the following name service environments: LDAP, NIS, NIS+, and files. You must be logged in as superuser to use this tool.

Using Solaris Printer Manager is the preferred method for managing printer access because Solaris Print Manager centralizes printer information when it is used in a name service environment.

Adding printer information to a name service makes access to printers available to all systems on the network and generally makes printer administration easier because all the information about printers is centralized.

Solaris Print Manager may be run on a remote system with the display sent to the local system. See the *System Administration Guide: Printing* for instructions on setting the `DISPLAY` environment variable.

Using Solaris Print Manager to perform printer-related tasks automatically updates the appropriate printer databases. Solaris Print Manager also includes a command-line console that displays the `lp` command line for the add, modify, and delete printer operations. Errors and warnings may also be displayed when Printer Manager operations are performed.

Help is available by clicking the Help button.

Usage Solaris Print Manager enables you to do the following tasks:

Select a Name Service	Select a name service for retrieving or changing printer information.
Add Access to a Printer	Add printer access on a printer client using Solaris Print Manager.
Add an Attached Printer	After physically attaching the printer to a system, use Solaris Print Manager to install a local printer and make it available for printing.
Add a Network Printer	After physically attaching the printer to a system, use Solaris Print Manager to install a local printer and make it available for printing.
Modify Printer Properties	After adding access to a printer or adding an attached or network printer, you can modify certain printer attributes.
Delete a Printer	Delete access to a printer from the print client or delete a printer from the print server or from the name service environment.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWppm

See Also `ldap(1)`, `lpget(1M)`, `lpset(1M)`, `attributes(5)`

System Administration Guide: Printing for information on LDAP server replication.

Although users can use the LDAP command line utilities `ldapadd(1)` and `ldapmodify(1)`to update printer entries in the directory, the preferred method is to use `lpset`. Otherwise, if the `lpadd` and `lpmodify` utilities are used, the administrator must ensure that the `printer-name` attribute value is unique within the `ou=printers` container on the LDAP server. If the value is not unique, the result of modifications done using `lpset` or the Solaris Print Manager, `printmgr` may be unpredictable.

Name privatepw – administer FTP Server enhanced group access file

Synopsis

```
privatepw [-c] [-f ftpgroups] [-g real_group_name] accessgroup

privatepw -d [-f ftpgroups] accessgroup

privatepw -l [-f ftpgroups]

privatepw -V
```

Description The `privatepw` utility is an administrative tool to add, delete and list enhanced access group information in the `ftpgroups` file. See `ftpgroups(4)`.

When `privatepw` is used without options, the help usage message is displayed. The `privatepw` utility prompts for a password when adding an enhanced access group entry or modifiying an existing one.

Options The following options are supported by the `privatepw` utility:

`-c`	Create a new `ftpgroups` file for the specified *accessgroup*.
`-d`	Delete information about the specified *accessgroup* from the `ftpgroups` file.
`-f` *ftpgroups*	Use the specifed `ftpgroups` file for all updates.
`-g` *group*	Set the real system group to the *group* specified. *group* is a valid group name returned by `getgrnam(3C)`. If the real system group is not supplied with the `-g` option when adding an ehanced access group entry, the `privatepw` utility prompts for it.
`-l`	List the contents of the `ftpgroups` file.
`-V`	Display program copyright and version information, then terminate.

Operands The following operands are supported:

accessgroup	The name of the enhanced access group to create or update It consists of an arbitrary string of alphanumeric and punctuation characters. See `ftpgroups(4)`.

Exit Status The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

Files

```
/etc/ftpd/ftpgroups

/etc/group
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu
Interface Stability	External

See Also in.ftpd(1M), getgrnam(3C), ftpgroups(4), attributes(5)

Name prodreg – Solaris Product Registry administration

Synopsis

```
prodreg [--help] | [subcommand operand ...]
```

Description The `prodreg` utility browses, unregisters, and uninstalls components in the Solaris Product Registry.

Some installers make use of the `libwsreg(3LIB)` interface to register information. The Solaris Product Registry contains information about this installed software.

The database associated with the Solaris Product Registry is relative to the root of the installed file system. Normally, this is the root of the file system (`/`). Sometimes, an alternate root, with a different Solaris Product Registry install database is used, as during live upgrade installation. See `live_upgrade(5)`.

The Registry database informs installers about installed software. The Registry and the `prodreg` utility do not directly perform installation or deinstallation. `prodreg` supports installers which are executed externally and launched by the `prodreg` utility or other means.

Depending on the subcommand, the `prodreg` command offers equivalent functions from the command line or a GUI viewer. Two versions of the GUI viewer are available. The default is the Java Swing GUI. The other version, the Java awt GUI is provided for environments without Java Swing support.

The only feature which exists in the CLI which is not present in the GUI is the `unregister` subcommand. It is possible for the product registry to become corrupted, if for example, some software is removed manually instead of by means of an uninstaller program. These entries can confuse installers which are run subsequently. The `unregister` subcommand allows stale entries to be removed, even forcefully. Care should be exercised when unregistering software with the recursive or force options so that valid entries in the registry are not removed by mistake.

The `prodreg` command, whether it launches the GUI or the command line interface browser, displays the contents of the registry at that time only. If software is installed or uninstalled subsequent to or concurrent with launching either `prodreg` viewer, the view can be inconsistent with the Solaris Product Registry.

Subcommands You can specify options to the `prodreg` command without specifying a subcommand. If the subcommand is omitted, the `swing` subcommand is assumed.

The following subcommands are supported:

awt — Launch the Java `awt` GUI.

The `awt` subcommand has the following format:

```
awt [-R alt_root | --help]
```

browse

Display the Solaris Product Registry using a command line interface. The text output of this command displays identifying information of any component in the product registry tree, including its ancestors and children. If you repeatedly invoke this subcommand, you can interactively browse the product registry.

The database components are related as a tree. Components may have one or more children. Except for the root, components have one parent. This subcommand displays the ancestors and children for a given component in the Solaris Product Registry database.

Each time the `prodreg browse` subcommand is executed, one component in the Registry is shown, along with its ancestry to the root of the Registry, as well as the component's children. To browse in the `prodreg` GUI, a user selects a node to expand and clicks on it. The analogous activity using the command line interface is to browse on children of nodes successively, which effectively expands a view into the registry.

Start by browsing the root of the Registry with `prodreg browse`. Select components to expand the scope of the browsing activity. Use browse numbers as a convenience during this interactive browsing, but not in scripts. Browse numbers can change from one session to the next or on different systems. This is because browse numbers are generated as they are first used, by a given user on a particular system.

The `browse` subcommand has the following format:

```
browse [-R alt_root] [-u uuid [-i instance | -p location]]
browse [-R alt_root] -n bnum [-i instance | -p location]
browse [-R alt_root] -m name
browse --help
```

This following information is output for each component:

BROWSE # — This is the browse number associated with each component. This number can be used as an argument to either the `prodreg browse` or `info` subcommands as a convenience

+/-/. — The + indicates a component in the tree with children who are not shown. - indicates a component with children of which at least one child is being shown. The . indicates a component which has no children. This field is

arranged so that each space (reading left to right) depicts a successive generation.

UUID — This is the component's unique identifier.

— This is the instance number of the component. Software components can be installed multiple times. The software registry assigns a unique instance to each one.

NAME — Each component in the Solaris Product Registry database has a localized name which is displayed in this field. It is possible that this name may not be unique in the registry since there could be another component that has the same name.

The `browse` subcommand provides four distinct options for viewing the registry database. If multiple instances are associated with the same component, then the output of the subcommand is the ambiguous list. The request must be made unambiguous. The instance or location operands can be used to disambiguate the browse subcommand when used with the `-u` or `-n` options.

- If no operand information is given, the root of the registry tree is displayed, as well as its children. This is the starting point for interactive browsing of the entire registry database.
- If the browse number is given, the component associated is output.
- If the `uuid` is given, the component associated with it is output.
- If the name is given, the component associated with it is output.

info — Display attributes for any component in the Solaris Product Registry by supplying identifying information for the component.

Components in the product registry are associated with attributes. These attributes are composed of a *name* and a single *value* string.

This subcommand outputs attribute information associated with components in the Solaris Product Registry. Individual components in the product registry are specified as for the `browse` subcommand, except that either the *uuid*, *name* or *bnum* must be specified.

If a component requested is ambiguous as it has more than one instance or the name is assigned to more than one component in the registry, the list of possibilities is output, not the attribute information.

The default output of this subcommand is a complete list of each attributes, each on a new line. The attribute name is followed by a colon (`:`) and a SPACE. The attribute value follows, after which a RETURN is appended. Other options include can be specified using `-a` and `-d`.

The `info` subcommand has the following format:

```
info --help
info [-R alt_root] -u uuid [-i instance | -p location]
info [-R alt_root] -n bnum [-i instance | -p location]
info [-R alt_root] -m name [-a attr | -d ]
```

help | `--help` | `-?`

Display help text.

The `help` subcommand has the following format:

```
help | --help | -?
```

swing

Launch the Java Swing GUI. If the Java Swing GUI is not available, this subcommand fails.

The `swing` subcommand has the following format:

```
swing  [-R alt_root | --help]
```

version | `--version` | `-V`

Outputs a current version string.

The `version` subcommand has the following format:

```
version | --version | -V
```

unregister

Unregister an entry in the registry.

Remove a component from the Solaris Product Registry. The component corresponding to the *uuid* specified with the `-u` option must be a single instance. If it is not, the subcommand fails and returns the list of instances with the associated *uuid*. The subcommand must be reissued using either `-p` or `-i` to uniquely determine which component instance to unregister.

The `unregister` subcommand fails if there are components in the registry which depend on the component which is to be unregistered.

The `unregister` subcommand fails if the user does not have write access to the registry. See `wsreg_can_access_registry(3WSREG)`. The `unregister` subcommand fails if the user attempts to unregister a system component, instead of a component registered with the Solaris Product Registry. System components include those which include the attribute `PKG` and certain special Registry nodes including the following:

```
UUID                                   Name
====================================  ============================
root                                   System Registry
a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  Solaris System Software
8f64eabf-1dd2-11b2-a3f1-0800209a5b6b  Unclassified Software
b96ae9a9-1dd1-11b2-a3f2-0800209a5b6b  System Software Localizations
b1c43601-1dd1-11b2-a3f2-0800209a5b6b  Additional System Software
a8dcab4f-1dd1-11b2-a3f2-0800209a5b6b  Software Localizations
```

Before the `unregister` subcommand with the `-f` option is used, you should carefully review what components depend upon the component which is to be unregistered. The `-r` option is even more dangerous, since all children and software components depending upon the component are also deregistered. You can obtain the list of dependent components for a component with UUID *uuid* using :

```
prodreg info -u uuid -a "Dependent Components"
```

You can obtain a list of required components using:

```
prodreg info -u <uuid> -a "Required Components"
```

The output lists the name, UUID and instance of the component.

The `unregister` subcommand has the following format:

```
unregister [-R alt_root] [-fr] -u uuid [-p location | -i instance]
unregister --help
```

uninstall

Launch an uninstaller program.

Each component in the registry can have an uninstaller associated with it. This subcommand executes this associated installer, if there is one, for a component in the registry given by the `-u` option. If there is no uninstaller associated with the component, the subcommand fails. If the component given by the `-u` option is not unique (as there is more than one instance of the component installed), the subcommand outputs a list of all instances. The subcommand must then be reissued using `-i` or `-p`

to disambiguate the uuid given with the `-u` option. Finally, if the component to uninstall is depended upon by other components, the command fails.

The command may also launch an uninstaller with a `-x` option. No checks for whether this uninstalls a component upon which other components depend in this case.

The `uninstall` command is not executed if the user does not have write access to the registry. See `wsreg_can_access_registry(3WSREG)`.

The `uninstall` command has the following format:

```
uninstall [-R alt_root] [-f] -u uuid -p location
uninstall [-R alt_root] -i instance[arguments ...]
uninstall --help
```

Options The `awt` subcommand supports the following options:

`--help` Display help text, do not launch the viewer.

`-R` *alt_root* Use the specified alternate root to locate the database to display with the GUI viewer.

See OPERANDS for information regarding specification of *alt_root*.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

The `browse` subcommand supports the following options:

`-help` Display help text, do not execute the browse subcommand.

`-i` *instance* Output the specified component instance.

`-m` *name* Output the component instances associated with the name.

`-n` *bnum* Output the component instances associated with the browse number.

`-p` *location* Output the component instance installed in the specified location. The install location for a component can be obtained using the 'info' subcommand.

`-R` *alt_root* Use the specified alternate root to locate the database.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-u` *uuid* Output the component instances associated with the uuid.

The `info` subcommand supports the following options:

`-a` *attr* Output only the attribute whose name is given by the operand 'attr', instead of all attributes of the specified component.

`-d` Output only the attribute whose name is `isDamaged`, instead of all attributes of the specified component. If the value is set to `true`, this attribute indicates that the component in the registry

`--help` Output help text, do not execute the browse subcommand.

`-i` *instance* The instance operand distinguishes among multiple instances of components with the same `uuid` or browse number.

`-m` *name* The name operand indicates one or more components in the registry.

`-n` *bnum* Output the attributes of the component instance associated with the browse number bnum. If there is more than one instance, the command must be disambiguated using the `-i`or `-p` options.

`-p` *location* The install location indicated distinguishes among multiple instances of components with the same uuid or browse number.

`-R` *alt_root* Use the specified alternate root to locate the database.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-u` *uuid* Output the attributes of the component instance associated with the uuid. If there is more than one instance, the subcommand must be disambiguated using the `-i` or `-p` options.

The `swing` subcommand supports the following options:

`--help` Output help text, do not execute the install subcommand.

`-R` *alt_root* Use the specified alternate root to locate the database.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global

zone's file system. See zones(5).

The uninstall subcommand supports the following options:

-f — Force the uninstall. A forced subcommand uninstalls all instances of a component, even if there are multiple ambiguous instances of the uuid operand.

--help — Output help text, do not execute the unregister subcommand.

-i *instance* — Disambiguate the *uuid* operand.

-p *location* — Disambiguate the *uuid* operand. *location* corresponds to the where the software component was installed.

-R *alt_root* — Use the specified alternate root to locate the database.

Note – The root file system of any non-global zones must not be referenced with the -R option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See zones(5).

-u *uuid* — Unregister the *uuid* component. If this component has been installed multiple times, the instance to unregister must be indicated unambiguously by using the -i or -p option.

The unregister subcommand supports the following options:

-f — Force the unregistration. A forced subcommand unregisters a component even if there are other components which are dependent on this component.

--help — Output help text, do not execute the unregister subcommand.

-i *instance* — Disambiguate the *uuid* operand.

-p *location* — Disambiguate the uuid operand. The *location* corresponds to the where the software component was installed.

-r — Causes a recursive deregistration of a component as well as that component's children and dependencies.

-R *alt_root* — Use the specified alternate root to locate the database.

Note – The root file system of any non-global zones must not be referenced with the -R option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See zones(5).

-u *uuid* — Unregister component *uuid* of the component to unregister. If this component has been installed multiple times, the instance to unregister must be indicated unambiguously by using the -i or -p option.

Operands The following operands are supported:

alt_root Pathname to a file indicating an alternate root. The Solaris Product Registry database is located relative to the alternate root. If database relative to this location does not exist, it is created.

Note – The root file system of any non-global zones must not be referenced by *alt_root*. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

attr Name of an attribute. This operand is used only with the `info` subcommand. If *attr* is associated with a component, the attribute name and value is displayed.

bnum The browse number.

Each component in the Solaris Product Registry is associated with a browse number. This number is generated for the convenience of an interactive user. The browse number can change if the system is rebooted or reinstalled. Do not store or use the browse number except to facilitate the `browse` and `info` subcommands. Browse numbers are always output by the `prodreg browse` subcommand. Only these values can be used as input values to the browse or info subcommand.

instance Software can be installed in more than one location. The Solaris Product Registry associates a unique instance number for each. The browse subcommand shows the instance number associated with each component in the registry. The instance operand is used to distinguish between installed, and possibly different, copies of software, when such exist.

location A path to a specific file or directory in the file system. This operand indicates the installed location of registered software. For instance, if software is installed relative to `/usr/local` the value of this operand would be `/usr/local`. The install location is used to installer or to indicate the location of an installer or to disambiguate which instance is intended, of a software component which can have multiple instances.

name Each software component in the Solaris Product Registry is associated with a name. This name is output by the `browse` subcommand. Some subcommands allow the user to input the software by name as an operand as a convenience. These names might not be unique. If the user supplies an ambiguous name, for which more than one components exist, the subcommand outputs a list of possible choices. The name can be localized; depending on the language setting the name can differ.

uuid Each software component in the Solaris Product Registry is associated with a unique identifier. This identifier is a handle which accesses an entry in the registry database. The uuid corresponds to the component irrespective of how many

instances of the component have been installed, and what the localized name of the component is.

Examples

EXAMPLE 1 Using the `prodreg` Command to Browse

Browsing is performed by means of the `prodreg` browse subcommand. Using these requests iteratively, one can peruse the tree, much as one would using a GUI by expanding components which are collections of other components. Browsing using browse numbers for convenience should be done only during this iterative browsing process, since the numbers are generated as a result of the browsing operation.

Evoking the `browse` subcommand without any arguments browses from the top of the registry. The output varies depending on the software installed on a particular system.

```
$ prodreg browse
BROWSE # +/-/.  UUID                                 #  NAME
======== =====  ===================================  =  ============
1        -      root                                 1  System
                                                       Registry
2         +     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                       System
                                                       Software
3         +     8f64eabf-1dd2-11b2-a3f1-0800209a5b6b  1  Unclassified
                                                       Software
```

The output of this command lists the browse number, UUID, instance number and name of the root component and its children. The ancestors of a component, each parent up to the root, are also shown. The `+/-/.` column indicates whether the component in the tree is an expanded parent (`-`), a child with children (`+`) or a child without children (`.`).

EXAMPLE 2 Requesting Information About the Components in a Tree

The UUID, name and browse number fields can be used to request browsing information about components in the tree. The next example shows how a component can be browsed by UUID.

```
$ prodreg browse -u a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b
BROWSE #  +/-/.  UUID                                 #  NAME
========  =====  ===================================  =  ===========
1         -      root                                 1  System
                                                       Registry
2          -     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                       System
                                                       Software
4           +    b96ae9a9-1dd1-11b2-a3f2-0800209a5b6b  1  System
                                                       Software
                                                       Localizations
5           +    SUNWCall                              1  Entire
```

EXAMPLE 2 Requesting Information About the Components in a Tree *(Continued)*

```
                                                                Distribution
```

EXAMPLE 3 Browsing a Node by Name

The following example shows how a node can be browsed by name.

```
$ prodreg browse -m "System Software Localizations"
BROWSE #  +/-/.  UUID                                 #  NAME
========  =====  ====================================  =  ===========
1         -      root                                  1  System
                                                        Registry
2          -     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                        System
                                                        Software
4           -    b96ae9a9-1dd1-11b2-a3f2-0800209a5b6b  1  System
                                                        Software
                                                        Localizations
316          .   SUNWceuow                            1  Central
                                                        Europe OW
                                                        Support
317          .   SUNWcsfw                             1  Simplified
                                                        Chinese
                                                        freeware
                                                        message
318          .   SUNWceuox                            1  Central
                                                        Europe
                                                        64-bit OS
                                                        Support
```

EXAMPLE 4 Browsing Iteratively

Additional output has been omitted. As a convenience, the browse number can be used for iterative browsing. This number should not be stored, as it differs depending on which system the `prodreg` command is run on, which user is running the command, and the log in session in which the command is run.

```
$ prodreg browse -n 3

BROWSE #  +/-/.  UUID                                 #  NAME
========  =====  ====================================  =  ===========
1         -      root                                  1  System
                                                        Registry
2          -     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                        System
                                                        Software
5           -    SUNWCall                              1  Entire
                                                        Software
```

EXAMPLE 4 Browsing Iteratively *(Continued)*

```
                                                        Distribution
6            .   SUNWrsmo                           1   RSMPI
                                                        Operations
                                                        Registration
                                                        Module
7            +   SUNWCjvx                           1   JavaVM
                                                        (64-bit)
8            .   SUNWrsmx                           1   Remote
                                                        Shared
                                                        Memory
                                                        (64-bit)
9            +   SUNWCacc                           1   System
                                                        Accounting
```

EXAMPLE 5 Browsing Using an Ambiguous Value

If the requested value is ambiguous, the list of ambiguous instances are displayed. In the following example, there are two distinct software components with the same name.

```
$ ./prodreg browse -m JavaVM
The request failed because multiple components correspond to the
criteria given. Use the list of possible components given below,
select one and try again.

BROWSE #  +/-/.  UUID                                #  NAME
========  =====  ==================================  =  ==========
12        .      org.spybeam.javavm                  1  JavaVM
51        .      SUNWCjv                             1  JavaVM
```

Issue one of the following requests again:

```
$ prodreg browse -u SUNWCjv
```

or

```
$ prodreg browse -u org.spybeam.javavm
```

EXAMPLE 6 Browsing Multiple Installations of Software

Another possible ambiguous response arises when a particular software component is installed multiple times. In the example below `Example` software is registered three times.

```
$ prodreg browse -m Example
The request failed because multiple components correspond to the
criteria given. Use the list of possible components given below,
select one and try again.
BROWSE #  +/-/.  UUID                                #  NAME
========  =====  ==================================  =  ==========
```

EXAMPLE 6 Browsing Multiple Installations of Software *(Continued)*

```
7         .      org.spybeam.example                  2  Example
7         .      org.spybeam.example                  3  Example
7         .      org.spybeam.example                  1  Example
The component requested could not be found.
```

EXAMPLE 7 Browsing Using a Particular Instance

The request can be repeated specifying a particular instance to disambiguate it. It is also possible to disambiguate a request with the `-p` option, followed by the install location. In this case, to browse the first instance of the `Example` software, one would use the command:

```
$ prodreg browse -u org.spybeam.example -i 1
```

EXAMPLE 8 Using the `info` Subcommand

The install location, as well as other attributes of a component can be obtained with the info subcommand. The info subcommand accepts the same disambiguating options and returns all the attributes of a component, each on a single line.

```
$ prodreg info -m Example
The request failed because multiple components correspond to the
criteria given. Use the list of possible components given below,
select one and try again.
BROWSE # +/-/. UUID                                 # NAME
======== ===== ==================================== = ===========
7        .     org.spybeam.example                  2 Example
7        .     org.spybeam.example                  3 Example
7        .     org.spybeam.example                  1 Example
The component requested could not be found.
```

This variation of the info subcommand outputs all information associated with instance 1 of the `Example` component. The output from this variation is not displayed

```
$ prodreg info -u org.spybeam.example -i 1
```

EXAMPLE 9 Obtaining Information on the Install Location

You can use the `info` subcommand to obtain the install location and other attributes of a component. The `info` subcommand accepts the same disambiguating options as the `browse` subcommand. It returns all the attributes of a component, each on a single line. You can also request a single attribute.

The following command outputs the value of the install location attribute:

```
$ prodreg info -n 23 -a Location
```

EXAMPLE 10 Idenitifying and Unregistering Damaged Software

Removing installed software without using the associated uninstaller can damage the software in the registry. A damaged component indicates that certain software is installed, when in fact it is not present. A component can be damaged by removing files or packages directly, without running the associated uninstaller. The general rule to follow is: If software has been installed by an installer program, it should be uninstalled using the supplied uninstaller program.

This example shows how to identify and repair damaged software components so that software can be reinstalled.

Browsing for `Examplesoft`, produces the following:

```
$ prodreg browse -m Examplesoft
BROWSE #  +/-/.  UUID                                  #  NAME
========  =====  ====================================  =  ============
1         -      root                                  1  System
                                                          Registry
2          +     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                          System
                                                          Software
3          +     8f64eabf-1dd2-11b2-a3f1-0800209a5b6b  1  Unclassified
                                                          Software
4          -     95842091-725a-8501-ef29-0472985982be  1  ExampleSoft
233         .    90209809-9785-b89e-c821-0472985982be  1  Example Doc
234         .    EXSOzzt                               1
235         .    EXSOblob                              1  Example Data
```

The `Examplesoft` child `EXSOzzt`, representing a package component of registered software does not display its name. This is likely to be because the software `Examplesoft` is damaged. Verify this with the following command:

```
$ prodreg info -u 95842091-725a-8501-ef29-0472985982be \
     -i 1 -d
isDamaged=TRUE
```

Since `Damaged` is `TRUE`, some part of `Examplesoft` is damaged. The following command lists the packages which make up `Examplesoft`:

```
$ prodreg info \
     -u 95842091-725a-8501-ef29-0472985982be\
     -i 1 -a PKGS pkgs:
EXSOzzt EXSOblob
```

Use the `pkginfo` command to verify if `EXSO` is installed:

```
$ pkginfo EXSOzzt
ERROR: information for "EXSOzzt" was not found
```

EXAMPLE 10 Idenitifying and Unregistering Damaged Software *(Continued)*

```
$ pkginfo EXSOblob
application EXSOblob       Example Data
```

The output of these commands shows that the package `EXSOzzt` has been removed, probably with the `pkgrm` command. The `Examplesoft` software will probably not function. To repair the software, one should run the uninstaller registered with `Examplesoft`. You probably need to run the uninstaller with root permissions, as it unregisters the software and runs `pkgrm` commands. Both of these operations require root permissions.

```
# prodreg uninstall -u 95842091-725a-8501-ef29-0472985982be -i 1
The install program requested could not be found.
```

Something is wrong, or else you would be able to access uninstall program to uninstall the software. One possibility is that the uninstaller program has been removed manually. It is possible to determine where the uninstaller is located by requesting the `uninstallprogram` attribute:

```
$ prodreg info -m ExampleSoft -a uninstallprogram
uninstallprogram: /usr/bin/java -mx64m -classpath
/var/sadm/prod/org.example.ExampleSoft/987573587 uninstall_ExampleSoft
```

Check to see if there is an uninstaller in the registered location.

```
# ls /var/sadm/prod/org.example.ExampleSoft/987573587
/var/sadm/prod/org.example.ExampleSoft/987573587:
No such file or directory
```

Since there is no uninstaller at the desired location, you have two options. One is to load the uninstaller from back up storage and run it manually. Use the command line stored in the registry:

```
# /usr/bin/java -mmx64m -classpath \
     /var/sadm/prod/org.example.ExampleSoft/987573587 \
     uninstall_ExampleSoft
```

If there is no other possibility, manually unregister the software.

```
# prodreg unregister -u 95842091-725a-8501-ef29-0472985982be -i 1
```

This does not remove the remaining package `EXSOblob`. You must do this manually.

```
# pkgrm EXSOblob
```

EXAMPLE 11 Removing Multiple Components

Component A has children B and C, and C has children D and E, and the you wish to remove all of the components at once. This is useful if the whole hierarchy has to be reinstalled and the uninstaller has been lost or cannot be run

EXAMPLE 11 Removing Multiple Components *(Continued)*

```
$ prodreg browse -u UUID-of-C
BROWSE #  +/-/.  UUID                                 #  NAME
========  =====  ====================================  =  ============
1         -      root                                  1  System
                                                          Registry
2          +     a01ee8dd-1dd1-11b2-a3f2-0800209a5b6b  1  Solaris 10
                                                          System
                                                          Software
3          +     8f64eabf-1dd2-11b2-a3f1-0800209a5b6b  1  Unclassified
                                                          Software
1423       -     UUID-of-A                             1  Example A
1436        .    UUID-of-B                             1  Example B
1437        -    UUID-of-C                             1  Example C
1462         .   UUID-of-D                             1  Example D
1463         .   UUID-of-E                             1  Example E

# prodreg uninstall -u UUID-of-A -i 1
```

The `uninstall` subcommand can fail various ways, for example if the java classes have been removed, if the user has insufficient permissions or if Java software is not present on the system. The recursive unregistration subcommand is very powerful and dangerous. Not only does it unregister every child of a component, it also unregisters every component which depends upon the component to unregister. It is a good idea to view all information about the component to determine if any components will be unintentionally unregistered with UUID-of-A.

```
$ prodreg info -u UUID-of-A
Title: Example A Software
Version: 5.8.0.2001.11.02
Location: /usr
Vendor: Example Vendor
uninstallprogram: /usr/bin/java -mx64m -classpath
/var/sadm/prod/org.example.ExampleA/90820965 uninstall_ExampleA
vendorurl: http://www.example.org
description: Example A Software has many uses
Supported Languages: en

Child Components:
Name                        UUID                                  #
--------------------------  ------------------------------------  -
Example B                   UUID-of-B                             1
Example C                   UUID-of-C                             1

Required Components:
Name                        UUID                                  #
```

EXAMPLE 11 Removing Multiple Components *(Continued)*

```
-------------------------  ----------------------------------  -
Example B                  UUID-of-B                           1
Example C                  UUID-of-C                           1
```

No software depends on Example A, or else an additional field, `Dependent Components` would be shown. To further ensure that there are no surprises, one should examine the dependent components and children of UUID-of-B and UUID-of-C, all the components which depend on UUID-of-B, UUID-of-C and their children, and so on.

If you examine the browse tree, you know the entire list of descendents of UUID-of-A. You can also examine the dependent component attributes of all of Example A's descendents.

```
$ prodreg info -u UUID-of-B -i 1 -a "Dependent Components"
Dependent Components:
Name                        UUID                                #
--------------------------  ----------------------------------  -
Example A                   UUID-of-A                           1

$ prodreg info -u UUID-of-C -i 1 -a "Dependent Components"
Dependent Components:
Name                        UUID                                #
--------------------------  ----------------------------------  -
Example A                   UUID-of-A                           1

$ prodreg info -u UUID-of-D -i 1 -a "Dependent Components"
Dependent Components:
Name                        UUID                                #
--------------------------  ----------------------------------  -
Example C                   UUID-of-C                           1

$ prodreg info -u UUID-of-E -i 1 -a "Dependent Components"
Dependent Components:
Name                        UUID                                #
--------------------------  ----------------------------------  -
Example C                   UUID-of-C                           1
```

A recursive unregistration of Example A only results in unregistering Example A and its descendents, as intended.

```
# prodreg unregister -r -u UUID-of-A -i 1
```

EXAMPLE 12 Reinstalling a Damaged Component

In this example, there is a component, Software ZZZ which is depended upon by other software. Software ZZZ has been damaged and you need to reinstall it. The reinstallation is impossible until Software ZZZ is unregistered.

EXAMPLE 12 Reinstalling a Damaged Component *(Continued)*

First, you check what depends upon Software ZZZ:

```
$ prodreg info -m "Software ZZZ" -a "Dependent Components"
Dependent Components:
Name                         UUID                                  #
---------------------------  ------------------------------------  -
Software Foobar              d9723500-9823-1432-810c-0100e09832ff  1
```

Normally, you would have to uninstall Software Foobar before unregistering Software ZZZ, since Software Foobar depends on Software ZZZ. You decide that it is impossible or unreasonable to reinstall Software Foobar. Performing a recursive unregister of Software ZZZ is not an option as it would unregister Software Foobar as well. Instead you can do a forced unregister of Software ZZZ. The UUID of Software ZZZ is 90843fb1-9874-3a20-9b88-984b32098432.

```
# prodreg unregister -f -u 90843fb1-9874-3a20-9b88-984b32098432 -i 1
```

You can then reinstall Software ZZZ:

```
# /usr/bin/java -cp /usr/installers/org.example.softwarezzz
```

Bugs

The registry can become out of date because of software being manually removed, or removed using `pkgrm(1M)` directly. To avoid damaging the registry, use uninstall programs to remove software which was initially installed using an install program.

Environment Variables

The following environment variable affects the execution of `prodreg`:

`PKG_INSTALL_ROOT` If present, defines the full path name of a directory to use as the system's `PKG_INSTALL_ROOT` path. All product and package information files are then looked for in the directory tree, starting with the specified `PKG_INSTALL_ROOT` path. If not present, the default system path of `/` is used.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwsr2, SUNWwsrv
Interface Stability	Evolving

See Also pkgadd(1M), pkgrm(1M), wsreg_can_access_registry(3WSREG), libwsreg(3LIB), live_upgrade(5), attributes(5)

Application Packaging Developer's Guide

Notes The prodreg GUI and command line interface view both the Solaris Product Registry and the package database. Both look like components in the registry, but some of these cannot be unregistered or uninstalled. Packages do not have an associated uninstaller, so they cannot be uninstalled using the prodreg uninstall subcommand. Solaris packages cannot be unregistered using the prodreg unregister subcommand. Packages are removed using the pkgrm(1M) command, after which time the packages do not appear in the GUI or CLI prodreg viewer.

It is preferable to remove software using the uninstaller associated with the software installed than to remove individual packages using pkgrm(1M), since the uninstaller software takes care of comprehensive removal of all resources associated with the installed software, including unregistering information in Registry and removing the appropriate packages.

The prodreg uninstall subcommand launches an external program. The command line conventions of these programs have to be used to indicate the alternate root for the product registry. Another possibility is to use the PKG_INSTALL_ROOT environment variable for this purpose as the install program is executed in the same environment as prodreg. Uninstall programs are frequently java classes which require Java to be installed. If Java software has been removed or is missing from a Solaris distribution, it is impossible to run java based uninstallers.

Only the prodreg unregister and uninstall subcommands can only be run with root permissions. This is because they modify the product registry in the case of unregister, and remove packages in the case of uninstall. The other operations merely read the registry and can be run with any user permissions. The prodreg uninstall subcommand might require root permissions as well, as installers can execute commands such as pkgadd(1M) or pkgrm(1M) which require root permissions to run.

Attributes associated with components are documented in various places -primarily in the *Application Packaging Developer's Guide*. The attributes associated with the Solaris Product Registry itself are described in the following glossary.

Dependent Components	List of components upon which the component depends.
Location	The location relative to which software was installed.
pkgs	List of packages which correspond to the component. These packages are added with pkgadd after the component is registered. They are removed with pkgrm before the component is unregistered.
Required Components	List of components on which the component depends.
Source	Media from which the install was done.

Supported Languages	List of locales for which there are registered titles.
Title	*Name* given by the `prodreg browse` subcommand. This name can be localized to the locale in which the shell is running.
Unique Name	Name used by previous versions of the Solaris Product Registry. This value is often set to the package name corresponding to a given component in the registry.
Vendor	Vendor who produced the component.
Version	Version string associated with the component.

The Registry can contain components which do not correspond to software actually installed on the system. This can be detected several ways. The easiest is to check using the `info` subcommand if a component is damaged. Another way is to determine where software was installed using the `info` subcommand, and verify it is still there.

Name projadd – administer a new project on the system

Synopsis

```
projadd [-n] [-f filename] [-p projid [-o]] [-c comment]
     [-U user [,user]... ] [-G group [,group]... ]
     [ [-K name [=value [,value]...]...]] project
```

Description `projadd` adds a new project entry to the `/etc/project` file. If the files backend is being used for the project database, the new project is available for use immediately upon the completion of the `projadd` command.

Options The following options are supported:

`-c` *comment*
: Add a project comment. Comments are stored in the project's entry in the `/etc/project` file. Generally, comments contain a short description of the project and are used as the field for the project's full name.

  Specify *comment* as a text string. *comment* cannot contain a colon (`:`) or NEWLINE.

`-f` *filename*
: Specify the project file to modify. If no *filename* is specified, the system project file, `/etc/project`, is modified.

`-G` *group*[,group...]
: Specify a group list for the project.

`-K` *name*[=*value*[,*value*...]
: Specify an attribute list for the project. Multiple `-K` options can be specified to set values on multiple keys, such as:

  `-K` *key1=value1* `-K "`*key2=*`(`*value2a*`),(`*value2b*`)"`

  Resource control attributes use parentheses to specify values for a key. Because many user shells interpret parentheses as special characters, it is best to enclose an argument to `-K` that contains parentheses with double quotes, as shown above and in EXAMPLES, below. See `resource_controls(5)` for a description of the resource controls you can specify for a project.

`-n`
: Syntax check. Check the format of the existing system project file and modifications only. The contents of the existing project file, such as user names, group names, and resources that are specified in the project attributes are not checked.

`-o`
: This option allows the project ID specified by the `-p` option to be non-unique within the project file.

`-p` *projid*
: Set the project ID of the new project.

  Specify *projid* as a non-negative decimal integer below `UID_MAX` as defined in `limits.h`. *projid* defaults to the next available

unique number above the highest number currently assigned. For example, if *projid*s `100`, `105`, and `200` are assigned, the next default *projid* is `201`. *projid*s between `0-99` are reserved by SunOS.

`-U` *user*[,user...] Specify a user list for the project.

Operands The following operands are supported:

project The name of the project to create. The *project* operand is a string consisting of characters from the set of alphabetic characters, numeric characters, underline (_), and hyphen (-). The period ('.') is reserved for projects with special meaning to the operating system. The first character of the project name must be a letter. An error message is displayed if these restrictions are not met.

Examples **EXAMPLE 1** Adding a Project

The following command creates the project `salesaudit` and sets the resource controls specified as arguments to the `-K` option.

```
projadd -p 111 -G sales,finance -c "Auditing Project" \
   -K "rcap.max-rss=10GB" \
   -K "process.max-file-size=(priv,50MB,deny)" \
   -K "task.max-lwps=(priv,100,deny)" salesaudit
```

This command would produce the following entry in `/etc/project`:

```
salesaudit:111:Auditing Project::sales,finance: \
process.max-file-size=(priv,52428800,deny); \
rcap.max-rss=10737418240;task.max-lwps=(priv,100,deny)
```

Note that the preceding would appear as one line in `/etc/project`.

Comparing the `projadd` command and resulting output in `/etc/project`, note the effect of the scaling factor in the resource cap (`rcap.max-rss=10GB`) and the resource control (`process.max-file-size=(priv,50MB,deny)`). Modifiers, such as B, KB, and MB, and scaling factors are specified in `resource_controls(5)`.

Exit Status The following exit values are returned:

`0` Successful completion.

`2` The command syntax was invalid. A usage message for `projadd` is displayed.

`3` An invalid argument was provided to an option.

`4` The *projid* given with the `-p` option is already in use.

`5` The project files contain an error. See `project(4)`.

`6` The project to be added, group, user, or resource does not exist.

9 The project is already in use.

10 Cannot update the `/etc/project` file.

Files `/etc/project` System project file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu
Interface Stability	See below.

Invocation is evolving. Human readable output is unstable.

See Also `projects(1)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `grpck(1M)`, `projdel(1M)`, `projmod(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `project(4)`, `attributes(5)`, `resource_controls(5)`

Notes In case of an error, `projadd` prints an error message and exits with a non-zero status.

`projadd` adds a project definition only on the local system. If a network name service such as NIS or LDAP is being used to supplement the local `/etc/project` file with additional entries, `projadd` cannot change information supplied by the network name service.

Name projdel – delete a project from the system

Synopsis

```
projdel [-f filename] project
```

Description The `projdel` utility deletes a project from the system and makes the appropriate changes to the system file.

Options The following options are supported:

`-f` *filename* Specify the project file to modify. If no *filename* is specified, the system project file, `/etc/project`, is modified.

Operands The following operands are supported:

project The name of the project to be deleted.

Exit Status The following exit values are returned:

`0` Successful completion.

`2` The command syntax was invalid. A usage message for `projdel` is displayed.

`3` An invalid argument was provided to an option.

`4` The *projid* given with the `-p` option is already in use.

`5` The project files contain an error. See `project(4)`.

`6` The project to be modified, group, user, or resource does not exist.

`9` The project is already in use.

`10` Cannot update the `/etc/project` file.

Files `/etc/project` System project file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu
Interface stabilty	See below.

Invocation is evolving. Human readable output is unstable.

See Also `projects(1)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `grpck(1M)`, `logins(1M)`, `projadd(1M)`, `projmod(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `project(4)`, `attributes(5)`

Diagnostics In case of an error, `projdel` prints an error message and exits with a non-zero status.

Notes `projdel` deletes a project definition only on the local system. If a network name service such as NIS or LDAP is being used to supplement the local `/etc/project` file with additional entries, `projdel` cannot change information supplied by the network name service.

Name projmod – modify a project's information on the system

Synopsis

```
projmod [-n] [-A|-f filename | -]

projmod [-n] [-A|-f filename | -] [-p projid [-o]]
     [-c comment] [-a|-s|-r] [-U user [,user]... ]
     [-G group [,group]... ]
     [ [-K name [=value [,value]...]...]]
     [-l new_projectname] project
```

Description The `projmod` utility modifies a project's definition on the system. `projmod` changes the definition of the specified project and makes the appropriate project-related system file and file system changes.

Options The following options are supported:

`-A`
Apply the project's resource controls, as defined in the system's project database, to the project if it is active.

`-a`
Specify that the users, groups, attributes, or attribute values specified by the `-U`, `-G` or `-K` options should be added to the project, rather than replacing the existing member or attribute list.

`-c` *comment*
Specify *comment* as a text string. Generally, *comment* contains a short description of the project. This information is stored in the project's `/etc/project` entry.

`-f` *filename*
Specify the project file to modify. If no *filename* is specified, the system project file, `/etc/project`, is modified.

`-G` *group* [,*group*...]
Specify a replacement list of member groups of the project. When used in conjunction with the `-a` or `-r` options, this option specifies a list of groups to be added or removed from the project.

`-K` *name*[=*value*[,*value*...]
Specify a replacement list of project attributes for the project. When used in conjunction with the `-a`, `-r`, or `-s` options, this option specifies a list of attribute values to be added, removed, or replaced in the project. Attributes must be delimited by semicolons (`;`). Multiple `-K` options can be specified to set, add, remove, or substitute values on multiple keys, such as:

```
-K key1=value1 -K "key2=(value2a),(value2b)"
```

Resource control attributes use parentheses to specify values for a key. Because many user shells interpret parentheses as special characters, it is best to enclose an argument to `-K` that

contains parentheses with double quotes, as shown above and in EXAMPLES, below. See `resource_controls(5)` for a description of the resource controls you can specify for a project.

`-l` *new_projectname*
Specify the new project name for the project. The *new_projectname* argument is a string consisting of characters from the set of alphabetic characters, numeric characters, period (`.`), underline (_), and hyphen (`-`). The first character should be alphabetic. An error message is written if these restrictions are not met. The project name must also be unique within the project file.

`-n`
Syntax check. Check the format of the existing system project file and modifications only. The contents of the existing project file, such as user names, group names, and resources that are specified in the project attributes are not checked.

`-o`
This option allows the project ID specified by the `-p` option to be non-unique within the project file.

`-p` *projid*
Specify a new project ID for the project. It must be a non-negative decimal integer less than MAXUID as defined in `param.h`. This value must be unique within the project file if the `-o` option is not specified.

`-r`
Specify that the users, groups, attributes, or attribute values specified by the `-U`, `-G` or `-K` options should be removed from the project, rather than replacing the existing member or attribute list.

`-s`
Specify that the list of attributes specified by the `-K` option should have their values replaced. If the attributes do not exist, they are added as if the a option was used. This option has no effect the `-U` or `-G` options.

`-U` *user* [,*user*...]
Specify a replacement list of member users of the project. When used in conjunction with the `-a` or `-r` options, this option specifies a list of users to be added or removed from the project.

Operands The following operands are supported:

project	An existing project name to be modified or displayed.
(none)	If no operand is given, the project file is validated without modifying any project.

Examples

EXAMPLE 1 Using the -K Option for Addition of an Attribute Value

Consider the following `project(4)` entry:

```
salesaudit:111:Auditing Project::sales,finance: \
  process.max-file-size=(priv,52428800,deny); \
  task.max-lwps=(priv,100,deny)
```

The preceding would appear as one line in `/etc/project`. For this and the following examples, the focus is on the attributes field in the `project` entry. That is, the last field, the field following the last semicolon.

The attributes field for the project `salesaudit` lists the following resource control:

```
task.max-lwps=(priv,1000,signal=KILL)
```

The following `projmod` command adds an action clause to the preceding entry:

```
# projmod -a -K "task.max-lwps=(priv,100,deny)" salesaudit
```

...with the resulting attributes field in the entry for `salesaudit`:

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

EXAMPLE 2 Using the -K Option for the Substitution of an Attribute Value

Assume an attributes field in a `project(4)` entry for the project `salesaudit` that lists the following resource control:

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

The following `projmod` command substitutes the action clause specified in the command for the action clauses in the preceding entry:

```
# projmod -s -K "task.max-lwps=(priv,500,signal=SIGSTOP)" salesaudit
```

...with the resulting attributes field in the entry for `salesaudit`:

```
task.max-lwps=(priv,500,signal=SIGSTOP)
```

EXAMPLE 3 Using the -K Option for Removal of an Attribute Value

Assume an attributes field in a `project(4)` entry for a project `salesaudit` that lists the following resource control:

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

The following `projmod` command removes the first action clause from the preceding entry:

```
# projmod -r -K "task.max-lwps=(priv,100,deny)" salesaudit
```

...with the resulting attributes field in the entry for `salesaudit`:

EXAMPLE 3 Using the -K Option for Removal of an Attribute Value *(Continued)*

```
task.max-lwps=(priv,1000,signal=KILL)
```

EXAMPLE 4 Specifying Multiple Attribute Values

Suppose you want to achieve the following resource controls for the project `salesaudit`:

```
task.max-lwps=(priv,100,deny)
process.max-file-size=(priv,50MB,deny)
```

The following `projmod` command adds these resource controls for `salesaudit`:

```
# projmod -a -K "task.max-lwps=(priv,100,deny)" \
-K "process.max-file-size=(priv,50MB,deny)" salesaudit
```

...with the resulting attributes field in the entry for `salesaudit`:

```
task.max-lwps=(priv,100,deny);process.max-file-size=(priv,52428800,deny)
```

In this example, note the effect of the use of the modifier and scaling factor for the resource control `process.max-file-size`. The specification in `projmod`:

```
"process.max-file-size=(priv,50MB,deny)"
```

...becomes, in `/etc/project`:

```
process.max-file-size=(priv,52428800,deny)
```

That is, `50MB` is expanded to `52428800`. The modifiers, such as MB, and scaling factors you can use for resource controls are specified in `resource_controls(5)`.

EXAMPLE 5 Binding a Pool to a Project

The following command sets the `project.pool` attribute for the project `sales`.

```
# projmod -a -K project.pool=salespool sales
```

Exit Status

In case of an error, `projmod` prints an error message and exits with one of the following values:

The following exit values are returned:

`0` Successful completion.

`2` The command syntax was invalid. A usage message for `projmod` is displayed.

`3` An invalid argument was provided to an option.

`4` The *projid* given with the `-p` option is already in use.

`5` The project files contain an error. See `project(4)`.

`6` The project to be modified, group, user, or resource does not exist.

9	The project is already in use.
10	Cannot update the `/etc/project` file.

Files

`/etc/group`	System file containing group definitions
`/etc/project`	System project file
`/etc/passwd`	System password file
`/etc/shadow`	System file containing users' encrypted passwords and related information

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu
Interface Stability	See below.

Invocation is committed. Human readable output is uncommitted.

See Also `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `projadd(1M)`, `projdel(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `passwd(4)`, `project(4)`, `attributes(5)`, `resource_controls(5)`

Notes The `projmod` utility modifies project definitions only in the local `/etc/project` file. If a network name service such as NIS or LDAP is being used to supplement the local files with additional entries, *projmod* cannot change information supplied by the network name service. However `projmod` verifies the uniqueness of project name and project ID against the external name service.

Name prstat – report active process statistics

Synopsis

```
prstat [-acJLmRtTv] [-C psrsetlist] [-j projlist]
     [-k tasklist] [-n ntop[,nbottom]] [-p pidlist]
     [-P cpulist] [-s key | -S key ] [-u euidlist]
     [-U uidlist] [-z zoneidlist] [-Z] [interval [count]]
```

Description The `prstat` utility iteratively examines all active processes on the system and reports statistics based on the selected output mode and sort order. `prstat` provides options to examine only processes matching specified PIDs, UIDs, zone IDs, CPU IDs, and processor set IDs.

The `-j`, `-k`, `-C`, `-p`, `-P`, `-u`, `-U`, and `-z` options accept lists as arguments. Items in a list can be either separated by commas or enclosed in quotes and separated by commas or spaces.

If you do not specify an option, `prstat` examines all processes and reports statistics sorted by CPU usage.

Options The following options are supported:

`-a`
Report information about processes and users. In this mode `prstat` displays separate reports about processes and users at the same time.

`-c`
Print new reports below previous reports instead of overprinting them.

`-C` *psrsetlist*
Report only processes or lwps that are bound to processor sets in the given list. Each processor set is identified by an integer as reported by `psrset(1M)`. The load averages displayed are the sum of the load averages of the specified processor sets (see `pset_getloadavg(3C)`). Processes with one or more LWPs bound to processor sets in the given list are reported even when the `-L` option is not used.

`-j` *projlist*
Report only processes or lwps whose project ID is in the given list. Each project ID can be specified as either a project name or a numerical project ID. See `project(4)`.

`-J`
Report information about processes and projects. In this mode `prstat` displays separate reports about processes and projects at the same time.

`-k` *tasklist*
Report only processes or lwps whose task ID is in *tasklist*.

`-L`
Report statistics for each light-weight process (LWP). By default, `prstat` reports only the number of LWPs for each process.

`-m`
Report microstate process accounting information. In addition to all fields listed in `-v` mode, this mode also includes the percentage of time the process has spent processing system traps, text page faults, data page faults, waiting for user locks and waiting for CPU (latency time).

`-n` *ntop*[,*nbottom*]
Restrict number of output lines. The *ntop* argument determines how many lines of process or `lwp` statistics are reported, and the *nbottom* argument determines how many lines of user, task, or projects statistics are reported if the `-a`, `-t`, `-T`, or `-J` options are specified. By default, `prstat` displays as many lines of output that fit in a window or terminal. When you specify the `-c` option or direct the output to a file, the default values for `ntop` and `nbottom` are `15` and `5`.

`-p` *pidlist*
Report only processes whose process ID is in the given list.

`-P` *cpulist*
Report only processes or lwps which have most recently executed on a CPU in the given list. Each CPU is identified by an integer as reported by `psrinfo(1M)`.

`-R`
Put `prstat` in the real time scheduling class. When this option is used, `prstat` is given priority over time-sharing and interactive processes. This option is available only for superuser.

`-s` *key*
Sort output lines (that is, processes, `lwps`, or users) by *key* in descending order. Only one *key* can be used as an argument.

There are five possible key values:

cpu
Sort by process CPU usage. This is the default.

pri
Sort by process priority.

rss
Sort by resident set size.

size
Sort by size of process image.

time
Sort by process execution time.

`-S` *key*
Sort output lines by *key* in ascending order. Possible *key* values are the same as for the `-s` option. See `-s`.

`-t`
Report total usage summary for each user. The summary includes the total number of processes or LWPs owned by the user, total size of process images, total resident set size, total cpu time, and percentages of recent cpu time and system memory.

`-T`
Report information about processes and tasks. In this mode `prstat` displays separate reports about processes and tasks at the same time.

`-u` *euidlist*
Report only processes whose effective user ID is in the given list. Each user ID may be specified as either a login name or a numerical user ID.

`-U` *uidlis*t
Report only processes whose real user ID is in the given list. Each user ID may be specified as either a login name or a numerical user ID.

`-v`
Report verbose process usage. This output format includes the percentage of time the process has spent in user mode, in system mode, and sleeping. It also includes the number of voluntary and involuntary context switches, system calls and the number of signals received. Statistics that are not reported are marked with the `-` sign.

`-z` *zoneidlist*
Report only processes or LWPs whose zone ID is in the given list. Each zone ID can be specified as either a zone name or a numerical zone ID. See `zones(5)`.

`-Z`
Report information about processes and zones. In this mode, `prstat` displays separate reports about processes and zones at the same time.

Output The following list defines the column headings and the meanings of a `prstat` report:

PID
The process ID of the process.

USERNAME
The real user (login) name or real user ID.

SWAP
The total virtual memory size of the process, including all mapped files and devices, in kilobytes (`K`), megabytes (`M`), or gigabytes (`G`).

RSS
The resident set size of the process (RSS), in kilobytes (K), megabytes (M), or gigabytes (G). The RSS value is an estimate provided by `proc(4)` that might underestimate the actual resident set size. Users who want to get more accurate usage information for capacity planning should use the `-x` option to `pmap(1)` instead.

STATE
The state of the process:

cpu*N*
Process is running on CPU *N*.

sleep
Sleeping: process is waiting for an event to complete.

wait
Waiting: process is waiting for CPU usage to drop to the CPU-caps enforced limits. See the description of `CPU-caps` in `resource_controls(5)`.

run
Runnable: process in on run queue.

zombie
Zombie state: process terminated and parent not waiting.

stop
Process is stopped.

PRI
The priority of the process. Larger numbers mean higher priority.

NICE
Nice value used in priority computation. Only processes in certain scheduling classes have a nice value.

TIME
The cumulative execution time for the process.

CPU
The percentage of recent CPU time used by the process. If executing in a non-global `zone` and the pools facility is active, the percentage will be that of the processors in the processor set in use by the pool to which the `zone` is bound.

PROCESS
The name of the process (name of executed file).

LWPID
The `lwp` ID of the `lwp` being reported.

NLWP
The number of `lwps` in the process.

With the some options, in addition to a number of the column headings shown above, there are:

NPROC
Number of processes in a specified collection.

MEMORY
Percentage of memory used by a specified collection of processes.

The following columns are displayed when the `-v` or `-m` option is specified

USR
The percentage of time the process has spent in user mode.

SYS
The percentage of time the process has spent in system mode.

TRP
The percentage of time the process has spent in processing system traps.

TFL
The percentage of time the process has spent processing text page faults.

DFL
The percentage of time the process has spent processing data page faults.

LCK
The percentage of time the process has spent waiting for user locks.

SLP
The percentage of time the process has spent sleeping.

LAT
The percentage of time the process has spent waiting for CPU.

VCX
The number of voluntary context switches.

ICX
The number of involuntary context switches.

SCL
The number of system calls.

SIG
The number of signals received.

Under the `-L` option, one line is printed for each `lwp` in the process and some reporting fields show the values for the `lwp`, not the process.

Operands The following operands are supported:

count
Specifies the number of times that the statistics are repeated. By default, `prstat` reports statistics until a termination signal is received.

interval
Specifies the sampling interval in seconds; the default interval is `5` seconds.

Examples

EXAMPLE 1 Reporting the Five Most Active Super-User Processes

The following command reports the five most active super-user processes running on CPU1 and CPU2:

```
example% prstat -u root -n 5 -P 1,2 1 1

PID   USERNAME  SWAP   RSS STATE  PRI  NICE      TIME  CPU PROCESS/LWP
 306   root     3024K 1448K sleep   58    0   0:00.00 0.3% sendmail/1
 102   root     1600K  592K sleep   59    0   0:00.00 0.1% in.rdisc/1
 250   root     1000K  552K sleep   58    0   0:00.00 0.0% utmpd/1
 288   root     1720K 1032K sleep   58    0   0:00.00 0.0% sac/1
   1   root      744K  168K sleep   58    0   0:00.00 0.0% init/1
TOTAL:       25, load averages:  0.05, 0.08, 0.12
```

EXAMPLE 2 Displaying Verbose Process Usage Information

The following command displays verbose process usage information about processes with lowest resident set sizes owned by users root and john.

```
example% prstat -S rss -n 5 -vc -u root,john

 PID USERNAME USR SYS TRP TFL DFL LCK SLP LAT VCX ICX SCL SIG PROCESS/LWP
   1 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 init/1
 102 root     0.0 0.0  -   -   -   -  100  -    0   0   3   0 in.rdisc/1
 250 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 utmpd/1
1185 john    0.0 0.0  -   -   -   -  100  -    0   0   0   0 csh/1
 240 root     0.0 0.0  -   -   -   -  100  -    0   0   0   0 powerd/4
 TOTAL:       71, load averages:  0.02, 0.04, 0.08
```

Exit Status

The following exit values are returned:

0
Successful completion.

1
An error occurred.

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

proc(1), psrinfo(1M), psrset(1M), sar(1M), pset_getloadavg(3C), proc(4), project(4), attributes(5), resource_controls(5), zones(5)

Notes The snapshot of system usage displayed by `prstat` is true only for a split-second, and it may not be accurate by the time it is displayed. When the `-m` option is specified, `prstat` tries to turn on microstate accounting for each process; the original state is restored when `prstat` exits. See `proc(4)` for additional information about the microstate accounting facility.

The total memory size reported in the SWAP and RSS columns for groups of processes can sometimes overestimate the actual amount of memory used by processes with shared memory segments.

Name prtconf – print system configuration

Synopsis

```
/usr/sbin/prtconf   [-V] |  [-F] |  [-x] |  [-bpv]
 |  [-acDPv] [dev_path]
```

Description The `prtconf` command prints the system configuration information. The output includes the total amount of memory, and the configuration of system peripherals formatted as a device tree.

If a device path is specified on the command line for those command options that can take a device path, `prtconf` will only display information for that device node.

Options The following options are supported:

`-a` Display all the ancestors device nodes, up to the root node of the device tree, for the device specified on the command line.

`-b` Display the firmware device tree root properties for the purpose of platform identification. These properties are “name”, “compatible”, “banner-name” and “model”.

`-c` Display the device subtree rooted at the device node specified on the command line, that is, display all the children of the device node specified on the command line.

`-D` For each system peripheral in the device tree, displays the name of the device driver used to manage the peripheral.

`-F` A SPARC-only option. Returns the device path name of the console frame buffer, if one exists. If there is no frame buffer, `prtconf` returns a non-zero exit code. This flag must be used by itself. It returns only the name of the console, frame buffer device or a non-zero exit code. For example, if the console frame buffer on a SUNW,Ultra-30 is `ffb`, the command returns: `/SUNW,ffb@1e,0:ffb0`. This option could be used to create a symlink for `/dev/fb` to the actual console device.

`-p` Displays information derived from the device tree provided by the firmware (PROM) on SPARC platforms or the booting system on x86 platforms.The device tree information displayed using this option is a snapshot of the initial configuration and may not accurately reflect reconfiguration events that occur later.

`-P` Includes information about pseudo devices. By default, information regarding pseudo devices is omitted.

`-v` Specifies verbose mode.

`-V` Displays platform-dependent PROM (on SPARC platforms) or booting system (on x86 platforms) version information. This flag must be used by itself. The output is a string. The format of the string is arbitrary and platform-dependent.

`-x` Reports if the firmware on this system is 64-bit ready. Some existing platforms may need a firmware upgrade in order to run the 64-bit kernel. If the operation is not applicable to this platform or the firmware is already 64-bit ready, it exits silently with a return code of zero. If the operation is applicable to this platform and the firmware is not 64-bit ready, it displays a descriptive message on the standard output and exits with a non-zero return code. The hardware platform documentation contains more information about the platforms that may need a firmware upgrade in order to run the 64-bit kernel.

This flag overrides all other flags and must be used by itself.

Operands The following operands are supported:

dev_path The path to a target device minor node, device nexus node, or device link for which device node configuration information is displayed

Exit Status The following exit values are returned:

`0` No error occurred.

`non-zero` With the `-F` option (SPARC only), a non-zero return value means that the output device is not a frame buffer. With the `-x` option, a non-zero return value means that the firmware is not 64–bit ready. In all other cases, a non-zero return value means that an error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu
Interface Stability	Unstable

See Also `fuser(1M)`, `modinfo(1M)`, `sysdef(1M)`, `attributes(5)`

Sun Hardware Platform Guide

SPARC Only `openprom(7D)`

Notes The output of the `prtconf` command is highly dependent on the version of the PROM installed in the system. The output will be affected in potentially all circumstances.

The `driver not attached` message means that no driver is currently attached to that instance of the device. In general, drivers are loaded and installed (and attached to hardware instances) on demand, and when needed, and may be uninstalled and unloaded when the device is not in use.

On x86 platforms, the use of `prtconf -vp` provides a subset of information from `prtconf -v`. The value of integer properties from `prtconf -vp` might require byte swapping for correct interpretation.

Name prtdiag – display system diagnostic information

Synopsis

```
/usr/sbin/prtdiag [-v] [-l]
```

Description `prtdiag` displays system configuration and diagnostic information on `sun4u` and `sun4v` systems.

The diagnostic information lists any failed field replaceable units (FRUs) in the system.

The interface, output, and location in the directory hierarchy for `prtdiag` are uncommitted and subject to change in future releases.

`prtdiag` does not display diagnostic information and environmental status when executed on the Sun Enterprise 10000 server. See the `/var/opt/SUNWssp/adm/${SUNW_HOSTNAME}/messages` file on the system service processor (SSP) to obtain such information for this server.

Options The following options are supported:

`-l` Log output. If failures or errors exist in the system, output this information to `syslogd(1M)` only.

`-v` Verbose mode. Displays the time of the most recent AC Power failure, and the most recent hardware fatal error information, and (if applicable) environmental status. The hardware fatal error information is useful to repair and manufacturing for detailed diagnostics of FRUs.

Exit Status The following exit values are returned:

`0` No failures or errors are detected in the system.

`1` Failures or errors are detected in the system.

`2` An internal `prtdiag` error occurred, for example, out of memory.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkvm
Interface Stability	Unstable*

*The output is unstable.

See Also `modinfo(1M)`, `prtconf(1M)`, `psrinfo(1M)`, `sysdef(1M)`, `syslogd(1M)`, `attributes(5)`, `openprom(7D)`

Notes Not all diagnostic and system information is available on every Solaris platform, and therefore cannot be displayed by `prtdiag`. On those platforms, further information can be obtained from the System Controller.

Name prtdscp – display DSCP IP addresses

Synopsis

```
prtdscp [-v ]

prtdscp [-v ] -h

prtdscp [-v ] -d

prtdscp [-v ] -s
```

Description `prtdscp` displays the IP addresses associated with a Domain to Service Processor Communications Protocol (DSCP) link. If no arguments are specified, `prtdscp` displays the IP addresses on both ends of the DSCP link. The IP address of either the Service Processor or domain side can be displayed separately by the use of the `-s` or `-d` options, respectively.

Options The following options are supported:

`-v` Verbose mode. Print additional details about the program's internal progress to `stderr`.

`-h` Help. Print a brief synopsis of the program's usage and exit. All other command line arguments are ignored.

`-d` Display only the local domain's IP address.

`-s` Display only the remote Service Processor's IP address.

Examples **EXAMPLE 1** Displaying both addresses

The following example displays both the local domain's IP address and the remote SP's IP address:

```
# prtdscp
Domain Address: 192.168.103.2
SP Address: 192.168.103.1
```

EXAMPLE 2 Displaying the local IP address

The following example displays the local domain's IP address:

```
# prtdscp -d
192.168.103.2
```

EXAMPLE 3 Displaying the remote IP address

The following example display the remote SP's IP address:

```
# prtdscp -s
192.168.103.1
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdscpr.u, SUNWdscpu.u
Interface Stability	Evolving

See Also attributes(5)

Name prtfru – print FRUID-specific information about the FRUs on a system or domain

Synopsis

```
/usr/sbin/prtfru [-d] | [-clx] [container]
```

Description The `prtfru` utility is used to obtain FRUID data from the system or domain. Its output is that of a tree structure echoing the path in the FRU (Field-Replaceable Unit) tree to each container. When a container is found, the data from that container is printed in a tree-like structure as well.

`prtfru` without any arguments will print the FRU hierarchy and all of the FRUID container data. `prtfru` prints to stdout which may be redirected to a file.

Options The following options are supported:

`-c` Prints *only* the containers and their data. This option does not print the FRU tree hierarchy.

`-d` Prints a DTD for the current registry to stdout.

`-l` Prints *only* the FRU tree hierarchy. This option does not print the container data.

`-x` Prints in XML format with a system identifier (SYSTEM) of `prtfrureg.dtd`.

Options `-c` and `-l` can be used together to obtain a list of the containers.

Operands The folowing operand is supported:

container The name of a particular container in the FRU hierarchy, that is, either the name or path/name of a container as displayed in the `-l` option.

Exit Status The following exit values are returned:

`0` All information was found and printed successfully.

`>0` An error has occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWfruid

See Also `fruadm(1M)`, `attributes(5)`

Name prtpicl – print PICL tree

Synopsis

```
/usr/sbin/prtpicl [-c picl_class] [-v]
```

Description The `prtpicl` command prints the PICL tree maintained by the PICL daemon. The output of `prtpicl` includes the name and PICL class of the nodes.

Options The following options are supported:

`-c` *picl_class* Print only the nodes of the named PICL class.

`-v` Print in verbose mode. In verbose mode, `prtpicl` prints a list of properties and values for each node. Verbose mode is disabled by default.

Exit Status The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpiclu

See Also `picld(1M)`, `attributes(5)`

Name prtvtoc – report information about a disk geometry and partitioning

Synopsis `prtvtoc [-fhs] [-t` *vfstab*`] [-m` *mnttab*`]` *device*

Description The `prtvtoc` command allows the contents of the label to be viewed. The command can be used only by the super-user.

The *device* name can be the file name of a raw device in the form of `/dev/rdsk/c?t?d?s2` or can be the file name of a block device in the form of `/dev/dsk/c?t?d?s2`.

Options The following options are supported:

`-f` Report on the disk free space, including the starting block address of the free space, number of blocks, and unused partitions.

`-h` Omit the headers from the normal output.

`-m` *mnttab* Use *mnttab* as the list of mounted filesystems, in place of `/etc/mnttab`.

`-s` Omit all headers but the column header from the normal output.

`-t` *vfstab* Use *vfstab* as the list of filesystem defaults, in place of `/etc/vfstab`.

Examples **EXAMPLE 1** Using the `prtvtoc` Command

The following example uses the `prtvtoc` command on a 424-megabyte hard disk:

```
example# prtvtoc /dev/rdsk/c0t3d0s2
* /dev/rdsk/c0t3d0s2 partition map
*
* Dimension:
*     512 bytes/sector
*      80 sectors/track
*       9 tracks/cylinder
*     720 sectors/cylinder
*    2500 cylinders
*    1151 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
* *                          First    Sector   Last
* Partition   Tag   Flags   Sector   Count    Sector   Mount Directory
     0         2     00          0    76320    76319   /
     1         3     01      76320   132480   208799
     2         5     00          0   828720   828719
     5         6     00     208800   131760   340559   /opt
     6         4     00     340560   447120   787679   /usr
     7         8     00     787680    41040   828719   /export/home
example#
```

EXAMPLE 1 Using the `prtvtoc` Command *(Continued)*

The data in the `Tag` column above indicates the type of partition, as follows:

Name	*Number*
UNASSIGNED	0x00
BOOT	0x01
ROOT	0x02
SWAP	0x03
USR	0x04
BACKUP	0x05
STAND	0x06
VAR	0x07
HOME	0x08
ALTSCTR	0x09
CACHE	0x0a
RESERVED	0x0b

The data in the `Flags` column above indicates how the partition is to be mounted, as follows:

Name	*Number*
MOUNTABLE, READ AND WRITE	0x00
NOT MOUNTABLE	0x01
MOUNTABLE, READ ONLY	0x10

EXAMPLE 2 Using the `prtvtoc` Command with the `-f` Option

The following example uses the `prtvtoc` command with the `-f` option on a 424-megabyte hard disk:

```
example# prtvtoc -f /dev/rdsk/c0t3d0s2
FREE_START=0 FREE_SIZE=0 FREE_COUNT=0 FREE_PART=34
```

EXAMPLE 3 Using the `prtvtoc` Command on a Disk Over One Terabyte

The following example uses uses the `prtvtoc` command on a disk over one terabyte:.

EXAMPLE 3 Using the `prtvtoc` Command on a Disk Over One Terabyte *(Continued)*

```
example# prtvtoc /dev/rdsk/c1t1d0s2
* /dev/rdsk/c1t1d0s2 partition map
*
* Dimensions:
*     512 bytes/sector
* 3187630080 sectors
* 3187630013 accessible sectors
*
* Flags:
*   1: unmountable
*  10: read-only
*
*                          First     Sector    Last
* Partition  Tag  Flags    Sector     Count    Sector  Mount Directory
0      2    00         34    262144    262177
1      3    01     262178    262144    524321
6      4    00     524322 3187089340 3187613661
8     11    00  3187613662     16384 318763004
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `devinfo(1M)`, `fmthard(1M)`, `format(1M)`, `mount(1M)`, `attributes(5)`

Warnings The `mount` command does not check the "not mountable" bit.

Name psradm – change processor operational status

Synopsis

```
psradm -f | -i | -n | -s [-v] [-F] processor_id

psradm -a -f | -i | -n | -s [-v] [-F]
```

Description The `psradm` utility changes the operational status of processors. The legal states for the processor are `on-line`, `off-line`, `spare`, `faulted`, and `no-intr`.

An `on-line` processor processes LWPs (lightweight processes) and can be interrupted by I/O devices in the system.

An `off-line` processor does not process any LWPs. Usually, an `off-line` processor is not interruptible by I/O devices in the system. On some processors or under certain conditions, it might not be possible to disable interrupts for an `off-line` processor. Thus, the actual effect of being `off-line` might vary from machine to machine.

A spare processor does not process any LWPs. A spare processor can be brought `on-line`, `off-line` or to `no-intr` by a privileged user of the system or by the kernel in response to changes in the system state.

A faulted processor is identified by the kernel, which monitors the behavior of processors over time. A privileged user can set the state of a faulted processor to be `on-line`, `off-line`, `spare` or `no-intr`, but must use the force option to do so.

A `no-intr` processor processes LWPs but is not interruptible by I/O devices.

A processor can not be taken `off-line` or made `spare` if there are LWPs that are bound to the processor unless the additional `-F` option is used. The `-F` option removes processor bindings of such LWPs before changing the processor's operational status. On some architectures, it might not be possible to take certain processors `off-line` or `spare` if, for example, the system depends on some resource provided by the processor.

At least one processor in the system must be able to process LWPs. At least one processor must also be able to be interrupted. Since an `off-line` or `spare` processor can be interruptible, it is possible to have an operational system with one processor `no-intr` and all other processors `off-line` or `spare` but with one or more accepting interrupts.

If any of the specified processors are powered off, `psradm` might power on one or more processors.

Only users with the `PRIV_SYS_RES_CONFIG` privilege can use the `psradm` utility.

Options The following options are supported:

`-a` Perform the action on all processors, or as many as possible.

`-f` Take the specified processors `off-line`.

`-F` Force the transition to the additional specified state. Required if one or more of the specified processors was in the faulted state. Set the specified processors to faulted, if no other transition option was specified. Forced transitions can only be made to `faulted`, `spare`, or `off-line` states. Administrators are encouraged to use the `-Q` option for `pbind(1M)` to find out which threads will be affected by forced a processor state transition.

`-i` Set the specified processors `no-intr`.

`-n` Bring the specified processors `on-line`.

`-s` Make the specified processors spare.

`-v` Output a message giving the results of each attempted operation.

Operands The following operands are supported:

processor_id The processor ID of the processor to be set `on-line` or `off-line`, `spare`, or `no-intr`.

Specify *processor_id* as an individual processor number (for example, `3`), multiple processor numbers separated by spaces (for example, `1 2 3`), or a range of processor numbers (for example, `1-4`). It is also possible to combine ranges and (individual or multiple) *processor_id*s (for example, `1-3 5 7-8 9`).

Examples **EXAMPLE 1** Setting Processors to `off-line`

The following example sets processors 2 and 3 `off-line`:

```
% psradm -f 2 3
```

EXAMPLE 2 Setting Processors to no-intr

The following example sets processors 1 and 2 `no-intr`:

```
% psradm -i 1 2
```

EXAMPLE 3 Setting Processors to `spare`

The following example sets processors 1 and 2 spare, even if either of the processors was in the faulted state:

```
% psradm -F -s 1 2
```

EXAMPLE 4 Setting All Processors `on-line`

```
% psradm -a -n
```

EXAMPLE 5 Forcing Processors to `off-line`

The following example sets processors 1 and 2 offline, and revokes the processor bindings from the processes bound to them:

```
% psradm -F -f 1 2
```

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/etc/wtmpx` Records logging processor status changes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `pbind(1M)`, `psrinfo(1M)`, `psrset(1M)`, `p_online(2)`, `processor_bind(2)`, `attributes(5)`

Diagnostics

`psradm: processor 4: Invalid argument`
The specified processor does not exist in the configuration.

`psradm: processor 3: Device busy`
The specified processor could not be taken `off-line` because it either has LWPs bound to it, is the last `on-line` processor in the system, or is needed by the system because it provides some essential service.

`psradm: processor 3: Device busy`
The specified processor could not be set `no-intr` because it is the last interruptible processor in the system, or or it is the only processor in the system that can service interrupts needed by the system.

`psradm: processor 3: Device busy`
The specified processor is powered off, and it cannot be powered on because some platform-specific resource is unavailable.

`psradm: processor 0: Not owner`
The user does not have permission to change processor status.

`psradm: processor 2: Operation not supported`
The specified processor is powered off, and the platform does not support power on of individual processors.

Name psrinfo – displays information about processors

Synopsis

```
psrinfo [-p] [-v] [processor_id]...

psrinfo [-p] -s processor_id
```

Description `psrinfo` displays information about processors. Each physical processor may support multiple virtual processors. Each virtual processor is an entity with its own interrupt ID, capable of executing independent threads.

Without the *processor_id* operand, `psrinfo` displays one line for each configured processor, displaying whether it is on-line, non-interruptible (designated by no-intr), spare, off-line, faulted or powered off, and when that status last changed. Use the processor_id operand to display information about a specific processor. See `OPERANDS`.

Options The following options are supported:

`-s` *processor_id* Silent mode. Displays `1` if the specified processor is fully on-line. Displays `0` if the specified processor is non-interruptible, spare, off-line, faulted or powered off.

Use silent mode when using `psrinfo` in shell scripts.

`-p` Display the number of physical processors in a system.

When combined with the `-v` option, reports additional information about each physical processor.

`-v` Verbose mode. Displays additional information about the specified processors, including: processor type, floating point unit type and clock speed. If any of this information cannot be determined, `psrinfo` displays `unknown`.

When combined with the `-p` option, reports additional information about each physical processor.

Operands The following operands are supported:

processor_id The processor ID of the processor about which information is to be displayed.

Specify *processor_id* as an individual processor number (for example, `3`), multiple processor numbers separated by spaces (for example, `1 2 3`), or a range of processor numbers (for example, `1-4`). It is also possible to combine ranges and (individual or multiple) *processor_id*s (for example, `1-3 5 7-8 9`).

Examples

EXAMPLE 1 Displaying Information About All Configured Processors in Verbose Mode

The following example displays information about all configured processors in verbose mode.

```
psrinfo -v
```

EXAMPLE 2 Determining If a Processor is On-line

The following example uses `psrinfo` in a shell script to determine if a processor is on-line.

```
if [ "`psrinfo -s 3 2> /dev/null`" –eq 1 ]
then
    echo "processor 3 is up"
fi
```

EXAMPLE 3 Displaying Information About the Physical Processors in the System

With no additional arguments, the `-p` option displays a single integer: the number of physical processors in the system:

```
> psrinfo -p
               8
```

`psrinfo` also accepts command line arguments (processor IDs):

```
> psrinfo -p 0 512   # IDs 0 and 512 exist on the
1                    # same physical processor

> psrinfo -p 0 1     # IDs 0 and 1 exist on different
2                    # physical processors
```

In this example, virtual processors `0` and `512` exist on the same physical processor. Virtual processors `0` and `1` do not. This is specific to this example and is and not a general rule.

Exit Status

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`psradm(1M)`, `p_online(2)`, `processor_info(2)`, `attributes(5)`

Diagnostics

`psrinfo: processor 9: Invalid argument` The specified processor does not exist.

Name psrset – creation and management of processor sets

Synopsis

```
psrset -a [-F] processor_set_id processor_id...

psrset -b processor_set_id pid [/lwpid]...

psrset -c [-F] [processor_id]...

psrset -d processor_set_id...

psrset -e processor_set_id command [argument(s)]

psrset -f processor_set_id

psrset [-i] [processor_set_id]...

psrset -n processor_set_id

psrset -p [processor_id]...

psrset [-q] [pid [/lwpid]]...

psrset -Q [processor_set_id]...

psrset -r [-F] processor_id...

psrset -u pid [/lwpid]...

psrset -U [processor_set_id]...
```

Description The `psrset` utility controls the management of processor sets. Processor sets allow the binding of processes or LWPs to groups of processors, rather than just a single processor. Processors assigned to processor sets can run only LWPs that have been bound to that processor set.

This command cannot be used to modify processor disposition when pools are enabled. Use `pooladm(1M)` and `poolcfg(1M)` to modify processor set configuration through the resource pools facility.

Options The following options are supported:

`-a` Assign the specified processors to the specified processor set. With the additional `-F` option, all LWPs bound to the specified processors will be unbound prior to changing processor sets.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-b` Bind all or a subset of the LWPs of the specified processes to the specified processor set.

LWPs bound to a processor set are restricted to run only on the processors in that set. Processes can only be bound to non-empty processor sets, that is, processor sets that have had processors assigned to them.

Bindings are inherited, so new LWPs and processes created by a bound LWP have the same binding. Binding an interactive shell to a processor, for example, binds all commands executed by the shell.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-c` Create a new processor set and displays the new processor set ID. With the additional `-F` option, all LWPs bound to the specified processors will be unbound prior to assigning them to the processor set being created.

If a list of processors is given, it also attempts to assign those processors to the processor set. If this succeeds, the processors are idle until LWPs are bound to the processor set. This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

Only a limited number of processor sets can be active (created and not destroyed) at a given time. This limit is always be greater than the number of processors in the system. If the `-c` option is used when the maximum number of processor sets is already active, the command fails.

The following format is used for the first line of output of the `-c` option when the `LC_MESSAGES` locale category specifies the "C" locale. In other locales, the strings `created`, `processor`, and `set` can be replaced with more appropriate strings corresponding to the locale.

`"created processor set %d\n"` *processor set ID*

`-d` Remove the specified processor set, releasing all processors and processes associated with it.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-e` Execute a command (with optional arguments) in the specified processor set.

The command process and any child processes are executed only by processors in the processor set.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-f` Disables interrupts for all processors within the specified processor set. See `psradm`(1M).

If some processors in the set cannot have their interrupts disabled, the other processors still have their interrupts disabled, and the command reports an error and return non-zero exit status.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-F` Forces the specified processor set operation by unbinding all threads bound to the specified processor. Only the `-a` or the `-r` option can be used in combination with this option. Administrators are encouraged to use the `-Q` option for `pbind(1M)` to find out which threads will be affected by such operation.

`-i` Display a list of processors assigned to each named processor set. If no argument is given, a list of all processor sets and the processors assigned to them is displayed. This is also the default operation if the `psrset` command is not given an option.

`-n` Enable interrupts for all processors within the specified processor set. See `psradm(1M)`.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-p` Display the processor set assignments for the specified list of processors. If no argument is given, the processor set assignments for all processors in the system is given.

`-q` Display the processor set bindings of the specified processes or of all processes. If a process is composed of multiple LWPs which have different bindings and the LWPs are not explicitly specified, the bindings of only one of the bound LWPs is displayed. The bindings of a subset of LWPs can be displayed by appending "/lwpids" to the process IDs. Multiple LWPs may be selected using "-" and "," delimiters. See EXAMPLES.

`-Q` Display the LWPs bound to the specified list of processor sets, or all LWPs with processor set bindings.

`-r` Remove a list of processors from their current processor sets. Processors that are removed return to the general pool of processors.

Processors with LWPs bound to them using `pbind(1M)` can be assigned to or removed from processor sets using the `-F` option.

This option is restricted to users with the `PRIV_SYS_RES_CONFIG` privilege.

`-u` Remove the processor set bindings of a subset or all the LWPs of the specified processes, allowing them to be executed on any on-line processor if they are not bound to individual processors through `pbind`.

Users with the `PRIV_SYS_RES_CONFIG` privilege can unbind any process or LWP from any active processor set. Other users can unbind processes and LWPs from processor sets that do not have the `PSET_NOESCAPE` attribute set. In addition, the user must have permission to control the affected processes; the real or effective user ID of the user must match the real or saved user ID of the target processes.

`-U` Removes the bindings of all LWPs bound to the specified list of processor sets, or to any processor set if no argument is specified.

Operands The following operands are supported:

pid Specify *pid* as a process ID.

lwpid The set of LWPIDs of the specified process to be controlled or queried. The syntax for selecting LWP IDs is as follows:

```
2,3,4-8       LWP IDs 2, 3, and 4 through 8
-4            LWPs whose IDs are 4 or below
4-            LWPs whose IDs are 4 or above
```

processor_id Specify *processor_id* as an individual processor number (for example, `3`), multiple processor numbers separated by spaces (for example, `1 2 3`), or a range of processor numbers (for example, `1-4`). It is also possible to combine ranges and (individual or multiple) *processor_id*s (for example, `1-3 5 7-8 9`).

processor_set_id Specify *processor_set_id* as a processor set ID.

Exit Status The following exit values are returned:

`0` Successful completion.

non-0 An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Stability Level	Stable

See Also `pbind(1M)`, `pooladm(1M)`, `poolcfg(1M)`, `psradm(1M)`, `psrinfo(1M)`, `processor_bind(2)`, `processor_info(2)`, `pset_bind(2)`, `pset_create(2)`, `pset_info(2)`, `sysconf(3C)`, `libpool(3LIB)`, `attributes(5)`, `privileges(5)`

Diagnostics The following output indicates that the specified process did not exist or has exited:

```
psrset: cannot query pid 31: No such process
```

The following output indicates that the user does not have permission to bind the process:

```
psrset: cannot bind pid 31: Not owner
```

The following output indicates that the user does not have permission to assign the processor:

```
psrset: cannot assign processor 4: Not owner
```

The following output indicates that the specified processor is not on-line, or the specified processor does not exist.

```
psrset: cannot assign processor 8: Invalid argument
```

The following output indicates that an LWP in the specified process is bound to a processor and cannot be bound to a processor set that does not include that processor:

```
psrset: cannot bind pid 67: Device busy
```

The following output indicates that the specified processor could not be added to the processor set. This can be due to bound LWPs on that processor, or because that processor cannot be combined in the same processor set with other processors in that set, or because the processor is the last one in its current processor set:

```
psrset: cannot assign processor 7: Device busy
```

The following output indicates that the specified processor set does not exist:

```
psrset: cannot execute in processor set 8: Invalid argument
```

The following output indicates that the maximum number of processor sets allowed in the system is already active:

```
psrset: cannot create processor set: Not enough space
```

The following output indicates that the pools facility is active.

```
psrset: cannot assign processor 7: Operation not supported
psrset: cannot bind pid 31: Operation not supported
psrset: cannot bind pid 31: Operation not supported
psrset: could not create processor set: Operation not supported
psrset: could not remove processor set 1: Operation not supported
psrset: cannot exec in processor set 1: Operation not supported
psrset: cannot remove processor 7: Operation not supported
psrset: cannot unbind pid 31: Operation not supported
```

Name putdev – edits device table

Synopsis

```
putdev -a alias [attribute=value [...]]

putdev -m device attribute=value [attribute = value [...]]

putdev -d device [attribute [...]]
```

Description `putdev` adds a new device to the device table, modifies an existing device description or removes a device entry from the table. The first synopsis is used to add a device. The second synopsis is used to modify existing entries by adding or changing attributes. If a specified attribute is not defined, this option adds that attribute to the device definition. If a specified attribute is already defined, it modifies the attribute definition. The third synopsis is used to delete either an entire device entry or, if the attribute argument is used, to delete an attribute assignment for a device.

Options The following options are supported:

`-a` Add a device to the device table using the specified attributes. The device must be referenced by its `alias`.

`-d` Remove a device from the device table, when executed without the *attributes* argument. Used with the *attribute* argument, it deletes the given attribute specification for *device* from the table.

`-m` Modify a device entry in the device table. If an entry already exists, it adds any specified attributes that are not defined. It also modifies any attributes which already have a value with the value specified by this command.

Operands The following operands are supported:

alias Designates the alias of the device to be added.

attribute Designates a device attribute to be added, modified, or deleted. Can be any of the device attributes described under `DEVICE ATTRIBUTES` except `alias`. This prevents an accidental modification or deletion of a device's alias from the table.

device Designates the pathname or alias of the device whose attribute is to be added, modified, or removed.

value Designates the value to be assigned to a device's attribute.

Device Attributes The following list shows the standard device attributes, used by applications such as `ufsdump(1M)` and `ufsrestore(1M)`, which can be defined for a device. You are not limited to this list, you can define any attribute you like.

`alias` The unique name by which a device is known. No two devices in the database may share the same alias name. The name is limited in length to 14 characters and should contain only alphanumeric characters and the following special characters if they are escaped with a backslash: underscore (_), dollar sign ($), hyphen (–), and period (.).

`bdevice`	The pathname to the block special device node associated with the device, if any. The associated major/minor combination should be unique within the database and should match that associated with the `cdevice` field, if any. (It is the administrator's responsibility to ensure that these `major/minor` numbers are unique in the database.)
`capacity`	The capacity of the device or of the typical volume, if removable.
`cdevice`	The pathname to the character special device node associated with the device, if any. The associated major/minor combination should be unique within the database and should match that associated with the `bdevice` field, if any. (It is the administrator's responsibility to ensure that these `major/minor` numbers are unique in the database.)
`cyl`	Used by the command specified in the `mkfscmd` attribute.
`desc`	A description of any instance of a volume associated with this device (such as floppy diskette).
`dpartlist`	The list of disk partitions associated with this device. Used only if `type=disk`. The list should contain device aliases, each of which must have `type=dpart`.
`dparttype`	The type of disk partition represented by this device. Used only if `type=dpart`. It should be either `fs` (for file system) or `dp` (for data partition).
`erasecmd`	The command string that, when executed, erases the device.
`fmtcmd`	The command string that, when executed, formats the device.
`fsname`	The file system name on the file system administered on this partition, as supplied to the `/usr/sbin/labelit` command. This attribute is specified only if `type=dpart` and `dparttype=fs`.
`gap`	Used by the command specified in the `mkfscmd` attribute.
`mkfscmd`	The command string that, when executed, places a file system on a previously formatted device.
`mountpt`	The default mount point to use for the device. Used only if the device is mountable. For disk partitions where `type=dpart` and `dparttype=fs`, this attribute should specify the location where the partition is normally mounted.
`nblocks`	The number of blocks in the file system administered on this partition. Used only if `type=dpart` and `dparttype=fs`.
`ninodes`	The number of inodes in the file system administered on this partition. Used only if `type=dpart` and `dparttype=fs`.
`norewind`	The name of the character special device node that allows access to the serial device without rewinding when the device is closed.

`pathname` Defines the pathname to an i-node describing the device (used for non-block or character device pathnames, such as directories).

`type` A token that represents inherent qualities of the device. Standard types include: 9-track, ctape, disk, directory, diskette, dpart, and qtape.

`volname` The volume name on the file system administered on this partition, as supplied to the `/usr/sbin/labelit` command. Used only if `type=dpart` and `dparttype=fs`.

`volume` A text string used to describe any instance of a volume associated with this device. This attribute should not be defined for devices which are not removable.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Command syntax was incorrect, an invalid option was used, or an internal error occurred.

`2` The device table could not be opened for reading, or a new device table could not be created.

`3` If executed with the `-a` option, indicates that an entry in the device table with the alias `alias` already exits. If executed with the `-m` or `-d` options, indicates that no entry exists for device *device*.

`4` Indicates that `-d` was requested and one or more of the specified attributes were not defined for the device.

Files `/etc/device.tab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `devattr(1M)`, `putdgrp(1M)`, `ufsdump(1M)`, `ufsrestore(1M)`, `attributes(5)`

System Administration Guide: Basic Administration

Name putdgrp – edits device group table

Synopsis

```
putdgrp [-d] dgroup [device]...
```

Description `putdgrp` modifies the device group table. It performs two kinds of modification. It can modify the table by creating a new device group or removing a device group. It can also change group definitions by adding or removing a device from the group definition.

When the command is invoked with only a *dgroup* specification, the command adds the specified group name to the device group table if it does not already exist. If the `-d` option is also used with only the *dgroup* specification, the command deletes the group from the table.

When the command is invoked with both a *dgroup* and a *device* specification, it adds the given device name(s) to the group definition. When invoked with both arguments and the `-d` option, the command deletes the device name(s) from the group definition.

When the command is invoked with both a *dgroup* and a *device* specification and the device group does not exist, it creates the group and adds the specified devices to that new group.

Options The following options are supported:

`-d` Delete the group or, if used with *device*, delete the device from a group definition.

Operands The following operands are supported:

dgroup Specify a device group name.

device Specify the pathname or alias of the device that is to be added to, or deleted from, the device group.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Command syntax was incorrect, an invalid option was used, or an internal error occurred.

`2` Device group table could not be opened for reading or a new device group table could not be created.

`3` If executed with the `-d` option, indicates that an entry in the device group table for the device group *dgroup* does not exist and so cannot be deleted. Otherwise, indicates that the device group *dgroup* already exists and cannot be added.

`4` If executed with the `-d` option, indicates that the device group *dgroup* does not have as members one or more of the specified devices. Otherwise, indicates that the device group *dgroup* already has one or more of the specified devices as members.

Examples

EXAMPLE 1 Adding a new device group.

The following example adds a new device group:

```
example#  putdgrp floppies
```

EXAMPLE 2 Adding a device to a device group.

The following example adds a device to a device group:

```
example#  putdgrp floppies diskette2
```

EXAMPLE 3 Deleting a device group.

The following example deletes a device group:

```
example#  putdgrp -d floppies
```

EXAMPLE 4 Deleting a device from a device group.

The following example deletes a device from a device group:

```
example#  putdgrp -d floppies diskette2
```

Files

`/etc/dgroup.tab`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`listdgrp(1M)`, `putdev(1M)`, `attributes(5)`

Name pwck, grpck – password/group file checkers

Synopsis

```
/usr/sbin/pwck [filename]

/usr/sbin/grpck [filename]
```

Description `pwck` scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and the program-to-use-as-shell exist. The default password file is `/etc/passwd`.

`grpck` verifies all entries in the group file. This verification includes a check of the number of fields, group name, group ID, whether any login names belong to more than `NGROUPS_MAX` groups, and that all login names appear in the password file. The default group file is `/etc/group`.

All messages regarding inconsistent entries are placed on the `stderr` stream.

Files `/etc/group`

`/etc/passwd`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `getpwent(3C)`, `group(4)`, `passwd(4)`, `attributes(5)`

Diagnostics Group entries in `/etc/group` with no login names are flagged.

`Group file '`*filename*`' is empty`
The `/etc/passwd` or `/etc/group` file is an empty file.

`cannot open file` *filename*`: No such file or directory`
The `/etc/passwd` or `/etc/group` file does not exist.

Notes If no filename argument is given, `grpck` checks the local group file, `/etc/group`, and also makes sure that all login names encountered in the checked group file are known to the system `getpwent(3C)` routine. This means that the login names may be supplied by a network name service.

Name pwconv – installs and updates /etc/shadow with information from /etc/passwd

Synopsis

```
pwconv
```

Description The `pwconv` command creates and updates `/etc/shadow` with information from `/etc/passwd`.

`pwconv` relies on a special value of 'x' in the password field of `/etc/passwd`. This value of 'x' indicates that the password for the user is already in `/etc/shadow` and should not be modified.

If the `/etc/shadow` file does not exist, this command will create `/etc/shadow` with information from `/etc/passwd`. The command populates `/etc/shadow` with the user's login name, password, and password aging information. If password aging information does not exist in `/etc/passwd` for a given user, none will be added to `/etc/shadow`. However, the last changed information will always be updated.

If the `/etc/shadow` file does exist, the following tasks will be performed:

Entries that are in the `/etc/passwd` file and not in the `/etc/shadow` file will be added to the `/etc/shadow` file.

Entries that are in the `/etc/shadow` file and not in the `/etc/passwd` file will be removed from `/etc/shadow`.

Password attributes (for example, password and aging information) that exist in an `/etc/passwd` entry will be moved to the corresponding entry in `/etc/shadow`.

The `pwconv` command can only be used by the super-user.

Files

`/etc/opasswd`

`/etc/oshadow`

`/etc/passwd`

`/etc/shadow`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `passwd(1)`, `passmgmt(1M)`, `usermod(1M)`, `passwd(4)`, `attributes(5)`

Diagnostics `pwconv` exits with one of the following values:

`0` SUCCESS.

`1` Permission denied.

`2` Invalid command syntax.

3 Unexpected failure. Conversion not done.

4 Unexpected failure. Password file(s) missing.

5 Password file(s) busy. Try again later.

6 Bad entry in `/etc/shadow` file.

Name quot – summarize file system ownership

Synopsis

```
quot [-acfhnv] filesystem...

quot -a [-cfhnv]
```

Description `quot` displays the number of blocks (1024 bytes) in the named *filesystem* (one or more) currently owned by each user. There is a limit of 2048 blocks. Files larger than this will be counted as a 2048 block file, but the total block count will be correct.

Options The following options are supported:

`-a` Generate a report for all mounted file systems.

`-c` Display three columns giving a file size in blocks, the number of files of that size, and a cumulative total of blocks containing files of that size or a smaller size.

`-f` Display three columns giving, for each user, the number of blocks owned, the count of number of files, and the user name. This option is incompatible with the `-c` and `-v` options.

`-h` Estimate the number of blocks in the file. This does not account for files with holes in them.

`-n` Attach names to the list of files read from standard input. `quot -n` cannot be used alone, because it expects data from standard input. For example, the pipeline

```
ncheck myfilesystem | sort +0n | quot -n myfilesystem
```

will produce a list of all files and their owners. This option is incompatible with all other options.

`-v` In addition to the default output, display three columns containing the number of blocks not accessed in the last 30, 60, and 90 days.

Operands *filesystem* mount-point of the filesystem(s) being checked

Usage See `largefile(5)` for the description of the behavior of `quot` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status

`0` Successful operation.

`32` Error condition (bad or missing argument, bad path, or other error).

Files

`/etc/mnttab` Lists mounted file systems.

`/etc/passwd` Used to obtain user names

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also du(1), mnttab(4), passwd(4), attributes(5), largefile(5)

Notes This command can only be used by the super-user.

Name quota – display a user's ufs or zfs file system disk quota and usage

Synopsis

```
quota [-v] [username]
```

Description `quota` displays users' UFS or ZFS disk usage and limits. Only the super-user may use the optional *username* argument to view the limits of other users.

`quota` without options only display warnings about mounted file systems where usage is over quota. Remotely mounted file systems which do not have quotas turned on are ignored.

username can be the numeric UID of a user.

Options `-v` Display user's quota on all mounted file systems where quotas exist.

Usage See `largefile(5)` for the description of the behavior of `quota` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files `/etc/mnttab` list of currently mounted filesystems

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `edquota(1M)`, `quotaon(1M)`, `quotacheck(1M)`, `repquota(1M)`, `rquotad(1M)`, `attributes(5)`, `largefile(5)`, `zones(5)`

Notes `quota` displays quotas for NFS mounted UFS- or ZFS-based file systems if the `rquotad` daemon is running. See `rquotad(1M)`. In a `zones(5)` environment, `quota` displays quotas only for the zone in which it is invoked.

`quota` can display entries for the same file system multiple times for multiple mount points. For example,

```
# quota -v user1
```

might display identical quota information for `user1` at the mount points `/home/user1`, `/home/user2`, and `/home/user`, if all three mount points are mounted from the same file system with quotas turned on.

Name quotacheck – ufs file system quota consistency checker

Synopsis

```
quotacheck [-fp] [-v] filesystem...

quotacheck -a [-fpv]
```

Description `quotacheck` examines each mounted ufs file system, builds a table of current disk usage, and compares this table against the information stored in the file system's disk quota file. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated.

filesystem is either a file system mount point or the block device on which the file system resides.

`quotacheck` expects each file system to be checked to have a quota file named `quotas` in the root directory. If none is present, `quotacheck` will not check the file system.

`quotacheck` accesses the character special device in calculating the actual disk usage for each user. Thus, the file systems that are checked should be quiescent while `quotacheck` is running.

Options The following options are supported:

`-a` Check the file systems which `/etc/mnttab` indicates are ufs file systems. These file systems must be read-write mounted with disk quotas enabled, and have an `rq` entry in the `mntopts` field in `/etc/vfstab`.

`-f` Force check on file systems with logging enabled. Use in combination with the `-p` option.

`-p` Check quotas of file systems in parallel. For file systems with logging enabled, no check is performed unless the `-f` option is also specified.

`-v` Indicate the calculated disk quotas for each user on a particular file system. `quotacheck` normally reports only those quotas modified.

Usage See `largefile(5)` for the description of the behavior of `quotacheck` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/mnttab` Mounted file systems

`/etc/vfstab` List of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also edquota(1M), quota(1M), quotaon(1M), repquota(1M), attributes(5), largefile(5), quotactl(7I), mount_ufs(1M)

Name quotaon, quotaoff – turn ufs file system quotas on and off

Synopsis

```
quotaon [-v] filesystem...

quotaon -a [-v]

quotaoff [-v] filesystem...

quotaoff -a [-v]
```

Description `quotaon` turns on disk quotas for one or more ufs file systems.

Before a file system may have quotas enabled, a file named `quotas`, owned by root, must exist in the root directory of the file system. See `edquota(1M)` for details on how to modify the contents of this file.

`quotaoff` turns off disk quotas for one or more ufs file systems.

The file systems specified must already be mounted.

These commands update the `mntopts` field of the appropriate entries in `/etc/mnttab` to indicate when quotas are on or off for each file system. If quotas are on, the string `quota` will be added to `mntopts`; if quotas are off, the `quota` string is not present.

filesystem must be either the mount point of a file system, or the block device on which the file system resides.

Options

quotaon

`-a` This option is normally used at boot time to enable quotas. It applies only to those file systems in `/etc/vfstab` which have "rq" in the `mntopts` field, are currently mounted "rw", and have a `quotas` file in the root directory.

`-v` Display a message for each file system after quotas are turned on.

quotaoff

`-a` Force all file systems in `/etc/mnttab` to have their quotas disabled.

`-v` Display a message for each file system affected.

Usage See `largefile(5)` for the description of the behavior of `quotaon` and `quotaoff` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`/etc/mnttab` mounted file systems

`/etc/vfstab` list of default parameters for each file system

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also edquota(1M), quota(1M), quotacheck(1M), repquota(1M), mnttab(4), vfstab(4), attributes(5), largefile(5), quotactl(7I)

Name raidctl – RAID hardware utility

Synopsis

```
raidctl -C "disks" [-r raid_level] [-z capacity] [-s stripe_size] [-f]
     controller

raidctl -d [-f] volume

raidctl -F filename [-f] controller...

raidctl -a {set | unset} -g disk {volume | controller}

raidctl -p "param=value" [-f] volume

raidctl -c [-f] [-r raid_level] disk1 disk2 [disk3...]

raidctl -l -g disk controller

raidctl -l volume

raidctl -l controller...

raidctl [-l]

raidctl -S [volume | controller]

raidctl -S -g disk controller

raidctl -h
```

Description The `raidctl` utility is a hardware RAID configuration tool that supports different RAID controllers by providing a CLI (command-line interface) to end-users to create, delete or display RAID volume(s). The utility can also used to set properties of a volume, assign hot-spare (HSP) disks to volumes or controllers, and to update firmware/fcode/BIOS for RAID controllers.

The `raidctl` utility requires privileges that are controlled by the underlying file-system permissions. Only privileged users can manipulate the RAID system configuration. If a non-privileged user attempts to run `raidctl`, the command fails with an exit status of 1.

The `raidctl` utility, as described in this man page, defines a broad set of command line options to provide management for full-featured RAID controllers. However, support for a given option depends on two elements:

- the presence of a software driver
- the firmware level of the RAID device

The dependency on a software driver is due to the design of `raidctl`. The utility is built on a common library that enables the insertion of plug-in modules for different drivers. Currently, the Solaris operating system is shipped with a plug-in for the `mpt` driver. This plug-in does not support all of the `raidctl` options. On a given storage device, options might be further limited by the device's firmware level.

The level of support for the various `raidctl` options cannot be determined by `raidctl`. The user must rely on the documentation for his RAID controller or hardware platform.

Currently, `raidctl` Currently, `raidctl` provides some level of support for the following RAID controllers:

- LSI1020 SCSI HBA
- LSI1030 SCSI HBA
- LSI1064 SAS HBA
- LSI1068 SAS HBA

All of the above HBAs are maintained by the `mpt` driver, on X86-32/64 and SPARC platforms.

Options The following options are supported:

`-C` *"disks"* [`-r` *raid_level*] [`-z` *capacity*] [`-s` *stripe_size*] [`-f`] controller
Create a RAID volume using specified disks.

When creating a RAID volume using this option, the identity of the newly created volume is automatically generated and raidctl reports it to the user.

The argument specified by this option contains the elements used to form the volume that will be created. Elements can be either disks or sub-volumes, where disks are separated by space(s) and a sub-volume is a set of disks grouped by parenthesis. All disks should be in `C.ID.L` expression (for example, `0.1.2` represents a physical disk of channel 0, target id 1, and logical unit number 2). The argument must match the RAID level specified by the `-r` option, even if it's omitted. This means the argument can only be:

for RAID 0
At least 2 disks

for RAID 1
Only 2 disks

for RAID 1E
At least 3 disks

for RAID 5
At least 3 disks

for RAID 10
At least 2 sub-volumes, each sub-volume must be formed by 2 disks

for RAID 50
At least 2 sub-volumes, each sub-volume must be formed by at least 3 disks, and the disk amount in each sub-volume should be the same

For example, the expression "0.0.0 0.1.0" means that the 2 specified disks form a RAID volume, which can either be a RAID 0 or a RAID 1 volume. "(0.0.0 0.1.0)(0.2.0 0.3.0)" means that the first 2 disks and the last 2 disks form 2 sub-volumes, and that these 2 sub-volumes form a RAID 10 volume. See the `EXAMPLES` section for more samples.

The `-r` option specifies the RAID level of the volume that will be created. Possible levels are 0, 1, 1E, 5, 10, 50. If this option is omitted, `raidctl` creates a RAID 1 volume by default.

The `-z` option specifies the capacity of the volume that will be created. The unit can be tera-bytes, giga-bytes, or mega-bytes (for example, 2t, 10g, 20m, and so on). If this option is omitted, `raidctl` calculates the maximum capacity of the volume that can be created by the specified disks and uses this value to create the volume.

The `-s` option specifies the stripe size of the volume that will be created. The possible values are 512, 1k, 2k, 4k, 8k, 16k, 32k, 64k, or 128k. If this option is omitted, `raidctl` chooses an appropriate value for the volume (for example, 64k).

In some cases, the creation of a RAID volume may cause data on specified disks to be lost (for instance, on LSI1020, LSI1030, SAS1064, or SAS1068 HBAs), and `raidctl` prompts the user for confirmation about the creation. Use the `-f` option to force the volume creation without prompting the user for confirmation.

The controller argument is used to identify which RAID controller the specified disks belongs. The `-l` option can be used to list the controller's ID number.

`-d` [`-f`] *volume*
Delete the RAID volume specified as volume. The volume is specified in canonical form (for example, `c0t0d0`).

When a volume is deleted, all data is lost. Therefore, unless the `-f` option is specified, `raidctl` prompts the user for confirmation before deleting the volume.

When a RAID 1 volume is deleted from a LSI1020, LSI1030, SAS1064, or SAS1068 HBA, the primary and secondary disks are "split". If the volume was in `SYNCING` state, the primary will contain the data, and the secondary will not. If the volume state was `OPTIMAL`, both disks will contain a complete image of the data.

`-F` *filename* [`-f`] *controller…*
Update the firmware running on the specified controller(s). The `raidctl` utility prompts the user for confirmation of this action, unless the `-f` option is provided.

`-a` {`set` | `unset`} `-g` *disk* {`volume` | `controller`}
If the volume is specified, `raidctl` sets or unsets the disk as a local hot-spare disk dedicated to the volume, depending on the value specified by the `-a` option. If the controller is specified, `raidctl` sets or unsets the disk as a global hot-spare disk.

`-p` *"param=value"* [`-f`] *volume*
Change the property value for a given RAID volume. This option can be used to change cache write policy or to activate a volume. When changing the cache write policy, *param* should be the string `wp` (`SET_WR_POLICY`), and *value* can be either `on` or `off`. When used to activate a volume, *param* should be `state` and *value* should be `activate`.

Changing a RAID volume's property may affect the internal behavior of the RAID controller, so `raidctl` prompts the user for a confirmation before applying the change, unless the `-f` option is specified.

`-c` [`-f`] [`-r` *raid_level*] *disk1 disk2* [*disk3*...]
Create a volume using the specified disks. This is an alternative to the `-C` option with similar functionality. This option is preserved for compatibility reasons, but only works with LSI1020, LSI1030, SAS1064, and SAS1068 HBAs to create RAID 0, RAID 1, or RAID 1E volumes. For other HBAs, the user can only use the `-C` option.

The `-r` option can be used to specify the RAID level of the target volume. If the `-r` option is omitted, `raidctl` will create a RAID 1 volume.

Disks must be specified in Solaris canonical format (for example, c0t0d0).

Creating a RAID 1 volume with this option replaces the contents of disk2 with the contents of disk1.

When the user creates a RAID volume with this option, the RAID volume assumes the identity of disk1. Other disks become invisible and the RAID volume appears as one disk.

Creating a volume with this option is by default interactive. The user must answer a prompt affirmatively to create the volume. Use the `-f` option to force the volume creation without prompting the user for confirmation.

`-l` -g *disk controller*
Display information about the specified disk of the given controller. The output includes the following information:

Disk
Displays the disk in `C.ID.L` expression disk.

Vendor
Displays the vendor ID string.

Product
Displays the product ID string.

Capacity
Displays the total capacity of the disk.

Status
Displays the current status of disk. The status can be either "`GOOD`" (operating normally), "`FAILED`" (non-functional), or "`MISSING`" (disk not present).

HSP
Indicates if the disk has been set as a global hot-spare disk, local hot-spare disk, or a normal one. If it is a local hot-spare disk, all volumes which this disk is assigned to are displayed.

GUID
GUID string for the specified disk. This is an additional datum and might be unavailable in some cases.

`-l` *volume*
Display information about the specified volume. The output includes the following information:

Volume
Displays volume in canonical format.

Sub
Displays sub-volumes, if the specified volume is of RAID 10 or RAID 50 volume.

Disk
Displays all disks that form the specified volume.

Stripe Size
Displays the stripe size of the volume.

Status
Displays the status of the specified volume, or the sub-volumes or disks that form the specified volume. For an inactive volume, the status should be `INACTIVE`; otherwise it can be `OPTIMAL` (operating optimally), `DEGRADED` (operating with reduced functionality), `FAILED` (non-functional), or `SYNC` (disks are syncing). For a disk, the status can be `GOOD`, `FAILED`, or `MISSING`.

Cache
Indicates whether the cache is applied to `I/O` write activities. The cache can be either "`ON`" or "`OFF`".

RAID level
Displays the RAID level. The RAID level can be either 0, 1, 1E, 5, 10, or 50.

`-l` *controller ...*
Display information about the specified controller(s). The output includes the following information:

Controller
Displays the RAID controller's ID number.

Type
Displays the RAID controller's product type.

fw_version
Displays the controller's firmware version.

`[-l]`
List all RAID related objects that the `raidctl` utility can manipulate, including all available RAID controllers, RAID volumes, and physical disks. The `-l` option can be omitted.

The output includes the following information:

Controller
Displays the RAID controller's ID number.

Volume
Displays the logical RAID volume name.

Disk
Displays the RAID disk in `C.ID.L` expression.

`-S` [`volume` | `controller`]
Takes a snapshot of the RAID configuration information including all available RAID devices, RAID controllers, volumes, and disks.

Each line of the output specifies a RAID device and its related information, separated by space(s). All volumes and disks belong to the last specified controller.

The output lists the following information:

Controller
Displays the controller ID number, and the controller type string in double-quotation marks.

Volume
Displays the RAID volume name, number of component disks, the `C.ID.L` expression of the component disks, the RAID level, and the status. The status can be either `OPTIMAL`, `DEGRADED`, `FAILED`, or `SYNCING`.

Disk
Displays the `C.ID.L` expression of the disk, and the status. The status can be either `GOOD`, `FAILED`, or `HSP` (disk has been set as a stand-by disk).

If a volume or a controller is specified, a snapshot is only taken of the information for the specified volume or controller.

`-S` -g *disk controller*
Takes a snapshot of the information for the specified disk.

`-h`
Print out the usage string.

Examples

EXAMPLE 1 Creating the RAID Configuration

The following command creates a RAID 0 volume of 10G on controller 0, and the stripe size will be set to 64k:

```
# raidctl -C "0.0.0 0.2.0" -r 0 -z 10g -s 64k 0
```

The following command creates a RAID 1 volume on controller 2:

```
# raidctl -C "0.0.0 1.1.0" -r 1 2
```

The following command creates a RAID 5 volume on controller 2:

```
# raidctl -C "0.0.0 0.1.0 0.2.0" -r 5 2
```

The following command creates a RAID 10 volume on controller 0:

EXAMPLE 1 Creating the RAID Configuration *(Continued)*

```
# raidctl -C "(0.0.0 0.1.0)(0.2.0 0.3.0)" -r 10 0
```

The following command creates a RAID 50 volume on controller 0:

```
# raidctl -C "(0.0.0 0.1.0 0.2.0)(0.3.0 0.4.0 0.5.0)" -r 50 0
```

EXAMPLE 2 Displaying the RAID Configuration

The following command displays all available controllers, volumes, and disks:

```
# raidctl -l

Controller: 0
Controller: 2
         Volume:c2t0d0
         Disk: 0.0.0
         Disk: 0.1.0
         Disk: 0.2.0
         Disk: 0.3.0(HSP)
```

The following command displays information about controller 2:

```
# raidctl -l 2

Controller      Type            Fw_version
------------------------------------------------------------
c2              LSI 1030        1.03.39.00
```

The following command displays information about the specified volume:

```
# raidctl -l c2t0d0

Volume                  Size    Stripe  Status  Cache   RAID
         Sub                     Size                    Level
                Disk
----------------------------------------------------------------
c2t0d0                 10240M  64K     OPTIMAL ON      RAID5
                0.0.0   5120M           GOOD
                0.1.0   5120M           GOOD
                0.2.0   5120M           GOOD
```

The following command displays information about disk 0.0.0 on controller 0:

```
# raidctl -l -g 0.0.0  0

Disk    Vendor  Product         Firmware Capacity        Status  HSP
-------------------------------------------------------------------
0.0.0   HITACHI H101473SCSUN72G SQ02     68.3G           GOOD    N/A
GUID:2000000cca02536c
```

EXAMPLE 3 Deleting the RAID Configuration

The following command deletes a volume:

```
# raidctl -d c0t0d0
```

EXAMPLE 4 Updating Flash Images on the Controller

The following command updates flash images on the controller 0:

```
# raidctl -F lsi_image.fw 0
```

EXAMPLE 5 Setting or Unsetting a Hot-Spare Disk

The following command sets disk 0.3.0 on controller 2 as a global hot-spare disk:

```
# raidctl -a set -g 0.3.0 2
```

The following command sets disk 0.3.0 on controller 2 as a local hot-spare disk to volume c2t0d0:

```
# raidctl -a set -g 0.3.0 c2t0d0
```

The following command converts disk 0.3.0 on controller 2 from a global hot-spare disk to a normal one:

```
# raidctl -a unset -g 0.3.0 2
```

The following command removes disk 0.3.0 from being a local hot-spare disk from volume c2t0d0:

```
# raidctl -a unset -g 0.3.0 c2t0d0
```

EXAMPLE 6 Setting the Volume's Property

The following command sets the write policy of the volume to "`off`":

```
# raidctl -a set -p "wp=off" c0t0d0
```

EXAMPLE 7 Creating Volumes with the - c Option

The following command creates a RAID 1 volume:

```
# raidctl -c c0t0d0 c0t1d0
```

The following command creates a RAID 0 volume:

```
# raidctl -c -r 0 c0t1d0 c0t2d0 c0t3d0
```

EXAMPLE 8 Taking a Snapshot of the RAID Configuration

The following command takes a snapshot of all RAID devices:

```
# # raidctl -S

1 "LSI 1030"
```

EXAMPLE 8 Taking a Snapshot of the RAID Configuration *(Continued)*

```
c1t1d0 2 0.2.0 0.3.0 1 DEGRADED
0.2.0 GOOD
0.3.0 FAILED
```

The following command takes a snapshot about volume c1t0d0:

```
# raidctl -S c1t0d0

c1t0d0 2 0.0.0 0.1.0 1 OPTIMAL
```

The following command takes a snapshot about disk 0.1.0 on controller 1:

```
# raidctl -S -g 0.1.0 1

0.1.0 GOOD
```

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`1`
: Invalid command line input or permission denied.

`2`
: Request operation failed.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Volatile

See Also

`attributes(5)`, `mpt(7D)`

System Administration Guide: Basic Administration

Warnings

Do not create raid volumes on internal SAS disks if you are going to use the Solaris Multipathing I/O feature (also known as MPxIO). Creating a new raid volume under Solaris Multipathing will give your root device a new GUID which does not match the GUID for the existing devices. This will cause a boot failure since your root device entry in `/etc/vfstab` will not match.

Notes The `-z` option is not supported on systems that use the `mpt` driver and LSI RAID controllers.

Name ramdiskadm – administer ramdisk pseudo device

Synopsis

```
/usr/sbin/ramdiskadm -a name size [g | m | k | b]

/usr/sbin/ramdiskadm -d name

/usr/sbin/ramdiskadm
```

Description The `ramdiskadm` command administers `ramdisk(7D)`, the ramdisk driver. Use `ramdiskadm` to create a new named ramdisk device, delete an existing named ramdisk, or list information about existing ramdisks.

Ramdisks created using `ramdiskadm` are not persistent across reboots.

Options The following options are supported:

`-a` *name size* Create a ramdisk named *name* of size *size* and its corresponding block and character device nodes.

name must be composed only of the characters a-z, A-Z, 0-9, _ (underbar), and - (hyphen), but it must not begin with a hyphen. It must be no more than 32 characters long. Ramdisk names must be unique.

The size can be a decimal number, or, when prefixed with `0x`, a hexadecimal number, and can specify the size in bytes (no suffix), 512-byte blocks (suffix `b`), kilobytes (suffix `k`), megabytes (suffix `m`) or gigabytes (suffix `g`). The size of the ramdisk actually created might be larger than that specified, depending on the hardware implementation.

If the named ramdisk is successfully created, its block device path is printed on standard out.

`-d` *name* Delete an existing ramdisk of the name *name*. This command succeeds only when the named ramdisk is not open. The associated memory is freed and the device nodes are removed.

You can delete only ramdisks created using `ramdiskadm`. It is not possible to delete a ramdisk that was created during the boot process.

Without options, `ramdiskadm` lists any existing ramdisks, their sizes (in decimal), and whether they can be removed by `ramdiskadm` (see the description of the `-d` option, above).

Examples **EXAMPLE 1** Creating a 2MB Ramdisk Named `mydisk`

```
# ramdiskadm -a mydisk 2m
/dev/ramdisk/mydisk
```

EXAMPLE 2 Listing All Ramdisks

```
# ramdiskadm
Block Device                    Size  Removable
/dev/ramdisk/miniroot     134217728    No
/dev/ramdisk/certfs         1048576    No
/dev/ramdisk/mydisk         2097152    Yes
```

Exit Status

`ramdiskadm` returns the following exit values:

`0` Successful completion.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr
Interface Stability	Evolving

See Also

`attributes(5)`, `ramdisk(7D)`

Notes

The abilities of `ramdiskadm` and the privilege level of the person who uses the utility are controlled by the permissions of `/dev/ramdiskctl`. Read access allows query operations, for example, listing device information. Write access is required to do any state-changing operations, for example, creating or deleting ramdisks.

As shipped, `/dev/ramdiskctl` is owned by `root`, in group `sys`, and mode 0644, so all users can do query operations but only root can perform state-changing operations. An administrator can give write access to non-privileged users, allowing them to add or delete ramdisks. However, granting such ability entails considerable risk; such privileges should be given only to a trusted group.

Name rcapadm – configure resource capping daemon

Synopsis

```
rcapadm
```

```
rcapadm [ [-n] -E | -D]
     [-i interval=value,...,interval=value] [-c percent]
     [-z zonename -m maxvalue]
```

Description The `rcapadm` command allows a user with the privileges described below to configure various attributes of the resource capping daemon. If used without arguments, `rcapadm` displays the current status of the resource capping daemon if it has been configured. See `rcapd(1M)` for more information.

In the current release of the Solaris operating environment, `rcapadm` is available to users with all privileges and to users who have the Process Management profile in their list of profiles. The System Administrator role includes the Process Management profile.

Options

`-c` *percent*
: Set the minimum physical memory utilization for memory cap enforcement. Caps will not be enforced until the physical memory available to processes is low. The *percent* value should be in the range 0 to 100. The minimum (and default) value is `0`, which means that memory caps are always enforced.

`-D`
: Disable the resource capping daemon so that it will not be started when the system is booted. Also stop the resource capping daemon now, if the `-n` option is not specified and it is currently running.

`-E`
: Enable the resource capping daemon so that it will be started each time the system is booted. Also start the resource capping daemon now, if the `-n` option is not specified and it is not currently running.

`-i` *interval=value,...,interval=value*
: Set intervals for various periodic operations performed by `rcapd`. All intervals are specified in seconds. You can set the following intervals:

  `scan` — The interval at which `rcapd` scans for new processes. The default scan interval is every 15 seconds. The minimum value is 1 second.

  `sample` — The interval of process resident set size sampling. The default sample interval is every 5 seconds. The minimum value is 1 second.

`report` The interval at which various paging statistics are updated by `rcapd`, in seconds. These statistics can be viewed by using `rcapstat`(1SRM). The default reporting interval is every 5 seconds. When the interval is set to `0`, statistics will not be updated.

Note – Paging refers to the act of relocating portions of memory, called pages, to or from physical memory. `rcapd` pages out the most infrequently used pages.

`config` The reconfiguration interval, in seconds. At each reconfiguration event, `rcapd` checks its configuration file for updates, and scans the project databases for new project caps. The default reconfiguration interval is every 60 seconds. The minimum interval is `0`. When the interval is set to `0`, no periodic reconfiguration occurs, although the running daemon can still be reconfigured by sending it `SIGHUP`.

`-m` *maxvalue* Used in conjunction with the `-z` option. Specifies a value for `rcap.max-rss`, a dynamically-set cap on the usage of physical memory for the zone specified by `-z`. You can apply a scale (`K`, `M`, `G`, `T`) to the value you specify. `K` means kilobyte; `M`, megabyte; `G`, gigabyte; and `T`, terabyte. For example, `100M` is 100 megabytes.

`-n` Do not affect the running state of the resource capping daemon when enabling or disabling it.

`-z` *zonename* Used in conjunction with the `-m` option. Specifies the zone for which you are dynamically specifying a cap on physical memory usage (using `-m`).

Note – To set a persistent cap on memory usage within a zone, use `zonecfg`(1M).

Examples

EXAMPLE 1 Configuring the Resource Capping Daemon with Immediate Enforcement

```
# rcapadm -E -i scan=15,sample=5,report=5,config=60 -c 0
```

EXAMPLE 2 Specifying a Resource Cap for a Zone

The command shown below specifies the maximum amount of memory that can be consumed by a specified zone. Note that this value lasts only until the next reboot. To set a persistent cap, use `zonecfg(1M)`.

```
# rcapadm -z testzone -m 512M
```

Exit Status

The following exit values are returned:

`0` Successful completion. The modifications to the current configuration were valid and made successfully.

`1` An error occurred. A fatal error occurred either in obtaining or modifying the resource capping configuration.

`2` Invalid command-line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcapu
Interface Stability	Evolving

The `-z` and `-m` options are committed interfaces.

See Also

`rcapstat(1)`, `rcapd(1M)`, `zonecfg(1M)`, `project(4)`, `attributes(5)`, `zones(5)`

"Physical Memory Control Using the Resource Capping Daemon" in *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*

Name rcapd – resource cap enforcement daemon

Synopsis

```
rcapd [-d]
```

Description The `rcapd` daemon enforces resource caps on collections of processes. Per-project and per-zone physical memory caps are supported. For information about projects, see `project(4)`. For zones information, see `zones(5)`

When the resident set size (RSS) of a collection of processes exceeds its cap, `rcapd` takes action and reduces the RSS of the collection.

The virtual memory system divides physical memory into segments known as pages. To read data from a file into memory, the virtual memory system reads in individual pages. To reduce resource consumption, the daemon can page out, or relocate, infrequently used pages to an area outside of physical memory.

In the `project` file, caps are defined for projects that have positive values for the following project attribute:

`rcap.max-rss` The total amount of physical memory, in bytes, that is available to the project's member processes

See `project`(4) for a description of project attributes.

For a system with one or more zones, you can dynamically set the `rcap.max-rss` value for a zone with `rcapadm(1M)`. To set a persistent cap on memory usage within a zone, you use `zonecfg(1M)`.

You configure `rcapd` through the use of `rcapadm(1M)`. The daemon can be monitored with `rcapstat(1)`. Configuration changes are incorporated into `rcapd` by sending it `SIGHUP` (see `kill(1)`), or according to the configuration interval (see `rcapadm(1M)`).

Options The following option is supported:

`-d` Enable debug mode. Messages are displayed on the invoking user's terminal.

Examples **EXAMPLE 1** Setting Resident Set Size Cap Attribute

The following line in the `/etc/project` database sets an RSS cap of 1073741824 bytes for a project named `foo`.

```
foo:100::foo,root::rcap.max-rss=10737418240
```

Exit Status The following exit values are returned:

0 Successful completion.

1 An error occurred.

2 Invalid command-line options were specified.

Files `/etc/project` Project database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcapu
Interface Stability	Evolving

See Also `rcapstat`(1), `svcs(1)`, `rcapadm(1M)`, `zonecfg(1M)`, `svcadm(1M)`, `project(4)`, `attributes(5)`, `smf(5)`, `zones(5)`

"Physical Memory Control Using the Resource Capping Daemon" in *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*

Notes If killed with `SIGKILL`, `rcapd` can leave processes in a stopped state. Use `SIGTERM` to cause `rcapd` to terminate properly.

A collection's RSS can exceed its cap for some time before the cap is enforced, even if sufficient pageable memory is available. This period of time can be reduced by shortening the RSS sampling interval with `rcapadm`.

The `rcapd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/rcap:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name rctladm – display or modify global state of system resource controls

Synopsis

```
rctladm [-lu] [-e action] [-d action] [name...]
```

Description The `rctladm` command allows the examination and modification of active resource controls on the running system. An instance of a resource control is referred to as an *rctl*. See `setrctl(2)` for a description of an `rctl`; see `resource_controls(5)` for a list of the `rctl`s supported in the current release of the Solaris operating system. Logging of `rctl` violations can be activated or deactivated system-wide and active `rctl`s (and their state) can be listed.

An `rctladm` command without options is the equivalent of an `rctladm` with the `-l` option. See the description of `-l` below.

Options The following options are supported:

`-d` *action*
`-e` *action*
Disable (`-d`) or enable (`-e`) the global action on the specified `rctl`s. If no `rctl` is specified, no action is taken and an error status is returned. You can use the special token `all` with the disable option to deactivate all global actions on a resource control.

You can set the `syslog` action to a specific degree by assigning a severity level. To do this, specify `syslog=`*level*, where *level* is one of the string tokens given as valid severity levels in `syslog(3C)`. You can omit the common `LOG_` prefix on the severity level. Note that not all `rctl`s support the `syslog` action. See `resource_controls(5)`.

`-l`
List information about `rctl`s. The name, global event actions and statuses, and global flags are displayed. If one or more name operands are specified, only those `rctl`s matching the names are displayed.

`-u`
Configure resource controls based on the contents of `/etc/rctladm.conf`. Any name operands are ignored.

Operands The following operands are supported:

name
The name of the `rctl` to operate on. Multiple `rctl` names can be specified. If no names are specified, and the list action has been specified, then all `rctl`s are listed. If the enable or disable action is specified, one or more `rctl` names must be specified.

Examples **EXAMPLE 1** Activating System Logging for Specific Violations

The following command activates system logging of all violations of `task.max-lwps`.

```
# rctladm -e syslog task.max-lwps
#
```

EXAMPLE 2 Examining the Current Status of a Specific Resource

The following command examines the current status of the `task.max-lwps` resource.

```
$ rctladm -l task.max-lwps
task.max-lwps               syslog=DEBUG
$
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`1`
A fatal error occurred. A message is written to standard error to indicate each resource control for which the operation failed. The operation was successful for any other resource controls specified as operands.

`2`
Invalid command line options were specified.

Files

`/etc/rctladm.conf`
Each time `rctladm` is executed, it updates the contents of `rctladm.conf` with the current configuration.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu

See Also

`setrctl(2)`, `getrctl(2)`, `prctl(1)`, `rctlblk_get_global_flags(3C)`, `rctlblk_get_global_action(3C)`, `attributes(5)`, `resource_controls(5)`

Notes

By default, there is no global logging of `rctl` violations.

Name rdate – set system date from a remote host

Synopsis `rdate` *hostname*

Description `rdate` sets the local date and time from the *hostname* given as an argument. You must have the authorization `solaris.system.date` on the local system. Typically, `rdate` is used in a startup script.

`rdate` requests are responded to by the "time" service on the specified host. To enable the "time" service, use the following commands:

```
svcadm enable time:stream
svcadm enable time:dgram
```

Usage The `rdate` command is IPv6–enabled. See `ip6(7P)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmdc

See Also `inetd(1M)`, `inetd.conf(4)`, `attributes(5)`, `ip6(7P)`

Name reboot – restart the operating system

Synopsis

```
/usr/sbin/reboot [-dlnq] [boot_arguments]
```

Description The `reboot` utility restarts the kernel. The kernel is loaded into memory by the PROM monitor, which transfers control to the loaded kernel.

Although `reboot` can be run by the super-user at any time, `shutdown(1M)` is normally used first to warn all users logged in of the impending loss of service. See `shutdown(1M)` for details.

The `reboot` utility performs a `sync(1M)` operation on the disks, and then a multi-user reboot is initiated. See `init(1M)` for details. On x86 systems, reboot may also update the boot archive as needed to ensure a successful reboot.

The `reboot` utility normally logs the reboot to the system log daemon, `syslogd(1M)`, and places a shutdown record in the login accounting file `/var/adm/wtmpx`. These actions are inhibited if the `-n` or `-q` options are present.

Normally, the system reboots itself at power-up or after crashes.

Options The following options are supported:

`-d` Force a system crash dump before rebooting. See `dumpadm(1M)` for information on configuring system crash dumps.

`-l` Suppress sending a message to the system log daemon, `syslogd(1M)` about who executed `reboot`.

`-n` Avoid calling `sync(2)` and do not log the reboot to `syslogd(1M)` or to `/var/adm/wtmpx`. The kernel still attempts to sync filesystems prior to reboot, except if the `-d` option is also present. If `-d` is used with `-n`, the kernel does not attempt to sync filesystems.

`-q` Quick. Reboot quickly and ungracefully, without shutting down running processes first.

Operands The following operands are supported:

boot_arguments An optional *boot_arguments* specifies arguments to the `uadmin(2)` function that are passed to the boot program and kernel upon restart. The form and list of arguments is described in the `boot(1M)` and `kernel(1M)` man pages.. If the arguments are specified, whitespace between them is replaced by single spaces unless the whitespace is quoted for the shell. If the *boot_arguments* begin with a hyphen, they must be preceded by the `--` delimiter (two hyphens) to denote the end of the `reboot` argument list.

Examples

EXAMPLE 1 Passing the `-r` and `-v` Arguments to `boot`

In the following example, the delimiter `--` (two hyphens) must be used to separate the options of `reboot` from the arguments of `boot(1M)`.

```
example# reboot -dl -- -rv
```

EXAMPLE 2 Rebooting Using a Specific Disk and Kernel

The following example reboots using a specific disk and kernel.

```
example# reboot disk1 kernel.test/unix
```

Files

`/var/adm/wtmpx` login accounting file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`mdb(1)`, `boot(1M)`, `dumpadm(1M)`, `fsck(1M)`, `halt(1M)`, `init(1M)`, `kernel(1M)`, `shutdown(1M)`, `sync(1M)`, `syslogd(1M)`, `sync(2)`, `uadmin(2)`, `reboot(3C)`, `attributes(5)`

Notes

The `reboot` utility does not execute the scripts in `/etc/rc`*num*`.d` or execute shutdown actions in `inittab(4)`. To ensure a complete shutdown of system services, use `shutdown(1M)` or `init(1M)` to reboot a Solaris system.

Name regadm – Solaris system registration utility

Synopsis

```
regadm auth [-u username] [-p password_file]

regadm register [-q] [-d]

regadm list

regadm clear

regadm set [-n name] [-v value]

regadm status

regadm disable

regadm enable
```

Description The `regadm` utility is used by a privileged system administrator to register products on a given Solaris system with My Oracle Support (MOS), and to manage the background auto-registration service that runs at system boot time. Registered products are extracted from the Service Tag Registry (see `stclient(1M)`) and uploaded to MOS by means of a secure HTTPS connection.

You can use `regadm` to explicitly register products. In addition, an `smf(5)` service runs at boot time that attempts to refresh the registration if the product installation base has changed. This service can be administratively enabled or disabled by means of subcommands of regadm. By default, the service is enabled.

Options The following options are supported:

`-d`
Dry-run mode. Show the registration information that will be transmitted to MOS but do not perform the actual registration. Output is displayed in XML format.

`-p` *password_file*
Used with the authentication subcommand, the `-p` option specifies a file name containing the password associated with the MOS username (the `-u` option). This should be a single-line file and it can be immediately removed after running this command.

`-q`
Quiet mode. Do not show any output during a registration.

`-u` *username*
Used with the authentication subcommand, the `-u` option specifies the MOS user name to be associated with registrations of products on a given system.

Sub-commands The subcommands for `regadm` are as follows:

`auth`
Each system must be authenticated with valid MOS credentials before its products can be registered. A successful authentication is made persistent and thus only needs to be done once.

`clear`
Remove all configured authentication and HTTPS connectivity information. This effectively configures an anonymous registration with no HTTP proxy information.

`disable`
Disable the auto-registration service at system boot time.

`enable`
Enable the auto-registration service at system boot time.

`list`
Displays the current authenticated MOS user name and the network connectivity information needed for HTTPS communication with MOS. If no authentication has yet been done (by means of the `auth` command), the anonymous account will be displayed.

`register`
Registers each product from the Solaris Service Tag registry (see `stclient(1M)`) with MOS, using the current authentication credentials. If no credentials have been supplied, the registration is performed using an anonymous MOS account.

`set`
Configure HTTP proxy information, to be used for HTTPS communication with the MOS backend.

`status`
Report the status (enabled/disabled) of the Solaris auto-registration `smf(5)` service. See the `disable` and `enable` subcommands.

Examples

EXAMPLE 1 Authenticating with MOS Interactively

The following command is used to interactively authenticate this system with MOS. You will be prompted for your support username and password.

```
# regadm auth
```

EXAMPLE 2 Authenticating Non-Interactively

This is similar to the interactive example, but specifies the MOS username and password on the command line.

```
# regadm auth -u "pat.smith@example.com" -p mypassword
```

The file `mypassword` must contain the MOS password.

EXAMPLE 3 Setting HTTP Proxy Information

The following command sequence configures an HTTP proxy and port, used for HTTPS communication with the MOS.

```
# regadm set -n http_proxy -v webcache.example.com
# regadm set -n http_proxy_port -v 8080
```

EXAMPLE 4 Setting HTTP Proxy Authentication Information

Some HTTP proxies require authentication. This example shows the configuring of a username/password for basic RFC 2617 HTTP authentication. Note that a filename containing the password (`mypassword`) is specified, not the actual password itself.

```
# regadm set -n http_proxy_user -v webuser
# regadm set -n http_proxy_user_pw -v mypassword
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWautoregu
Interface Stability	Committed

See Also

`stclient(1M)`, `attributes(5)`, `smf(5)`

Name rem_drv – remove a device driver from the system

Synopsis

```
rem_drv [-b basedir] device_driver
```

Description The `rem_drv` command informs the system that the device driver *device_driver* is no longer valid. If possible, `rem_drv` unloads *device_driver* from memory. `rem_drv` also updates the system driver configuration files.

If `rem_drv` has been executed, the next time the system is rebooted it automatically performs a reconfiguration boot (see `kernel(1M)`).

Options The following options are supported:

`-b` *basedir*
Sets the path to the root directory of the diskless client. Used on the server to execute `rem_drv` for a client. The client machine must be rebooted to unload the driver.

Note – The root file system of any non-global zones must not be referenced with the `-b` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-C`
Remove dangling device attribute nodes bound to the driver being removed. This causes any device ownership or permissions customizations made to any node not to be preserved if the driver is added back. Recommended for use when reprovisioning a machine from one configuration or use to another where past administrative customizations might not be desired.

Examples

EXAMPLE 1 Removing the `sd` Driver

The following example removes the `sd` driver from use:

```
example% rem_drv sd
```

EXAMPLE 2 Removing a Diskless Client

The following example removes the driver from the `sun1` diskless client. The driver is not uninstalled or unloaded until the client machine is rebooted.

```
example% rem_drv -b /export/root/sun1 sd
```

Note the caveat on the use of the `-b` option in the description of that option, above.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also add_drv(1M), kernel(1M), update_drv(1M), attributes(5), zones(5), devfs(7FS)

Name remove_allocatable – remove entries from allocation databases

Synopsis

```
/usr/sbin/remove_allocatable  [-f] -n name

/usr/sbin/remove_allocatable  [-f] [-d] -t dev-type
```

Description `remove_allocatable` removes entries of user allocatable devices from the device allocation mechanism. `remove_allocatable` also removes entries of some non-allocatable devices, such as printers, whose label range is managed by the mechanism.

Options The following options are supported:

`-d` Removes system-supplied default attributes of the device type that is specified with `-t`.

`-f` Force the removal of an entry. `remove_allocatable` exits with an error if this option is not specified when an entry with the specified device name no longer exists.

`-n` *name* Removes the entry for the device *name*.

`-t` *dev-type* Removes devices of type *dev-type*.

Exit Status When successful, `remove_allocatable` returns an exit status of `0` (true). `remove_allocatable` returns a nonzero exit status in the event of an error. The exit codes are as follows:

1 Invocation syntax error

2 Unknown system error

3 Device *name* or *dev-type* not found. This error occurs only when the `-f` option is not specified.

4 Permission denied. User does not have DAC or MAC access to database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	See below.

The invocation is Uncommitted. The options are Uncommitted. The output is Not-an-Interface.

See Also `allocate(1)`, `deallocate(1)`, `add_allocatable(1M)`, `attributes(5)`, `device_clean(5)`

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

Name removef – remove a file from software database

Synopsis

```
removef [ [-M] -R root_path] [-V fs_file] pkginst path...
removef [ [-M] -R root_path] [-V fs_file] -f pkginst
```

Description `removef` informs the system that the user, or software, intends to remove a pathname. Output from `removef` is the list of input pathnames that may be safely removed (no other packages have a dependency on them).

Options The following options are supported:

`-f`
After all files have been processed, `removef` should be invoked with the `-f` option to indicate that the removal phase is complete.

`-M`
Instruct `removef` not to use the $*root_path*`/etc/vfstab` file for determining the client's mount points. This option assumes the mount points are correct on the server and it behaves consistently with Solaris 2.5 and earlier releases.

`-R` *root_path*
Define the full path name of a directory to use as the *root_path*. All files, including package system information files, are relocated to a directory tree starting in the specified *root_path*. The *root_path* may be specified when installing to a client from a server (for example, `/export/root/client1`).

`removef` inherits the value of the `PKG_INSTALL_ROOT` environment variable. (See ENVIRONMENT VARIABLES, below.) If `PKG_INSTALL_ROOT` is set, such as when the `-R` option is used with `pkgadd(1M)` or `pkgrm(1M)`

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-V` *fs_file*
Specify an alternative *fs_file* to map the client's file systems. For example, used in situations where the $*root_path*`/etc/vfstab` file is non-existent or unreliable.

Operands The following operands are supported:

path
The pathname to be removed.

pkginst
The package instance from which the pathname is being removed.

Examples

EXAMPLE 1 Using `removef`

The following example uses the `removef` command in an optional pre-install script:

```
echo "The following files are no longer part of this package
      and are being removed."
removef $PKGINST /myapp/file1 /myapp/file2 |
while read pathname
do
     echo "$pathname"
     rm -f $pathname
done
removef -f $PKGINST || exit 2
```

Environment Variables

`removef` inherits the value of the following environment variable. This variable is set when `pkgadd(1M)` or `pkgrm(1M)`

`PKG_INSTALL_ROOT`
: If present, defines the full path name of a directory to use as the system's `PKG_INSTALL_ROOT` path. All product and package information files are then looked for in the directory tree, starting with the specified `PKG_INSTALL_ROOT` path. If not present, the default system path of `/` is used.

Exit Status

`0`
: Successful completion.

`>0`
: An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`pkginfo(1)`, `pkgmk(1)`, `pkgparam(1)`, `pkgproto(1)`, `pkgtrans(1)`, `installf(1M)`, `pkgadd(1M)`, `pkgask(1M)`, `pkgchk(1M)`, `pkgrm(1M)`, `attributes(5)`, `largefile(5)`

Application Packaging Developer's Guide

Notes

Package commands are `largefile(5)`-aware. They handle files larger than 2 GB in the same way they handle smaller files. In their current implementations, `pkgadd(1M)`, `pkgtrans(1)` and other package commands can process a datastream of up to 4 GB.

Name repquota – summarize quotas for a ufs file system

Synopsis

```
repquota [-v] filesystem...

repquota -a [-v]
```

Description `repquota` prints a summary of the disk usage and quotas for the specified ufs file systems. The current number of files and amount of space (in kilobytes) is printed for each user along with any quotas created with `edquota(1M)`.

The *filesystem* must have the file `quotas` in its root directory.

Only the super-user may view quotas which are not their own.

Options The following options are supported:

`-a` Report on all mounted ufs file systems that have `rq` in the `mntopts` field of the `/etc/vfstab` file.

`-v` Report quotas for all users, even those who do not consume resources.

Usage See `largefile(5)` for the description of the behavior of `repquota` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `edquota(1M)`, `quota(1M)`, `quotacheck(1M)`, `quotaon(1M)`, `attributes(5)`, `largefile(5)`, `quotactl(7I)`

Name re-preinstall – installs the JumpStart software on a system

Synopsis

```
cdrom-mnt-pt/Solaris_XX/Tools/Boot/usr/sbin/install.d/re-preinstall
     [-m Solaris_boot_dir]
     [-k platform_name] target-slice
```

Description `re-preinstall` installs the JumpStart software (preinstall boot image) on a system, so you can power-on the system and have it automatically install the Solaris software (perform a JumpStart installation on the system). When you turn on a re-preinstalled system, the system looks for the JumpStart software on the system's default boot disk. *All* new SPARC systems have the JumpStart software already preinstalled. The *XX* in Solaris_*XX* is the version number of the Solaris release being used.

You can use the `re-preinstall` command in several ways. The most common way is to run `re-preinstall` on a system to install the JumpStart software on its own default boot disk. This is useful if you want to restore a system to its original factory conditions. (See the first procedure described in `EXAMPLES`.)

You can also run `re-preinstall` on a system to install JumpStart software on any attached disk (non-boot disk). After you install the JumpStart software on a disk, you can move the disk to a different system and perform a JumpStart installation on the different system. (See the second procedure described in `EXAMPLES`.)

`re-preinstall` creates a standard file system on the specified *target-slice* (usually slice 0), and `re-preinstall` makes sure there is enough space on the *target-slice* for the JumpStart software. If sufficient space is not available, `re-preinstall` fails with the following message:

`re-preinstall:` *target-slice* `too small` *xx* `Megabytes required`

You can use the `format(1M)` command to create sufficient space on the *target-slice* for the JumpStart software.

Options The following options are supported:

`-k` *platform_name* — Platform name of the system that will use the disk with the JumpStart software. The default is the platform name of the system running `re-preinstall`. (Use the `uname(1)` command (`-i` option) to determine a system's platform name.)

`-m` *Solaris_boot_dir* — Absolute path to the `Solaris_`*XX*`/Tools/Boot` subdirectory of a mounted Solaris CD or a Solaris CD copied to disk that `re-preinstall` uses to install the JumpStart software. The default is root (`/`), which is where the Solaris CD is mounted in single-user mode.

Operands The following operands are supported:

target-slice — Device name of the disk slice where the JumpStart software will be installed (usually slice 0), for example, `c0t3d0s0`.

Examples

EXAMPLE 1 Installing the JumpStart Software on a System's Own Default Boot Disk

The following procedure installs the JumpStart software on a system's own default boot disk:

1. From the `ok` prompt, boot the system from the Solaris media CD or DVD in single-user mode:

   ```
   ok boot cdrom -s
   ```

2. The following command installs the Jumpstart software on the System default boot disk, `c0t0d0s0` on a Solaris system:

   ```
   example# /usr/sbin/install.d/re-preinstall c0t0d0s1
   ```

3. Reboot the slice:

   ```
   example# reboot disk:b
   ```

EXAMPLE 2 Installing the JumpStart Software on a System's Attached (non-boot) Disk

The following procedure installs the JumpStart software on a system's attached (non-boot) disk:

1. Mount the Solaris CD or DVD if `vold(1M)` is not running or CD or DVD is not mounted.
2. Use the `format(1M)` command to determine the target-slice where JumpStart will be installed.
3. Use the `uname(1)` command (`-i` option) to determine the platform name of the system that will use the re-preinstalled disk
4. Run `re-preinstall` with the `-m` *Solaris_boot_dir* option if the Solaris CD or DVD is not mounted on `/cdrom`.

   The following command installs the JumpStart software on the system's attached disk for a system with a Sun4u kernel architecture, and it uses the Solaris CD or DVD mounted with `vold(1M)` on a Solaris system:

   ```
   example#  /cdrom/cdrom/s1/usr/bin/install.d/re-preinstall  -m
             /cdrom/cdrom/s1 -k sun4u c0t2d0s0
   ```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` An error has occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcdrom (Solaris CD, SPARC Platform Edition)

See Also uname(1), eeprom(1M), format(1M), mount(1M), vold(1M), attributes(5)

Solaris 10 Installation Guide: Basic Installations

Name rmmount – removable media mounter for CD-ROM, floppy, Jaz drive, and others

Synopsis

```
/usr/sbin/rmmount [-D]
```

Description The `rmmount` utility is a removable media mounter that is executed by Volume Management whenever a removable medium, such as a CD-ROM or a floppy, is inserted. The Volume Management daemon, `vold(1M)`, manages removable media. `rmmount` can also be called by using `volrmmount(1)`.

Upon insertion of a medium and following invocation of the `volcheck(1)` command, `rmmount` determines what type of file system (if any) is on that medium. If a file system is present, `rmmount` mounts the file system in one of the locations listed below.

For a diskette (floppy):

`/floppy/floppy0`	symbolic link to mounted floppy in local floppy drive
`/floppy/floppy_name`	mounted named floppy
`/floppy/unnamed_floppy`	mounted unnamed floppy

For a CD-ROM or a DVD-ROM:

`/cdrom/cdrom0`	symbolic link to mounted CD-ROM in local CD-ROM drive
`/cdrom/CD-ROM_name`	mounted named CD-ROM
`/cdrom/CD-ROM_name/partition`	mounted named CD-ROM with partitioned file system
`/cdrom/unnamed_cdrom`	mounted unnamed CD-ROM

For a Zip drive:

`/rmdisk/zip0`	symbolic link to mounted Zip medium in local Zip drive
`/rmdisk/`*Zip_name*	mounted named Zip medium
`/rmdisk/`*Zip_name*`/partition`	mounted named Zip medium with partitioned file system
`/rmdisk/unnamed_zip`	mounted unnamed Zip medium

For a Jaz drive:

`/rmdisk/jaz0`	symbolic link to mounted Jaz medium in local Jaz drive
`/rmdisk/`*Jaz_name*	mounted named Jaz medium
`/rmdisk/`*Jaz_name*`/partition`	mounted named Jaz medium with partitioned file system
`/rmdisk/unnamed_Jaz`	mounted unnamed Jaz medium

For a generic “rmdisk” drive:

`/rmdisk/rmdisk0`	symbolic link to mounted removable medium in local removable medium drive
`/rmdisk/`*`rmdisk_name`*	mounted named removable medium
`/rmdisk/`*`rmdisk_name`*`/partition`	mounted named removable medium with partitioned file system
`/rmdisk/unnamed_rmdisk`	mounted unnamed removable medium

If the media is read-only (for example, a CD-ROM or a floppy with write-protect tab set), the file system is mounted read-only.

If a file system is not identified, `rmmount` does not mount a file system. See the *System Administration Guide: Basic Administration* for more information on the location of CD-ROM, floppy, and other media without file systems. Also see `volfs`(7FS).

If a file system type has been determined, it is then checked to see that it is "clean." If the file system is "dirty," `fsck -p` (see `fsck`(1M)) is run in an attempt to clean it. If `fsck` fails, the file system is mounted read-only.

After the mount is complete, "actions" associated with the media type are executed. These actions allow for the notification to other programs that new media are available. These actions are shared objects and are described in the configuration file, `/etc/rmmount.conf`. See `rmmount.conf`(4).

Actions are executed in the order in which they appear in the configuration file. The action function can return either `1` or `0`. If it returns `0`, no further actions will be executed. This allows the function to control which applications are executed.

In order to execute an action, `rmmount` performs a `dlopen`(3C) on the shared object and calls the action function defined within it. The definition of the interface to actions can be found in `/usr/include/rmmount.h`.

File systems mounted by `rmmount` are always mounted with the `nosuid` flag set, thereby disabling setuid programs and access to block or character devices in that file system. Upon ejection, `rmmount` unmounts mounted file systems and executes actions associated with the media type. If a file system is "busy" (that is, it contains the current working directory of a live process), the ejection will fail.

Options

`-D` Turn on the debugging output from the `rmmount` `dprintf` calls.

Files

`/etc/rmmount.conf`	removable media mounter configuration file
`/usr/lib/rmmount/*.so.1`	shared objects used by `rmmount`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWvolu

See Also volcancel(1), volcheck(1), volmissing(1), volrmmount(1), fsck(1M), vold(1M), dlopen(3C), rmmount.conf(4), vold.conf(4), attributes(5), volfs(7FS)

System Administration Guide: Basic Administration

Name rmt – remote magtape protocol module

Synopsis `/usr/sbin/rmt`

Description `rmt` is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection. `rmt` is normally started up with an `rexec(3SOCKET)` or `rcmd(3SOCKET)` call.

The `rmt` program accepts requests that are specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. All responses are in ASCII and in one of two forms. Successful commands have responses of:

`A`*number*`\n` where *number* is an ASCII representation of a decimal number.

Unsuccessful commands are responded to with:

`E`*error-number*`\n`*error-message*`\n` where *error-number* is one of the possible error numbers described in `Intro(3)`, and *error-message* is the corresponding error string as printed from a call to `perror(3C)`.

The protocol consists of the following commands:

`S\n` Return the status of the open device, as obtained with a `MTIOCGET` `ioctl` call. If the operation was successful, an "ack" is sent with the size of the status buffer, then the status buffer is sent (in binary).

`C`*device*`\n` Close the currently open device. The *device* specified is ignored.

`I`*operation*`\n`*count*`\n` Perform a `MTIOCOP` `ioctl(2)` command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the *mt_op* and *mt_count* fields of the structure used in the `ioctl` call. When the operation is successful the return value is the *count* parameter.

`L`*offset*`\nwhence\n` Perform an `lseek(2)` operation using the specified parameters. The response value is returned from the `lseek` call.

`O`*device*`\n`*mode*`\n` Open the specified *device* using the indicated *mode*. *device* is a full pathname, and *mode* is an ASCII representation of a decimal number suitable for passing to `open(9E)`. If a device is already open, it is closed before a new open is performed.

`R`*count*`\n` Read *count* bytes of data from the open device. `rmt` performs the requested `read(9E)` and responds with `A`*count-read*`\n` if the read was successful; otherwise an error in standard format is returned. If the read was successful, the data read is sent.

`W`*count*`\n` Write data onto the open device. `rmt` reads *count* bytes from the connection, aborting if a premature EOF is encountered. The

response value is returned from the write(9E) call.

Any other command causes rmt to exit.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmdc

See Also ufsdump(1M), ufsrestore(1M), Intro(3), ioctl(2), lseek(2), perror(3C), rcmd(3SOCKET), rexec(3SOCKET), attributes(5), mtio(7I), open(9E), read(9E), write(9E)

Diagnostics All responses are of the form described above.

Bugs Do not use this for a remote file access protocol.

Name rndc – name server control utility

Synopsis

```
rndc [-V] [-b src-addr] [-c config-file] [-k key-file] [-s server]
     [-p port] [-y key_id] command
```

Description The `rndc` utility controls the operation of a name server. It supersedes the `ndc` utility that was provided in previous BIND releases. If `rndc` is invoked with no command line options or arguments, it prints a short summary of the supported commands and the available options and their arguments.

The `rndc` utility communicates with the name server over a TCP connection, sending commands authenticated with digital signatures. The only supported authentication algorithm in the current versions of `rndc` and `named(1M)` is HMAC-MD5, which uses a shared secret on each end of the connection. This algorithm provides TSIG-style authentication for the command request and the name server's response. All commands sent over the channel must be signed by a *key_id* known to the server.

The `rndc` utility reads a configuration file to determine how to contact the name server and decide what algorithm and key it should use.

Options The following options are supported:

`-b` *source-address*
Use *source-address* as the source address for the connection to the server. Multiple instances are permitted to allow setting of both the IPv4 and IPv6 source addresses.

`-c` *config-file*
Use *config-file* as the configuration file instead of the default `/etc/rndc.conf`.

`-k` *key-file*
Use *key-file* as the key file instead of the default, `/etc/rndc.key`. The key in `/etc/rndc.key` is used to authenticate commands sent to the server if the *config-file* does not exist.

`-s` *server*
The *server* argument is the name or address of the server that matches a server statement in the configuration file for `rndc`. If no server is supplied on the command line, the host named by the default-server clause in the options statement of the `rndc` configuration file is used.

`-p` *port*
Send commands to TCP port *port* instead of BIND 9's default control channel port, 953.

`-V`
Enable verbose logging.

`-y` *key_id*
Use the key *key_id* from the configuration file. The *key_id* argument must be known by `named` with the same algorithm and secret string for control message validation to succeed. If no *key_id* is specified, `rndc` will first look for a key clause in the server statement of the

server being used, or if no server statement is present for that host, then the `default-key` clause of the options statement. The configuration file contains shared secrets that are used to send authenticated control commands to name servers. It should therefore not have general read or write access.

For the complete set of commands supported by `rndc`, see the *BIND 9 Administrator Reference Manual* or run `rndc` without arguments to see its help message.

Limitations The `rndc` utility does not support all the commands of the BIND 8 `ndc` utility.

There is no way to provide the shared secret for a *key_id* without using the configuration file.

Several error messages tend toward the cryptic.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dns/bind
Interface Stability	Volatile

See Also `named(1M)`, `rndc-confgen(1M)`, `rndc.conf(4)`, `attributes(5)`

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name rndc-confgen – rndc key generation tool

Synopsis

```
rndc-confgen [-ah] [-b keysize] [-c keyfile] [-k keyname]
     [-p port] [-r randomfile] [-s address] [-t chrootdir]
     [-u user]
```

Description The `rndc-confgen` utility generates configuration files for `rndc(1M)`. This utility can be used as a convenient alternative to writing by hand the `rndc.conf(4)` file and the corresponding `controls` and `key` statements in `named.conf`. It can also be run with the `-a` option to set up a `rndc.key` file and avoid altogether the need for a `rndc.conf` file and a `controls` statement.

Options The following options are supported:

`-a`
Perform automatic `rndc` configuration. This option creates a file `rndc.key` in `/etc` (or however *sysconfdir* was specified when BIND was built) that is read by both `rndc` and `named(1M)` on startup. The `rndc.key` file defines a default command channel and authentication key allowing `rndc` to communicate with `named` with no further configuration.

Running `rndc-confgen` with `-a` specified allows BIND 9 and `rndc` to be used as drop-in replacements for BIND 8 and `ndc`, with no changes to the existing BIND 8 `named.conf` file.

If a more elaborate configuration than that generated by `rndc-confgen -a` is required, for example if `rndc` is to be used remotely, you should run `rndc-confgen` without the `-a` option and set up `rndc.conf` and `named.conf` files, as directed.

`-b` *keysize*
Specify the size of the authentication key in bits. The *keysize* argument must be between 1 and 512 bits; the default is 128.

`-c` *keyfile*
Used with the `-a` option to specify an alternate location for `rndc.key`.

`-h`
Print a short summary of the options and arguments to `rndc-confgen`.

`-k` *keyname*
Specify the key name of the `rndc` authentication key. The *keyname* argument must be a valid domain name. The default is `rndc-key`.

`-p` *port*
Specify the command channel port where `named` listens for connections from `rndc`. The default is 953.

`-r` *randomfile*
Specify a source of random data for generating the authorization. By default, `/dev/random` is used. The *randomdev* argument specifies the name of a character device or file containing random data to be used instead of the default. The special value `keyboard` indicates that keyboard input should be used.

`-s` *address*
Specify the IP address where `named` listens for command channel connections from `rndc`. The default is the loopback address 127.0.0.1.

`-t` *chrootdir*
Used with the `-a` option to specify a directory where `named` will run after the root directory is changed with `chroot(2)`. An additional copy of the `rndc.key` will be written relative to this directory so that it will be found by the `named` in the new directory.

`-u` *user*
Used with the `-a` option to set the owner of the `rndc.key` file generated. If `-t` is also specified only the file in the chroot area has its owner changed.

Examples

EXAMPLE 1 Create Automatic `rndc` Configuration

The following command creates an automatic `rndc` configuration, so that `rndc` can be used immediately.

```
# rndc-confgen -a
```

EXAMPLE 2 Print a Sample `rndc.conf` File

The following command prints a sample `rndc.conf` file with corresponding `controls` and `key` statements. These statements can subsequently be manually inserted in the file `named.conf`.

```
# rndc-confgen
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/dns/bind
Interface Stability	Volatile

See Also

`chroot(2)`, `named(1M)`, `rndc(1M)`, `rndc.conf(4)`, `attributes(5)`

See the BIND 9 *Administrator's Reference Manual*. As of the date of publication of this man page, this document is available at `https://www.isc.org/software/bind/documentation`.

Name roleadd – administer a new role account on the system

Synopsis

```
roleadd [-c comment] [-d dir] [-e expire] [-f inactive]
     [-g group] [-G group [, group...]] [-m [-k skel_dir]]
     [-u uid [-o]] [-s shell]
     [-A authorization [,authorization...]] [-K key=value] role

roleadd -D [-b base_dir] [-e expire] [-f inactive]
     [-g group] [-A authorization [,authorization...]]
     [-P profile [,profile...] [-K key=value]]
```

Description `roleadd` adds a role entry to the `/etc/passwd` and `/etc/shadow` and `/etc/user_attr` files. The `-A` and `-P` options respectively assign authorizations and profiles to the role. Roles cannot be assigned to other roles. The `-K` option adds a *key=value* pair to `/etc/user_attr` for a role. Multiple *key=value* pairs can be added with multiple `-K` options.

`roleadd` also creates supplementary group memberships for the role (`-G` option) and creates the home directory (`-m` option) for the role if requested. The new role account remains locked until the `passwd(1)` command is executed.

Specifying `roleadd -D` with the `-g`, `-b`, `-f`, `-e`, or `-K` option (or any combination of these option) sets the default values for the respective fields. See the `-D` option. Subsequent `roleadd` commands without the `-D` option use these arguments.

The system file entries created with this command have a limit of 512 characters per line. Specifying long arguments to several options can exceed this limit.

The role (`role`) field accepts a string of no more than eight bytes consisting of characters from the set of alphabetic characters, numeric characters, period (`.`), underscore (`_`), and hyphen (`-`). The first character should be alphabetic and the field should contain at least one lower case alphabetic character. A warning message is written if these restrictions are not met. A future Solaris release might refuse to accept role fields that do not meet these requirements.

The `role` field must contain at least one character and must not contain a colon (`:`) or a newline (`\n`).

Options The following options are supported:

`-A` *authorization* — One or more comma separated authorizations defined in `auth_attr(4)`. Only a user or role who has grant rights to the authorization can assign it to an account

`-b` *base_dir* — The default base directory for the system if `-d` *dir* is not specified. *base_dir* is concatenated with the account name to define the home directory. If the `-m` option is not used, *base_dir* must exist.

`-c` *comment* — Any text string. It is generally a short description of the role. This information is stored in the role's `/etc/passwd` entry.

`-d` *dir*
: The home directory of the new role. It defaults to *base_dir*/*account_name*, where *base_dir* is the base directory for new login home directories and *account_name* is the new role name.

`-D`
: Display the default values for *group*, *base_dir*, *skel_dir*, *shell*, *inactive*, *expire* and *key=value* pairs. When used with the `-g`, `-b`, `-f`, or `-K`, options, the `-D` option sets the default values for the specified fields. The default values are:

group	`other` (GID of 1)
base_dir	`/home`
skel_dir	`/etc/skel`
shell	`/bin/pfsh`
inactive	`0`
expire	Null
auths	Null
profiles	Null
key=value (pairs defined in `user_attr(4)`	not present

`-e` *expire*
: Specify the expiration date for a role. After this date, no user is able to access this role. The expire option argument is a date entered using one of the date formats included in the template file `/etc/datemsk`. See `getdate(3C)`.

  If the date format that you choose includes spaces, it must be quoted. For example, you can enter `10/6/90` or `October 6, 1990`. A null value (`""`) defeats the status of the expired date. This option is useful for creating temporary roles.

`-f` *inactive*
: The maximum number of days allowed between uses of a role ID before that ID is declared invalid. Normal values are positive integers. A value of `0` defeats the status.

`-g` *group*
: An existing group's integer ID or character-string name. Without the `-D` option, it defines the new role's primary group membership and defaults to the default group. You can reset this default value by invoking `roleadd -D -g` *group*.

`-G` *group*
: An existing group's integer ID or character-string name. It defines the new role's supplementary group membership. Duplicates between *group* with the `-g` and `-G` options are ignored. No more than `NGROUPS_MAX` groups can be specified.

`-k` *skel_dir* — A directory that contains skeleton information (such as `.profile`) that can be copied into a new role's home directory. This directory must already exist. The system provides the `/etc/skel` directory that can be used for this purpose.

`-K` *key=value* — A *key=value* pair to add to the role's attributes. Multiple `-K` options can be used to add multiple *key=value* pairs. The generic `-K` option with the appropriate key can be used instead of the specific implied key options (`-A` and `-P`). See `user_attr(4)` for a list of valid *key=value* pairs. The "type" key is not a valid key for this option. Keys can not be repeated.

`-m` — Create the new role's home directory if it does not already exist. If the directory already exists, it must have read, write, and execute permissions by *group*, where *group* is the role's primary group.

`-o` — This option allows a UID to be duplicated (non-unique).

`-P` *profile* — One or more comma-separated execution profiles defined in `prof_attr(4)`.

`-s` *shell* — Full pathname of the program used as the user's shell on login. It defaults to an empty field causing the system to use `/bin/pfsh` as the default. The value of *shell* must be a valid executable file.

`-u` *uid* — The UID of the new role. This UID must be a non-negative decimal integer below `MAXUID` as defined in `<sys/param.h>`. The UID defaults to the next available (unique) number above the highest number currently assigned. For example, if UIDs 100, 105, and 200 are assigned, the next default UID number is 201. (UIDs from `0-99` are reserved for possible use in future applications.)

Files

`/etc/datemsk`

`/etc/passwd`

`/etc/shadow`

`/etc/group`

`/etc/skel`

`/usr/include/limits.h`

`/etc/user_attr`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also passwd(1), pfexec(1), profiles(1), roles(1), users(1B), groupadd(1M), groupdel(1M), groupmod(1M), grpck(1M), logins(1M), pwck(1M), userdel(1M), usermod(1M), getdate(3C), auth_attr(4), passwd(4), prof_attr(4), user_attr(4), attributes(5)

Diagnostics In case of an error, `roleadd` prints an error message and exits with a non-zero status.

The following indicates that `login` specified is already in use:

```
UX: roleadd: ERROR: login is already in use. Choose another.
```

The following indicates that the *uid* specified with the `-u` option is not unique:

```
UX: roleadd: ERROR: uid uid is already in use. Choose another.
```

The following indicates that the *group* specified with the `-g` option is already in use:

```
UX: roleadd: ERROR: group group does not exist. Choose another.
```

The following indicates that the *uid* specified with the `-u` option is in the range of reserved UIDs (from `0-99`):

```
UX: roleadd: WARNING: uid uid is reserved.
```

The following indicates that the *uid* specified with the `-u` option exceeds `MAXUID` as defined in `<sys/param.h>`:

```
UX: roleadd: ERROR: uid uid is too big. Choose another.
```

The following indicates that the `/etc/passwd` or `/etc/shadow` files do not exist:

```
UX: roleadd: ERROR: Cannot update system files - login cannot be created.
```

Notes If a network nameservice such as NIS or NIS+ is being used to supplement the local `/etc/passwd` file with additional entries, `roleadd` cannot change information supplied by the network nameservice.

Name roledel – delete a role's login from the system

Synopsis

```
roledel [-r] role
```

Description The `roledel` utility deletes a role account from the system and makes the appropriate account-related changes to the system file and file system. `roledel` also removes the role from each user's list of assumable roles.

Options The following options are supported:

`-r` Remove the role's home directory from the system. This directory must exist. The files and directories under the home directory will no longer be accessible following successful execution of the command.

Operands The following operands are supported:

role An existing role name to be deleted.

Exit Status The following exit values are returned:

`0` Successful completion.

`2` Invalid command syntax. A usage message for the `roledel` command is displayed.

`6` The account to be removed does not exist.

`8` The account to be removed is in use.

`10` Cannot update the `/etc/group` or `/etc/user_attr` file but the login is removed from the `/etc/passwd` file.

`12` Cannot remove or otherwise modify the home directory.

Files

`/etc/passwd`	system password file
`/etc/shadow`	system file containing roles' encrypted passwords and related information
`/etc/group`	system file containing group definitions
`/etc/user_attr`	system file containing additional role attributes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `auths(1)`, `passwd(1)`, `profiles(1)`, `roles(1)`, `users(1B)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `logins(1M)`, `roleadd(1M)`, `rolemod(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `passwd(4)`, `prof_attr(4)`, `user_attr(4)`, `attributes(5)`

Notes The `roledel` utility only deletes an account definition that is in the local `/etc/group`, `/etc/passwd`, `/etc/shadow`, and `/etc/user_attr` file. file. If a network name service such as NIS or NIS+ is being used to supplement the local `/etc/passwd` file with additional entries, `roledel` cannot change information supplied by the network name service.

Name rolemod – modify a role's login information on the system

Synopsis

```
rolemod [-u uid [-o]] [-g group] [-G group [, group...]]
     [-d dir [-m]] [-s shell] [-c comment] [-l new_name]
     [-f inactive] [-e expire]
     [-A authorization [, authorization]]
     [-P profile [, profile]] [-K key=value] role
```

Description The `rolemod` utility modifies a role's login information on the system. It changes the definition of the specified login and makes the appropriate login-related system file and file system changes.

The system file entries created with this command have a limit of 512 characters per line. Specifying long arguments to several options may exceed this limit.

Options The following options are supported:

`-A` *authorization*
One or more comma separated authorizations as deined in `auth_attr(4)`. Only role with `grant` rights to the `authorization` can assign it to an account. This replaces any existing authorization setting. If no authorization list is specified, the existing setting is removed.

`-c` *comment*
Specify a comment string. *comment* can be any text string. It is generally a short description of the login, and is currently used as the field for the user's full name. This information is stored in the user's `/etc/passwd` entry.

`-d` *dir*
Specify the new home directory of the role. It defaults to *base_dir/login*, where *base_dir* is the base directory for new login home directories, and `login` is the new login.

`-e` *expire*
Specify the expiration date for a role. After this date, no role will be able to access this login. The expire option argument is a date entered using one of the date formats included in the template file `/etc/datemsk`. See `getdate(3C)`.

For example, you may enter `10/6/90` or `October 6, 1990`. A value of "`''` defeats the status of the expired date.

`-f` *inactive*
Specify the maximum number of days allowed between uses of a login ID before that login ID is declared invalid. Normal values are positive integers. A value of `0` defeats the status.

`-g` *group*
Specify an existing group's integer ID or character-string name. It redefines the role's primary group membership.

-G *group*
Specify an existing group's integer ID or character string name. It redefines the role's supplementary group membership. Duplicates between *group* with the -g and -G options are ignored. No more than NGROUPS_UMAX groups may be specified as defined in <param.h>.

-K *key=value*
Replace existing or add to a role's *key=value* pair attributes. Multiple -K options can be used to replace or add multiple *key=value* pairs. However, keys must not be repeated. The generic -K option with the appropriate key may be used instead of the specific implied key options (-A and -P). See user_attr(4) for a list of valid *key=value* pairs.

The keyword type can be specified with the value role or the value normal. When using the value normal, the account changes from a role user to a normal user; using the value role keeps the account a role user.

-l *new_logname*
Specify the new login name for the role. The *new_logname* argument is a string no more than eight bytes consisting of characters from the set of alphabetic characters, numeric characters, period (.), underline (_), and hypen (–). The first character should be alphabetic and the field should contain at least one lower case alphabetic character. A warning message will be written if these restrictions are not met. A future Solaris release may refuse to accept login fields that do not meet these requirements. The *new_logname* argument must contain at least one character and must not contain a colon (:) or NEWLINE (\n).

-m
Move the role's home directory to the new directory specified with the -d option. If the directory already exists, it must have permissions read/write/execute by *group*, where *group* is the role's primary group.

-o
This option allows the specified UID to be duplicated (non-unique).

-P *profile*
One or more comma-separated execution profiles defined in auth_attr(4). This replaces any existing profile setting. If no profile list is specified, the existing setting is removed.

-s *shell*
Specify the full pathname of the program that is used as the role's shell on login. The value of *shell* must be a valid executable file.

-u *uid*
Specify a new UID for the role. It must be a non-negative decimal integer less than MAXUID as defined in <param.h>. The UID associated with the role's home directory is not modified with this option; a role will not have access to their home directory until the UID is manually reassigned using chown(1).

Operands The following operands are supported:

`login`
An existing login name to be modified.

Exit Status In case of an error, `rolemod` prints an error message and exits with one of the following values:

`2`
The command syntax was invalid. A usage message for the `rolemod` command is displayed.

`3`
An invalid argument was provided to an option.

`4`
The *uid* given with the `-u` option is already in use.

`5`
The password files contain an error. `pwconv(1M)` can be used to correct possible errors. See `passwd(4)`.

`6`
The login to be modified does not exist, the *group* does not exist, or the login shell does not exist.

`8`
The login to be modified is in use.

`9`
The *new_logname* is already in use.

`10`
Cannot update the `/etc/group` or `/etc/user_attr` file. Other update requests will be implemented.

`11`
Insufficient space to move the home directory (`-m` option). Other update requests will be implemented.

`12`
Unable to complete the move of the home directory to the new home directory.

Files `/etc/group`
system file containing group definitions

`/etc/datemsk`
system file of date formats

`/etc/passwd`
system password file

`/etc/shadow`
system file containing users' and roles' encrypted passwords and related information

`/etc/user_attr`
system file containing additional user and role attributes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also `chown(1)`, `passwd(1)`, `users(1B)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `logins(1M)`, `pwconv(1M)`, `roleadd(1M)`, `roledel(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `getdate(3C)`, `auth_attr(4)`, `passwd(4)`, `attributes(5)`

Name root_archive – manage bootable miniroot archives

Synopsis

```
/boot/solaris/bin/root_archive pack archive root

/boot/solaris/bin/root_archive unpack archive root

/boot/solaris/bin/root_archive packmedia solaris_image root

/boot/solaris/bin/root_archive unpackmedia solaris_image root
```

Description The `root_archive` utility is used to manipulate boot archives and the bootable miniroot(s) in a Solaris install image. The utility can pack and unpack boot archives and image miniroots. Both `ufs` and `hsfs` (`iso9660`) format archives can be unpacked, although only `ufs` format is generated when packing.

For normal, boot-related system administration, `bootadm(1M)` is recommended. `root_archive`'s primary purpose is to enable OEMs to add or update a driver or other component on the Solaris install media.

A miniroot and a boot archive is closely associated with the release it is intended to boot. To ensure that the tools and system services used to construct the miniroot match, miniroot manipulation must be performed only on a system running the same release for which the miniroot is intended to install.

Subcommands The `root_archive` command has the following subcommands:

`pack` *archive root* Pack the contents of the root directory into the boot archive *archive*.

`unpack` *archive root* Unpack the contents of the boot archive named *archive* to the directory named *root*.

`packmedia` *solaris_image root* Create and pack the miniroot(s) in *solaris_image* from the contents of the directory named *root*.

`unpackmedia` *solaris_image root* Unpack the contents of the miniroot(s) in *solaris_image* to the directory named *root*.

The contents of a miniroot are constructed to need the requirements of the release. When unpacking a miniroot, all the contents of the miniroot(s) are unpacked. When packing a miniroot, the source directory must contain all the necessary components with which to construct a miniroot. In general, this can only be achieved by first unpacking an existing miniroot.

Examples **EXAMPLE 1** Unpacking the Miniroots in a Solaris x86 Install Image

The following command unpacks the miniroots in a Solaris image to the root directory in `/export/release/latest`.

```
# root_archive unpackmedia \
/export/nv/solarisdvd.nvx_dvd/latest /export/release/latest/root
```

EXAMPLE 1 Unpacking the Miniroots in a Solaris x86 Install Image *(Continued)*

In the preceding, `/export/nv/solarisdvd.nvx_dvd/latest` represents a path to a Solaris x86 install image and `/export/release/latest/root` is a directory that will be purged or created, as necessary.

EXAMPLE 2 Packing the Miniroots in a Solaris x86 Install Image

The following command creates and packs the miniroot(s) in a Solaris image from the contents of the directory `/export/release/latest/root`.

```
# root_archive packmedia \
/export/nv/solarisdvd.nvx_dvd/latest /export/release/latest/root
```

Exit Status

The following exit values are returned:

`0` The command completed successfully.

`1` The command exited due to an error.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs
Interface Stability	Committed

See Also

`cpio(1)`, `bootadm(1M)`, `mount(1M)`, `attributes(5)`, `lofi(7D)`

Name route – manually manipulate the routing tables

Synopsis

```
route [-fnvq] sub-command [ [modifiers] args]

route [-fnvq] [-p [-R root-dir]] add | delete [modifiers] destination gateway
     [args]

route [-fnvq] change | get [modifiers] destination
     [gateway [args]]

route [-fn] monitor [modifiers]

route [-fnvq] flush [modifiers]

route -p [-R root-dir] show
```

Description `route` manually manipulates the network routing tables. These tables are normally maintained by the system routing daemon, such as `in.routed(1M)` and `in.ripngd(1M)`.

`route` supports a limited number of general options, but a rich command language. Users can specify an arbitrary request that can be delivered by means of the programmatic interface discussed in `route(7P)`.

`route` uses a routing socket and the new message types `RTM_ADD`, `RTM_DELETE`, `RTM_GET`, and `RTM_CHANGE`. While only superusers can modify routing tables, the `RTM_GET` operation is allowed for non-privileged users.

Options

`-f` Flush the routing tables of all gateway entries. If you use the `-f` option in conjunction with any of the `route` sub-commands, `route` flushes the gateways before performing the sub-command. Specify the table to flush by placing the `inet` or `inet6` modifier immediately after the `-f` option. If unspecified, flushing IPv4 (`inet`) routes is the default.

`-n` Prevent attempts to print host and network names symbolically when reporting actions. This option is useful when name servers are unavailable.

`-p` Make changes to the network route tables persistent across system restarts. The operation is applied to the network routing tables first and, if successful, is then applied to the list of saved routes used at system startup. In determining whether an operation was successful, a failure to add a route that already exists or to delete a route that is not in the routing table is ignored. Particular care should be taken when using host or network names in persistent routes, as network-based name resolution services are not available at the time routes are added at startup.

`-q` Suppress all output.

`-R` *root-dir* Specify an alternate root directory where `route` applies changes. This option is ignored unless used in conjunction with the `-p` option. When `-R` is specified, route changes are applied only to the list of saved routes to be used at startup, *not* to the network routing tables. In addition, certain checks, such as the

existance of network interfaces used with `-ifp`, are skipped. This can be useful from within JumpStart scripts, where the root directory of the system being modified is in a location other than `/`.

`-v` Print additional details in verbose mode.

Subcommands

The following subcommands are supported:

`add` Add a route.

`change` Change aspects of a route (such as its gateway).

`delete` Delete a specific route.

`flush` Remove all gateway entries from the routing table.

`get` Look up and display the route for a destination.

`monitor` Continuously report any changes to the routing information base, routing lookup misses, or suspected network partitionings.

`show` Display the list of routes to be applied at system startup. Can be used only in conjunction with the `-p` option.

The `add` and `delete` sub-commands have the following syntax:

```
route [ -fnvq ] cmd destination gateway [metric/netmask]
```

where *cmd* is `add` or `delete`, *destination* is the destination host or network, and *gateway* is the next-hop intermediary through which packets should be routed. Modifiers described in `OPERANDS` can be placed anywhere on the command line.

The `get` and `change` sub-commands have the following syntax:

```
route [ -fnvq ] cmd destination [gateway [metric/netmask]]
```

where *cmd* is `get` or `change`, *destination* is the destination host or network, and *gateway* is the next-hop intermediary through which packets should be routed. Modifiers described in `OPERANDS` can be placed anywhere on the command line.

The `monitor` sub-command has the following syntax:

```
route monitor [ -inet | -inet6 ]
```

Operands

`route` executes its sub-commands on routes to destinations by way of gateways.

Destinations and Gateways

By default, destination and gateway addresses are interpreted as IPv4 addresses. All symbolic names are tried first as a host name, using `getipnodebyname(3SOCKET)`. If this lookup fails in the AF_INET case, `getnetbyname(3SOCKET)` interprets the name as that of a network.

Including an optional modifier on the command line before the address changes how the `route` sub-command interprets it.

The following modifiers are supported:

`-inet` Force the address to be interpreted as an IPv4 address, that is, under the `AF_INET` address family.

`-inet6` Force the address to be interpreted as an IPv6 address, that is, under the `AF_INET6` address family.

For IPv4 addresses, routes to a particular host are by default distinguished from those to a network by interpreting the Internet address specified as the destination. If the destination has a *local address part* (that is, the portion not covered by the netmask) of `0`, or if the destination is resolved as the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host.

You can force this selection by using one of the following modifiers:

`-host` Force the destination to be interpreted as a host.

`-net` Force the destination to be interpreted as a network.

For example:

Destination	Destination Equivalent
`128.32`	`-host 128.0.0.32`
`128.32.130`	`-host 128.32.0.130`
`-net 128.32`	`128.32.0.0`
`-net 128.32.130`	`128.32.130.0`

Two modifiers avoid confusion between addresses and keywords (for example., `host` used as a symbolic host name). You can distinguish a *destination* by preceding it with the `-dst` modifier. You can distinguish a gateway address by using the `-gateway` modifier. If the destination is directly reachable by way of an interface requiring no intermediary IP router to act as a gateway, this can be indicated by using the `-interface` or `-iface` modifier.

In the following example, the route does not refer to an external gateway (router), but rather to one of the machine's interfaces. Packets with IP destination addresses matching the destination and mask on such a route are sent out on the interface identified by the gateway address. For interfaces using the ARP protocol, this type of route is used to specify that all matching destinations are local to the physical link. That is, a host could be configured to ARP for all addresses, without regard to the configured interface netmask, by adding a default route using this command. For example:

```
example# route add default hostname -interface
```

where gateway address *hostname* is the name or IP address associated with the network interface over which all matching packets should be sent. On a host with a single network interface, *hostname* is usually the same as the *nodename* returned by the `uname -n` command. See `uname(1)`.

For backward compatibility with older systems, directly reachable routes can also be specified by placing a `0` after the gateway address:

```
example# route add default hostname 0
```

This value was once a route metric, but this metric is no longer used. If the value is specified as `0`, then the destination is directly reachable (equivalent to specifying `-interface`). If it is non-zero but cannot be interpreted as a subnet mask, then a gateway is used (default).

With the `AF_INET` address family or an IPv4 address, a separate subnet mask can be specified. This can be specified in one of the following ways:

- IP address following the gateway address . This is typically specified in *decimal dot* notation as for `inet_addr(3SOCKET)` rather than in symbollic form.
- IP address following the `-netmask` qualifier.
- Slash character and a decimal length appended to the destination address.

If a subnet mask is not specified, the mask used is the subnet mask of the output interface selected by the gateway address, if the classful network of the destination is the same as the classful network of the interface. Otherwise, the classful network mask for the destination address is used.

Each of the following examples creates an IPv4 route to the destination `192.0.2.32` subnet with a subnet mask of `255.255.255.224`:

```
example# route add 192.0.2.32/27 somegateway
example# route add 192.0.2.32 -netmask 255.255.255.224 somegateway
example# route add 192.0.2.32 somegateway 255.255.255.224
```

For IPv6, only the slash format is accepted. The following example creates an IPv6 route to the destination `33fe::` with a netmask of `16` one-bits followed by 112 zero-bits.

```
example# route add -inet6 3ffe::/16 somegateway
```

In cases where the gateway does not uniquely identify the output interface (for example, when several interfaces have the same address), you can use the `-ifp` *ifname* modifier to specify the interface by name. For example, `-ifp lo0` associates the route with the `lo0` interface.

Routing Flags

Routes have associated flags that influence operation of the protocols when sending to destinations matched by the routes. These flags can be set (and in some cases cleared, indicated by ~) by including the following modifiers on the command line:

Modifier	Flag	Description
-interface	~RTF_GATEWAY	Destination is directly reachable
-iface	~RTF_GATEWAY	Alias for interface modifier
-static	RTF_STATIC	Manually added route
-nostatic	~RTF_STATIC	Pretend route was added by kernel or routing daemon
-reject	RTF_REJECT	Emit an ICMP unreachable when matched
-blackhole	RTF_BLACKHOLE	Silently discard packets duing updates
-proto1	RTF_PROTO1	Set protocol specific routing flag #1
-proto2	RTF_PROTO2	Set protocol specific routing flag #2
-private	RTF_PRIVATE	Do not advertise this route
-multirt	RTF_MULTIRT	Creates the specified redundant route
-setsrc	RTF_SETSRC	Assigns the default source address

The optional modifiers `-rtt`, `-rttvar`, `-sendpipe`, `-recvpipe`, `-mtu`, `-hopcount`, `-expire`, and `-ssthresh` provide initial values to quantities maintained in the routing entry by transport level protocols, such as TCP. These can be individually locked either by preceding each modifier to be locked by the `-lock` meta-modifier, or by specifying that all ensuing metrics can be locked by the `-lockrest` meta-modifier.

Some transport layer protocols can support only some of these metrics. The following optional modifiers are supported:

`-expire` Lifetime for the entry. This optional modifier is not currently supported.

`-hopcount` Maximum hop count. This optional modifier is not currently supported.

`-mtu` Maximum MTU in bytes.

`-recvpipe` Receive pipe size in bytes.

`-rtt` Round trip time in microseconds.

`-rttvar` Round trip time variance in microseconds.

`-sendpipe` Send pipe size in bytes.

`-ssthresh` Send pipe size threshold in bytes.

`-secattr` Security attributes of the route. This modifier is available only if the system is configured with the Solaris Trusted Extensions feature.

The `-secattr` modifier has the following format:

`min_sl=`*val*`,max_sl=`*val*`,doi=`*val*`,cipso`

or:

`sl=VAL,doi=VAL,cipso`

In the first form, above, the *val* for `min_sl` and `max_sl` is a sensitivity label in either hex or string form. The *val* for `doi` is a non-negative integer. The route will apply only for packets with the same domain of interpretation as defined by the `doi` value and within the accreditation range defined by the `min_sl` and `max_sl` values. The `cipso` keyword is optional and set by default. Valid `min_sl`, `max_sl` and `doi` keyword/value pairs are mandatory. Note that if *val* contains a space, it must be protected by double quotes.

The second form, above, is equivalent to specifying the first form with the same `VAL` for `min_sl` and `max_sl`. The second form should be used for the `get` command, because `get` uses only a single sensitivity label.

Compatibility The modifiers `host` and `net` are taken to be equivalent to `-host` and `-net`. To specify a symbolic address that matches one of these names, use the `dst` or `gateway` keyword to distinguish it. For example: `-dst host`

The following two flags are also accepted for compatibility with older systems, but have no effect.

Modifier	Flag
`-cloning`	RTF_CLONING
`-xresolve`	RTF_XRESOLVE

The `-ifa hostname` modifier is also accepted, but has no effect.

Files

`/etc/defaultrouter`	List of default routers
`/etc/hosts`	List of host names and net addresses
`/etc/networks`	List of network names and addresses

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `uname(1)`, `in.ripngd(1M)`, `in.routed(1M)`, `netstat(1M)`, `routed(1M)`, `ioctl(2)`, `getipnodebyname(3SOCKET)`, `getnetbyname(3SOCKET)`, `inet_addr(3SOCKET)`, `defaultrouter(4)`, `hosts(4)`, `networks(4)`, `attributes(5)`, `arp(7P)`, `ip(7P)`, `route(7P)`, `routing(7P)`

Diagnostics

`add` [`host` | `network`] *destination*:*gateway* *flags*	The specified route is being added to the tables. The values printed are from the routing table entry supplied in the `ioctl(2)` call. If the gateway address used was not the primary address of the gateway (the first one returned by `getipnodebyname(3SOCKET)`) the gateway address is printed numerically as well as symbolically.
`delete` [`host` | `network`] *destination*:*gateway* *flags* `change` [`host` | `network`] *destination*:*gateway* *flags*	As `add`, but when deleting or changing an entry.
destination `done`	When the `-f` flag is specified, or the `flush` sub-command is used, each routing table entry deleted is indicated with a message of this form.
`Network is unreachable`	An attempt to add a route failed because the gateway listed was not on a directly-connected network. Give the next-hop gateway instead.
`not in table`	A `delete` operation was attempted for an entry that is not in the table.
`entry exists`	An `add` operation was attempted for a route that already exists in the kernel.
`routing table overflow`	An operation was attempted, but the system was unable to allocate memory to create the new entry.
`insufficient privileges`	An attempt to add, delete, change, or flush a route failed because the calling process does not have appropriate privileges.

Notes Specifying that destinations are local (with the `-interface`modifier) assumes that the routers implement `proxy ARP`, meaning that they respond to ARP queries for all reachable destinations. Normally, using either router discovery or RIP is more reliable and scalable than using proxy ARP. See `in.routed(1M)` for information related to RIP.

Combining the all destinations are local route with subnet or network routes can lead to unpredictable results. The search order as it relates to the all destinations are local route are undefined and can vary from release to release.

Name routeadm – IP forwarding and routing configuration

Synopsis

```
routeadm [-p [option]]

routeadm [-R root-dir] [-e option ...] [-d option...]
 [-r option...] [-s var=value]

routeadm [-l fmri]

routeadm [-m fmri key=value [key=value]...]

routeadm [-u]
```

Description The `routeadm` command is used to administer system-wide configuration for IP forwarding and routing. IP forwarding is the passing of IP packets from one network to another; IP routing is the use of a routing protocol to determine routes.

IP forwarding and routing functions are also represented as services within the service management facility (SMF), and can be administered by means of `svcadm(1M)` also, using the following fault management resource identifiers (FMRIs):

```
svc:/network/ipv4-forwarding:default
svc:/network/ipv6-forwarding:default
svc:/network/routing/route:default
svc:/network/routing/ripng:default
```

See EXAMPLES for relevant examples.

In addition to enabling and disabling routing and forwarding, `routeadm` is used to interact with SMF-based routing daemon services. Routing daemon services are identified by the presence of a `routeadm` application property group, which `routeadm` uses in administering the given service. Routing daemon services can also specify properties relating to their operation in the `routing` application property group; these can be modified by means of routeadm -m. If an FMRI for a service without such a property group is specified, an error is issued and the operation is not carried out. If a routing daemon has not been converted to SMF, the `ipv4`*`[or 6]`*`-routing-daemon`, `ipv4`*`[or 6]`*`-routing-daemon-args`, and `ipv4`*`[or 6]`*`-routing-stop-cmd` variables can be used to specify the appropriate daemon for IPv4 or IPv6 routing. `routeadm` will then run that daemon using the `svc:/network/routing/legacy-routing:ipv4`*`[or 6]`* service as appropriate. This conversion process occurs when you issue an enable (`-e`), disable (`-d`) or an update (`-u`) command.

The first usage, in the SYNOPSIS above, reports the current configuration.

Options The following command-line options are supported:

`-p` [*option*]
: Print the configuration in parseable format. If *option* is specified, only the configuration for the specified option or variable is displayed.

`-R` *root-dir*
Specify an alternate root directory where `routeadm` applies changes. This can be useful from within JumpStart scripts, where the root directory of the system being modified is mounted elsewhere.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-e` *option...*
Enable the specified option. The effect is to prepare the associated services (`svc:/network/ipv4-forwarding:default` in the case of `ipv4-forwarding`) for enabling. By means of the `routing-svcs` variable, the routing daemons are specified to be enabled on subsequent boot or when `routeadm -u` is run.

`-d` *option...*
Disable the specified option. The effect is to prepare the associated services (`svc:/network/ipv4-forwarding:default` in the case of `ipv4-forwarding`) for enabling. By means of the `routing-svcs` variable, the routing daemons are specified to be disabled on subsequent boot or when `routeadm -u` is run.

`-l` *fmri*
List all properties in the routing application property group for the SMF routing daemon service.

`-m` *fmri key=value*
Change property value of property *key* to *value* in routing application property group for the SMF routing daemon service. For multi-valued properties, the property name can be used multiple times in the modify operation, and each associated value will be added.

`-r` *option...*
Revert the specified option to the system default. The system defaults are specified in the description of each *option*.

`-u`
Apply the currently configured options to the running system. These options might include enabling or disabling IP forwarding and launching or killing routing daemons, if any are specified. It does not alter the state of the system for those settings that have been set to default. This option is meant to be used by administrators who do not want to reboot to apply their changes. In addition, this option upgrades non-SMF configurations from the invocations of daemon stop commands, which might include a set of arguments, to a simple enabling of the appropriate service.

`-s` *key=value*
Specify string values for specific variables in a comma-separated list with no intervening spaces. If invalid options are specified, a warning message is displayed and the program exits. The following variables can be specified:

`routing-svcs=`*fmrilist*
Specifies the routing daemon services to be enabled. Routing daemon services are determined to be IPv4 or IPv6 (and so enabled or disabled when `routeadm -e/-d ipv4`*(6)*`-routing` is run) on the basis of property values in the `routeadm` application property group. Default: `route:default ripng:default`

`ipv4-routing-daemon=`<*full_path_to_routing_daemon*>
Specifies the routing daemon to be started when `ipv4-routing` is enabled. The routing daemon specified must be an executable binary or shell-script. If the specified program maps to an SMF service, the service will be used, and daemon arguments to the program will be transferred to the properties of the service at enable time. Default: ""

`ipv4-routing-daemon-args=`<*args*>
Specifies the startup arguments to be passed to the `ipv4-routing-daemon` when `ipv4-routing` is enabled. Default: no arguments

`ipv4-routing-stop-cmd=`<*command*>
Specifies the command to be executed to stop the routing daemon when `ipv4-routing` is disabled. <*command*> can be an executable binary or shell-script, or a string that can be parsed by `system`(3C). Default: ""

`ipv6-routing-daemon=`<*full_path_to_routing_daemon*>
Specifies the routing daemon to be started when `ipv6-routing` is enabled. The routing daemon specified must be an executable binary or shell-script. If the specified program maps to an SMF service, the service will be used, and daemon arguments to the program will be transferred to the properties of the service at enable time. Default: ""

`ipv6-routing-daemon-args=`<*args*>
Specifies the startup arguments to be passed to the `ipv6-routing-daemon` when `ipv6-routing` is enabled. Default: ""

`ipv6-routing-stop-cmd=`<*command*>
Specifies the command to be executed to stop the routing daemon when `ipv6-routing` is disabled. <*command*> can be an executable binary or shell-script, or a string that can be parsed by `system`(3C). Default: ""

Multiple `-e`, `-d`, and `-r` options can be specified on the command line. Changes made by `-e`, `-d`, and `-r` are persistent, but are not applied to the running system unless `routeadm` is called later with the `-u` option.

Use the following options as arguments to the `-e`, `-d`, and `-r` options (shown above as *option...*).

`ipv4-forwarding`
Controls the global forwarding configuration for all IPv4 interfaces. The system default is `disabled`. If enabled, IP will forward IPv4 packets to and from interfaces when appropriate. If disabled, IP will not forward IPv4 packets to and from interfaces when appropriate. The SMF service associated with this configuration variable is

svc:/network/routing/ipv4-forwarding. This service will be enabled or disabled as appropriate when routeadm is called with the u option. As an alternative, you can use svcadm(1M). Services that require ipv4-forwarding to be enabled should specify a dependency on this service.

ipv4-routing
Determines whether an IPv4 routing daemon is run. The system default is enabled unless the /etc/defaultrouter file exists (see defaultrouter(4)), in which case the default is disabled. The value of this option reflects the state of all IPv4 routing services, such that if any IPv4 routing service is enabled, ipv4-routing is enabled. This allows users to interact with routing services using svcadm(1M), as well as through routeadm. IPv4 routing services, specified by means of the routing-svcs variable, will be prepared for enable on next boot when the user explicitly enables ipv4-routing. The SMF routing daemon service for in.routed (svc:/network/routing/route:default) is specified by default.

ipv6-forwarding
Controls the global forwarding configuration for all IPv6 interfaces. The system default is disabled. If enabled, IP will forward IPv6 packets to and from interfaces when appropriate. If disabled, IP will not forward IPv6 packets to and from interfaces when appropriate. The SMF service associated with this configuration variable is svc:/network/routing/ipv6-forwarding. This service will be enabled or disabled as appropriate when routeadm is called with the -u option, or svcadm(1M) is used. Services that require ipv6-forwarding to be enabled should specify a dependency on this service.

ipv6-routing
Determines whether an IPv6 routing daemon is run. The system default is disabled. The value of this option reflects the state of all IPv6 routing services, such that, if any IPv6 routing service is enabled, ipv6-routing is enabled. This allows users to interact with routing services via svcadm(1M) as well as through routeadm. IPv6 routing services, specified by means of the routing-svcs variable, will be prepared for enable on next boot when the user explicitly enables ipv6-routing. The SMF routing daemon service for in.ripngd (svc:/network/routing/ripng:default) is specified by default.

The forwarding and routing settings are related but not mutually dependent. For example, a router typically forwards IP packets and uses a routing protocol, but nothing would prevent an administrator from configuring a router that forwards packets and does not use a routing protocol. In that case, the administrator would enable forwarding, disable routing, and populate the router's routing table with static routes.

The forwarding settings are global settings. Each interface also has an IFF_ROUTER forwarding flag that determines whether packets can be forwarded to or from a particular interface. That flag can be independently controlled by means of ifconfig(1M)'s router option. When the global forwarding setting is changed (that is, -u is issued to change the value from enabled to disabled or vice-versa), all interface flags in the system are changed simultaneously to reflect the new global policy. Interfaces configured by means of DHCP automatically have their interface-specific IFF_ROUTER flag cleared.

When a new interface is plumbed by means of `ifconfig`, the value of the interface-specific forwarding flag is set according to the current global forwarding value. Thus, the forwarding value forms the "default" for all new interfaces.

Examples

EXAMPLE 1 Enabling IPv4 Forwarding

IPv4 forwarding is disabled by default. The following command enables IPv4 forwarding:

```
example# routeadm -e ipv4-forwarding
```

EXAMPLE 2 Apply Configured Settings to the Running System

In the previous example, a system setting was changed, but will not take effect until the next reboot unless a command such as the following is used:

```
example# routeadm -u
```

An alternative to the above two steps is to simply enable the equivalent SMF service:

```
example# svcadm enable svc:/network/ipv4-forwarding
```

...or, using the abbreviated FMRI:

```
example# svcadm enable ipv4-forwarding
```

EXAMPLE 3 Making a Setting Revert to its Default

To make the setting changed in the first example revert to its default, enter the following:

```
example# routeadm -r ipv4-forwarding
example# routeadm -u
```

EXAMPLE 4 Starting `in.routed` with the `-q` Flag

Setting the `-q` flag is represented in the SMF service by setting the `quiet_mode` property to true. The following sequence of commands starts `in.routed` with the `-q` flag:

```
example# routeadm -m route:default quiet_mode=true
example# routeadm -e ipv4-routing -u
```

See `in.routed(1M)` for details of property names and how they relate to daemon behavior.

Exit Status

The following exit values are returned:

`0` Successful completion.

`!=0` An error occurred while obtaining or modifying the system configuration.

Files

`/etc/inet/routing.conf` Parameters for IP forwarding and routing. (Not to be edited.)

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Stable

See Also ifconfig(1M), in.routed(1M), svcadm(1M), gateways(4), attributes(5), smf(5)

Name rpcbind – universal addresses to RPC program number mapper

Synopsis

```
rpcbind [-d] [-w]
```

Description `rpcbind` is a server that converts RPC program numbers into universal addresses. It must be running on the host to be able to make RPC calls on a server on that machine.

When an RPC service is started, it tells `rpcbind` the address at which it is listening, and the RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it first contacts `rpcbind` on the server machine to determine the address where RPC requests should be sent.

`rpcbind` should be started before any other RPC service. Normally, standard RPC servers are started by port monitors, so `rpcbind` must be started before port monitors are invoked.

When `rpcbind` is started, it checks that certain name-to-address translation-calls function correctly. If they fail, the network configuration databases can be corrupt. Since RPC services cannot function correctly in this situation, `rpcbind` reports the condition and terminates.

`rpcbind` maintains an open transport end for each transport that it uses for indirect calls. This is the UDP port on most systems.

The `rpcbind` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/bind
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. `rpcbind` can only be started by the superuser or someone in the Primary Administrator role.

The configuration properties of this service can be modified with `svccfg(1M)`.

The following SMF property is used to allow or disallow access to `rpcbind` by remote clients:

```
config/local_only = true
```

The default value, `true`, shown above, disallows remote access; a value of `false` allows remove access. See EXAMPLES.

The FMRI `svc:network/rpc/bind` property group `config` contains the following property settings:

`enable_tcpwrappers` Specifies that the TCP wrappers facility is used to control access to TCP services. The value `true` enables checking. The default value for `enable_tcpwrappers` is `false`. If the `enable_tcpwrappers` parameter is enabled, then all calls to `rpcbind` originating from non-local addresses are automatically wrapped by the TCP wrappers facility. The `syslog` facility code daemon is used to log

allowed connections (using the `info` severity level) and denied traffic (using the `warning` severity level). See `syslog.conf(4)` for a description of `syslog` codes and severity levels. The stability level of the TCP wrappers facility and its configuration files is External. As the TCP wrappers facility is not controlled by Sun, intrarelease incompatibilities are not uncommon. See `attributes(5)`.

`verbose_logging` Specifies whether the TCP wrappers facility logs all calls orjust the denied calls. The default is `false`. This option has no effect if TCP wrappers are not enabled.

`allow_indirect` Specifies whether `rpcbind` allows indirect calls at all. By default, `rpcbind` allows most indirect calls, except to a number of standard services(`keyserv`, `automount`, `mount`, `nfs`, `rquota`, and selected NIS and `rpcbind` procedures). Setting `allow_indirect` to `false` causes all indirect calls to be dropped. The default is `true`. NIS broadcast clients rely on this functionality on NIS servers.

Options

The following options are supported:

`-d` Run in debug mode. In this mode, `rpcbind` does not fork when it starts. It prints additional information during operation, and aborts on certain errors. With this option, the name-to-address translation consistency checks are shown in detail.

`-w` Do a warm start. If `rpcbind` aborts or terminates on `SIGINT` or `SIGTERM`, it writes the current list of registered services to `/var/run/portmap.file` and `/var/run/rpcbind.file`. Starting `rpcbind` with the `-w` option instructs it to look for these files and start operation with the registrations found in them. This allows `rpcbind` to resume operation without requiring all RPC services to be restarted.

Examples

EXAMPLE 1 Allowing Remote Access

The following sequence of commands allows remote access to `rpcbind`.

```
# svccfg -s svc:/network/rpc/bind setprop config/local_only = false
# svcadm refresh svc:/network/rpc/bind
```

Files

`/var/run/portmap.file` Stores the information for RPC services registered over IP based transports for warm start purposes.

`/var/run/rpcbind.file` Stores the information for all registered RPC services for warm start purposes.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	See below.

TCP wrappers is External.

See Also smf(5), rpcinfo(1M), svcadm(1M), svccfg(1M), rpcbind(3NSL), syslog.conf(4), attributes(5), smf(5)

For information on the TCP wrappers facility, see the `hosts_access(4)` man page, delivered as part of the Solaris operating environment in `/usr/sfw/man` and available in the `SUNWtcpd` package.

Notes Terminating `rpcbind` with `SIGKILL` prevents the warm-start files from being written.

All RPC servers are restarted if the following occurs: `rpcbind` crashes (or is killed with `SIGKILL`) and is unable to to write the warm-start files; `rpcbind` is started without the `-w` option after a graceful termination. Otherwise, the warm start files are not found by `rpcbind`.

Name rpc.bootparamd, bootparamd – boot parameter server

Synopsis

```
/usr/sbin/rpc.bootparamd [-d]
```

Description `rpc.bootparamd` is a server process that provides information from a `bootparams` database to diskless clients at boot time. See `bootparams(4)`

The source for the `bootparams` database is determined by the `nsswitch.conf(4)` file (on the machine running the `rpc.bootparamd` process).

Options The following options are supported:

`-d` Display debugging information.

Files

`/etc/bootparams`	boot parameter data base
`/etc/nsswitch.conf`	configuration file for the name-service switch

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbsu

See Also `svcs(1)`,`svcadm(1M)`, `bootparams(4)`, `nsswitch.conf(4)`, `attributes(5)`, `smf(5)`

Notes A diskless client requires service from at least one `rpc.bootparamd` process running on a server that is on the same IP subnetwork as the diskless client.

Some routines that compare hostnames use case-sensitive string comparisons; some do not. If an incoming request fails, verify that the case of the hostname in the file to be parsed matches the case of the hostname called for, and attempt the request again.

The `rpc.bootparamd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/bootparams
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name rpcinfo – report RPC information

Synopsis

```
rpcinfo [-m | -s] [host]

rpcinfo -p [host]

rpcinfo -T transport host prognum [versnum]

rpcinfo -l [-T transport] host prognum versnum

rpcinfo [-n portnum] -u host prognum [versnum]

rpcinfo [-n portnum] -t host prognum [versnum]

rpcinfo -a serv_address -T transport prognum [versnum]

rpcinfo -b [-T transport] prognum versnum

rpcinfo -d [-T transport] prognum versnum
```

Description `rpcinfo` makes an RPC call to an RPC server and reports what it finds.

In the first synopsis, `rpcinfo` lists all the registered RPC services with `rpcbind` on *host*. If *host* is not specified, the local host is the default. If `-s` is used, the information is displayed in a concise format.

In the second synopsis, `rpcinfo` lists all the RPC services registered with `rpcbind`, version 2. Note that the format of the information is different in the first and the second synopsis. This is because the second synopsis is an older protocol used to collect the information displayed (version 2 of the `rpcbind` protocol).

The third synopsis makes an RPC call to procedure 0 of *prognum* and *versnum* on the specified *host* and reports whether a response was received. *transport* is the transport which has to be used for contacting the given service. The remote address of the service is obtained by making a call to the remote `rpcbind`.

The *prognum* argument is a number that represents an RPC program number (see `rpc(4)`).

If a *versnum* is specified, `rpcinfo` attempts to call that version of the specified *prognum*. Otherwise, `rpcinfo` attempts to find all the registered version numbers for the specified *prognum* by calling version 0, which is presumed not to exist; if it does exist, `rpcinfo` attempts to obtain this information by calling an extremely high version number instead, and attempts to call each registered version. Note that the version number is required for `-b` and `-d` options.

The `EXAMPLES` section describe other ways of using `rpcinfo`.

Options

`-T` *transport* — Specify the transport on which the service is required. If this option is not specified, `rpcinfo` uses the transport specified in the `NETPATH` environment variable, or if that is unset or `NULL`, the transport in the `netconfig(4)` database is used. This is a generic option, and can be used in conjunction with other options as shown in the `SYNOPSIS`.

`-a` *serv_address* Use *serv_address* as the (universal) address for the service on *transport* to `ping` procedure 0 of the specified *prognum* and report whether a response was received. The `-T` option is required with the `-a` option. If *versnum* is not specified, `rpcinfo` tries to ping all available version numbers for that program number. This option avoids calls to remote `rpcbind` to find the address of the service. The *serv_address* is specified in universal address format of the given transport.

`-b` Make an RPC broadcast to procedure 0 of the specified *prognum* and *versnum* and report all hosts that respond. If *transport* is specified, it broadcasts its request only on the specified transport. If broadcasting is not supported by any transport, an error message is printed. Use of broadcasting should be limited because of the potential for adverse effect on other systems.

`-d` Delete registration for the RPC service of the specified *prognum* and *versnum*. If *transport* is specified, unregister the service on only that transport, otherwise unregister the service on all the transports on which it was registered. Only the owner of a service can delete a registration, except the superuser, who can delete any service.

`-l` Display a list of entries with a given *prognum* and *versnum* on the specified *host*. Entries are returned for all transports in the same protocol family as that used to contact the remote `rpcbind`.

`-m` Display a table of statistics of `rpcbind` operations on the given *host*. The table shows statistics for each version of `rpcbind` (versions 2, 3 and 4), giving the number of times each procedure was requested and successfully serviced, the number and type of remote call requests that were made, and information about RPC address lookups that were handled. This is useful for monitoring RPC activities on *host*.

`-n` *portnum* Use *portnum* as the port number for the `-t` and `-u` options instead of the port number given by `rpcbind`. Use of this option avoids a call to the remote `rpcbind` to find out the address of the service. This option is made obsolete by the `-a` option.

`-p` Probe `rpcbind` on *host* using version 2 of the `rpcbind` protocol, and display a list of all registered RPC programs. If *host* is not specified, it defaults to the local host. This option is not useful for IPv6; use `-s` (see below) instead. Note that version 2 of the `rpcbind` protocol was previously known as the portmapper protocol.

`-s` Display a concise list of all registered RPC programs on *host*. If *host* is not specified, it defaults to the local host.

`-t` Make an RPC call to procedure 0 of *prognum* on the specified *host* using TCP, and report whether a response was received. This option is made obsolete by the `-T` option as shown in the third synopsis.

`-u` Make an RPC call to procedure 0 of *prognum* on the specified *host* using UDP, and report whether a response was received. This option is made obsolete by the `-T` option as shown in the third synopsis.

Examples

EXAMPLE 1 RPC services.

To show all of the RPC services registered on the local machine use:

```
example% rpcinfo
```

To show all of the RPC services registered with `rpcbind` on the machine named `klaxon` use:

```
example% rpcinfo klaxon
```

The information displayed by the above commands can be quite lengthy. Use the `-s` option to display a more concise list:

```
example% rpcinfo -s klaxon
```

program	vrsn	netid(s)	service	owner
100000	2,3,4	tcp,udp,ticlts,ticots,ticotsord	rpcbind	superuser
100008	1	ticotsord,ticots,ticlts,udp,tcp	walld	superuser
100002	2,1	ticotsord,ticots,ticlts,udp,tcp	rusersd	superuser
100001	2,3,4	ticotsord,ticots,tcp,ticlts,udp	rstatd	superuser
100012	1	ticotsord,ticots,ticlts,udp,tcp	sprayd	superuser
100007	3	ticotsord,ticots,ticlts,udp,tcp	ypbind	superuser
100029	1	ticotsord,ticots,ticlts	keyserv	superuser
100078	4	ticotsord,ticots,ticlts	-	superuser
100024	1	ticotsord,ticots,ticlts,udp,tcp	status	superuser
100021	2,1	ticotsord,ticots,ticlts,udp,tcp	nlockmgr	superuser
100020	1	ticotsord,ticots,ticlts,udp,tcp	llockmgr	superuser

To show whether the RPC service with program number *prognum* and version *versnum* is registered on the machine named `klaxon` for the transport TCP use:

```
example% rpcinfo -T tcp klaxon prognum versnum
```

To show all RPC services registered with version 2 of the `rpcbind` protocol on the local machine use:

```
example% rpcinfo -p
```

To delete the registration for version 1 of the `walld` (program number `100008`) service for all transports use:

```
example# rpcinfo -d 100008 1
```

or

```
example# rpcinfo -d walld 1
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `rpcbind(1M)`, `rpc(3NSL)`, `netconfig(4)`, `rpc(4)`, `attributes(5)`

Name rpc.mdcommd – multi-node disk set services

Synopsis

```
/usr/sbin/rpc.mdcommd
```

Description `rpc.mdcommd` is an `rpc(4)` daemon that functions as a server process. `rpc.mdcommd` manages communication among hosts participating in a multi-node disk set configuration.

`rpc.mdcommd` is invoked by `inetd(1M)`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmfu
Stability	Evolving

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `metaset(1M)`, `svcadm(1M)`, `rpc(3NSL)`, `rpc(4)`, `services(4)`, `attributes(5)`, `smf(5)`

Solaris Volume Manager Administration Guide

Notes The `rpc.mdcommd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/mdcomm
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.metad – remote metaset services

Synopsis

```
/usr/sbin/rpc.metad
```

Description `rpc.metad` is an rpc(4) daemon (functioning as a server process) that is used to manage local copies of metadevice diskset information. The `rpc.metad` daemon is controlled by inetadm(1M).

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also svcs(1), inetadm(1M), inetd(1M), metaset(1M), rpc.metamhd(1M), svcadm(1M), rpc(3NSL), services(4), attributes(5), smf(5)

Solaris Volume Manager Administration Guide

Notes The `rpc.metad` service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/rpc/meta:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). Responsibility for initiating and restarting this service is delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

Name rpc.metamedd – remote mediator services

Synopsis

```
/usr/sbin/rpc.metamedd
```

Description `rpc.metamedd` is an `rpc(4)` server which is used to manage mediator information for use in 2–string HA configurations. The `rpc.metamedd` daemon is controlled by `inetadm(1M)`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu
Interface Stability	Evolving

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `rpc(4)`, `services(4)`, `attributes(5)`, `smf(5)`

Sun Cluster documentation, *Solaris Volume Manager Administration Guide*

Notes The `rpc.metamedd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/metamed:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.metamhd – remote multihost disk services

Synopsis

```
/usr/sbin/rpc.metamhd
```

Description `rpc.metamhd` is an `rpc(4)` daemon (functioning as a server process) that is used to manage multi-hosted disks. The `rpc.metamhd` daemon is controlled by `inetadm(1M)`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmdu

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `metaset(1M)`, `rpc.metad(1M)`, `svcadm(1M)`, `rpc(3NSL)`, `services(4)`, `attributes(5)`, `smf(5)`

Solaris Volume Manager Administration Guide

Notes The `rpc.metamhd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/metamh:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.nisd, nisd – NIS+ service daemon

Synopsis

```
/usr/sbin/rpc.nisd [-ACDFhlv] [-Y [-B [-t netid]]]
     [-d dictionary] [-L load] [-S level] [-m mappingfile]
     [-x attribute=value]... [-z number]
```

Description The `rpc.nisd` daemon is an RPC service that implements the NIS+ service. This daemon must be running on all machines that serve a portion of the NIS+ namespace.

`rpc.nisd` is usually started from a system startup script.

The `-B` option causes `rpc.nisd` to start an auxiliary process, `rpc.nisd_resolv`, which provides ypserv compatible DNS forwarding for NIS host requests. `rpc.nisd_resolv` can also be started independently. See `rpc.nisd_resolv(1M)` for more information on using `rpc.nisd_resolv` independently.

The `/etc/default/rpc.nisd` file contains the following default parameter settings. See `FILES`.

`ENABLE_NIS_YP_EMULATION`	Specifies whether the server is put into NIS (YP) compatibility mode. `ENABLE_NIS_YP_EMULATION=YES` is equivalent to the `-Y` command-line option. The default value for `ENABLE_NIS_YP_EMULATION` is `NO`.

Options

`-A`	Authentication verbose mode. The daemon logs all the authentication related activities to `syslogd(1M)` with `LOG_INFO` priority.
`-B`	Provide ypserv compatible DNS forwarding for NIS host requests. The DNS resolving process, `rpc.nisd_resolv`, is started and controlled by `rpc.nisd`. This option requires that the `/etc/resolv.conf` file be setup for communication with a DNS nameserver. The `nslookup` utility can be used to verify communication with a DNS nameserver. See `resolv.conf(4)` and `nslookup(1M)`.
`-C`	Open diagnostic channel on `/dev/console`.
`-D`	Debug mode. Do not fork.
`-d` *dictionary*	Specify an alternate dictionary for the NIS+ database. The primary use of this option is for testing. Note that the string is not interpreted, rather it is simply passed to the `db_initialize` function.>
`-F`	Force the server to do a checkpoint of the database when it starts up. Forced checkpoints may be required when the server is low on disk space. This option removes updates from the transaction log that have propagated to all of the replicas.
`-h`	Print list of options.

`-L` *number*
: Specify the "load" the NIS+ service is allowed to place on the server. The load is specified in terms of the *number* of child processes that the server may spawn. The value of *number* must be at least 1 for the callback functions to work correctly. The default is 128.

`-m` *mappingfile*
: Specify the name of a configuration file that maps NIS+ objects (especially tables and columns) to LDAP (entries and attributes). See `NIS+LDAPmapping(4)`. The default path is `/var/nis`. The default mapping file is `NIS+LDAPmapping`. If this file exists, the `rpc.nisd` daemon will map data to and from LDAP. A template mapping file that covers the normal NIS+ directories and tables is installed as `/var/nis/NIS+LDAPmapping.template`.

  A NIS+ object must have a valid mapping entry in the mapping file in order to have data for that table read from or written to the LDAP repository.

  The `rpc.nisd(4)` file contains specifications for LDAP server addresses, LDAP authentication method, and the like. See `NIS+LDAPmapping(4)` for an overview of the setup you need to map NIS+ data to or from LDAP.

`-S` *level*
: Set the authorization security level of the service. The argument is a number between 0 and 2. By default, the daemon runs at security level 2.

  `0` Security level 0 is designed to be used for testing and initial setup of the NIS+ namespace. When running at level 0, the daemon does not enforce any access controls. Any client is allowed to perform any operation, including updates and deletions.

  `1` At security level 1, the daemon accepts both `AUTH_SYS` and `AUTH_DES` credentials for authenticating clients and authorizing them to perform NIS+ operations. This is not a secure mode of operation since `AUTH_SYS` credentials are easily forged. It should not be used on networks in which any untrusted users may potentially have access.

  `2` At security level 2, the daemon only accepts authentication using the security mechanisms configured by `nisauthconf(1M)`. The default security mechanism is `AUTH_DES`. Security level 2 is the default if the `-S` option is not used.

`-t` *netid*
: Use *netid* as the transport for communication between `rpc.nisd` and `rpc.nisd_resolv`. The default transport is `ticots(7D)` (`tcp` on SunOS 4.x systems).

`-v`	Verbose. With this option, the daemon sends a running narration of what it is doing to the syslog daemon (see `syslogd(1M)`) at `LOG_INFO` priority. This option is most useful for debugging problems with the service. See also `-A` option.
`-x` *attribute=value*	Specify the value of the named *attribute*. Attributes that control the NIS+ to LDAP mapping operation are derived as follows: 1. Retrieve from LDAP. 2. Override with values from the *mappingfile*, if any. See the `-m` option. 3. Override with values from the command line `-x` options. See `NIS+LDAPmapping(4)` and `rpc.nisd(4)` for the recognized attributes and their syntax. As a special case, you can use the `nisplusLdapConfig*` attributes to derive additional information from LDAP. You can only specify the `nisplusLdapConfig*` attributes in `rpc.nisd(4)` or by means of the command line.
`-Y`	Put the server into NIS (YP) compatibility mode. When operating in this mode, the NIS+ server will respond to NIS Version 2 requests using the version 2 protocol. Because the YP protocol is not authenticated, only those items that have read access to nobody (the unauthenticated request) will be visible through the V2 protocol. It supports only the standard Version 2 maps in this mode (see `-B` option and `NOTES` in `ypfiles(4)`). See `FILES`.
`-z` *number*	Specify the maximum RPC record size that can be used over connection oriented transports. The default is 9000 bytes. If you specify a size less than the default value, the default value will be used instead.

Examples

EXAMPLE 1 Setting up the NIS+ Service

The following example sets up the NIS+ service.

```
example% rpc.nisd
```

EXAMPLE 2 Setting Up NIS+ Service Emulating YP With DNS Forwarding

The following example sets up the NIS+ service, emulating YP with DNS forwarding.

```
example% rpc.nisd -YB
```

EXAMPLE 3 Specifying NIS+ and LDAP Mapping Information

The following example shows how to specify that all additional NIS+ and LDAP mapping information should be retrieved from DN "dc=x,dc=y,dc=z", from the LDAP server at IP address 1.2.3.4, port 389. The examples uses the simple authentication method and the `cn=nisplusAdmin,ou=People,` proxy user. The `-m` option is omitted for clarity in this example..

```
-x nisplusLDAPconfigDN=dc=x,dc=y,dc=z \
-x nisplusLDAPconfigPreferredServerList=127.0.0.1:389 \
-x nisplusLDAPconfigAuthenticationMethod=simple \
-x nisplusLDAPconfigProxyUser=cn=nisplusAdmin,ou=People, \
-x nisplusLDAPconfigProxyPassword=xyzzy
```

Environment Variables

`NETPATH` The transports that the NIS+ service will use can be limited by setting this environment variable. See `netconfig(4)`.

Files

`/var/nis/data/parent.object`	This file describes the namespace that is logically above the NIS+ namespace. The most common type of parent object is a DNS object. This object contains contact information for a server of that domain.
`/var/nis/data/root.object`	This file describes the root object of the NIS+ namespace. It is a standard XDR-encoded NIS+ directory object that can be modified by authorized clients using the `nis_modify(3NSL)` interface.
`/etc/default/rpc.nisd`	LDAP connection and general `rpc.nisd` configuration. You can override some of the settings by command-line options.
`/var/nis/NIS+LDAPmapping`	Default path for LDAP mapping file. See the discussion of the `-m` option.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also

`svcs(1)`, `nis_cachemgr(1M)`, `nisauthconf(1M)`, `nisinit(1M)`, `nissetup(1M)`, `nisldapmaptest(1M)`, `nslookup(1M)`, `rpc.nisd_resolv(1M)`, `rpc.nispasswdd(1M)`, `svcadm(1M)`, `syslogd(1M)`, `nis_modify(3NSL)`, `NIS+LDAPmapping(4)`, `netconfig(4)`, `nisfiles(4)`, `resolv.conf(4)`, `rpc.nisd(4)`, `ypfiles(4)`, `attributes(5)`, `smf(5)`, `ticots(7D)`

Notes

NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

The `rpc.nisd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/nisplus:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name rpc.nisd_resolv, nisd_resolv – NIS+ service daemon

Synopsis

```
rpc.nisd_resolv [-v | -V] [-F [-C fd]] [-t xx] [-p yy]
```

Description `rpc.nisd_resolv` is an auxiliary process which provides DNS forwarding service for NIS hosts requests to both `ypserv` and `rpc.nisd` that are running in the NIS compatibility mode. It is generally started by invoking `rpc.nisd(1M)` with the `-B` option or `ypserv(1M)` with the `-d` option. Although it is not recommended, `rpc.nisd_resolv` can also be started independently with the following options.

This command requires that the `/etc/resolv.conf` file be setup for communication with a DNS nameserver. The `nslookup` utility can be used to verify communication with a DNS nameserver. See `resolv.conf(4)` and `nslookup(1M)`.

Options

`-F` Run in foreground.

`-C` *fd* Use *fd* for service xprt (from `nisd`).

`-v` Verbose. Send output to the syslog daemon.

`-V` Verbose. Send output to stdout.

`-t` *xx* Use transport *xx*.

`-p` *yy* Use transient program# *yy*.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `nslookup(1M)`, `rpc.nisd(1M)`, `resolv.conf(4)`, `attributes(5)`

Notes NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name rpc.nispasswdd, nispasswdd – NIS+ password update daemon

Synopsis

```
/usr/sbin/rpc.nispasswdd [-a attempts] [-c minutes] [-D]
     [-g] [-v]
```

Description `rpc.nispasswdd` daemon is an ONC+ RPC service that services password update requests from `nispasswd(1)` and `yppasswd(1)`. It updates password entries in the NIS+ `passwd` table.

`rpc.nispasswdd` is normally started from a system startup script after the NIS+ server (`rpc.nisd(1M)`) has been started. `rpc.nispasswdd` will determine whether it is running on a machine that is a master server for one or more NIS+ directories. If it discovers that the host is not a master server, then it will promptly exit. It will also determine if `rpc.nisd(1M)` is running in NIS (YP) compatibility mode (the `-Y`option) and will register as `yppasswdd` for NIS (YP) clients as well.

`rpc.nispasswdd` will syslog all failed password update attempts, which will allow an administrator to determine whether someone was trying to "crack" the passwords.

`rpc.nispasswdd` has to be run by a superuser.

Options

`-a` *attempts* Set the maximum number of attempts allowed to authenticate the caller within a password update request session. Failed attempts are `syslogd(1M)` and the request is cached by the daemon. After the maximum number of allowed attempts the daemon severs the connection to the client. The default value is set to `3`.

`-c` *minutes* Set the number of minutes a failed password update request should be cached by the daemon. This is the time during which if the daemon receives further password update requests for the same user and authentication of the caller fails, then the daemon will simply not respond. The default value is set to `30`minutes.

`-D` Debug. Run in debugging mode.

`-g` Generate DES credential. By default the DES credential is not generated for the user if they do not have one. By specifying this option, if the user does not have a credential, then one will be generated for them and stored in the NIS+ cred table.

`-v` Verbose. With this option, the daemon sends a running narration of what it is doing to the syslog daemon. This option is useful for debugging problems.

Exit Status

`0` success

`1` an error has occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also svcs(1), nispasswd(1), passwd(1), yppasswd(1), rpc.nisd(1M), syslogd(1M), svcadm(1M), nsswitch.conf(4), attributes(5), smf(5)

Notes NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

The `rpc.nispasswdd` service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/rpc/nisplus:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name rpc.rexd, rexd – RPC-based remote execution server

Synopsis

```
/usr/sbin/rpc.rexd [-s]
```

Description `rpc.rexd` is the Sun RPC server for remote program execution. This daemon is started by `inetd(1M)` whenever a remote execution request is made.

For non-interactive programs, the standard file descriptors are connected directly to TCP connections. Interactive programs involve pseudo-terminals, in a fashion that is similar to the login sessions provided by `rlogin(1)`. This daemon may use NFS to mount file systems specified in the remote execution request.

There is a `10240` byte limit for arguments to be encoded and passed from the sending to the receiving system.

Options The following option is supported:

`-s` Secure. When specified, requests must have valid DES credentials. If the request does not have a DES credential it is rejected. The default publickey credential is rejected. Only newer `on(1)` commands send DES credentials.

If access is denied with an authentication error, you may have to set your publickey with the `chkey(1)` command.

Specifying the `-s` option without presenting secure credentials will result in an error message: `Unix too weak auth (DesONly)!`

Security `rpc.rexd` uses `pam(3PAM)` for account and session management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the modules to be used for `rpc.rexd`. Here is a partial `pam.conf` file with `rpc.rexd` entries for account and session management using the UNIX module.

```
rpc.rexd   account requisite       pam_roles.so.1
rpc.rexd   account required        pam_projects.so.1
rpc.rexd   account required        pam_unix_account.so.1

rpc.rexd      session required      pam_unix_session.so.1
```

If there are no entries for the `rpc.rexd` service, the entries for the "other" service will be used. `rpc.rexd` uses the `getpwuid()` call to determine whether the given user is a legal user.

Files

`/dev/pts`*n*	Pseudo-terminals used for interactive mode
`/etc/passwd`	Authorized users
`/tmp_rex/rexd??????`	Temporary mount points for remote file systems

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also chkey(1), on(1), rlogin(1), svcs(1), inetadm(1M), inetd(1M), svcadm(1M), pam(3PAM), pam.conf(4), publickey(4), attributes(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5), smf(5)

Diagnostics Diagnostic messages are normally printed on the console, and returned to the requestor.

Notes Root cannot execute commands using rexd client programs such as on(1).

The pam_unix(5) module is no longer supported. Similar functionality is provided by pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), and pam_unix_session(5).

The rpc.rexd service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/rpc/rex:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). Responsibility for initiating and restarting this service is delegated to inetd(1M). Use inetadm(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the svcs(1) command.

Name rpc.rstatd, rstatd – kernel statistics server

Synopsis

```
/usr/lib/netsvc/rstat/rpc.rstatd
```

Description `rpc.rstatd` is a server which returns performance statistics obtained from the kernel. `rup(1)` uses `rpc.rstatd` to collect the `uptime` information that it displays.

`rpc.rstatd` is an RPC service.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also `rup(1)`, `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `services(4)`, `attributes(5)`, `smf(5)`

Notes The `rpc.rstatd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/rstat:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.rusersd, rusersd – network username server

Synopsis

```
/usr/lib/netsvc/rusers/rpc.rusersd
```

Description `rpc.rusersd` is a server that returns a list of users on the host. The `rpc.rusersd` daemon may be started by `inetd`(1M) or `listen`(1M).

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also `svcs`(1), `inetadm`(1M), `inetd`(1M), `listen`(1M), `pmadm`(1M), `sacadm`(1M), `svcadm`(1M), `attributes`(5), `smf`(5)

Notes The `rpc.rusersd` service is managed by the service management facility, `smf`(5), under the service identifier:

```
svc:/network/rpc/rusers:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm`(1M). Responsibility for initiating and restarting this service is delegated to `inetd`(1M). Use `inetadm`(1M) to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs`(1) command.

Name rpc.rwalld, rwalld – network rwall server

Synopsis

```
/usr/lib/netsvc/rwall/rpc.rwalld
```

Description `rpc.rwalld` is a server that handles `rwall(1M)` requests. It is implemented by calling `wall(1M)` on all the appropriate network machines. The `rpc.rwalld` daemon may be started by `inetd(1M)` or `listen(1M)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `listen(1M)`, `rwall(1M)`, `svcadm(1M)`, `wall(1M)`, `attributes(5)`, `smf(5)`

Notes The `rpc.rwalld` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/wall:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.smserverd – removable media device server

Synopsis

```
/usr/lib/smedia/rpc.smserverd
```

Description `rpc.smserverd` is a server that handles requests from client applications, including the Volume Management daemon (`vold(1M)`), for access to removable media devices. In addition to `vold`, `rmformat(1)` and the CDE Filemanager (when performing removable media operations) are `rpc.smserverd` clients. The `rpc.smserverd` daemon is started by `inetd(1M)` when a client makes a call to a Solaris-internal library to access a SCSI, IDE, or USB device. The daemon is not started if a client attempts to access a floppy or PCMCIA device. Once started, the daemon remains active until such time as it is idle for three minutes or more.

The `rpc.smserverd` daemon is provided for the exclusive use of the client applications mentioned above. It has no external, customer-accessible interfaces, including no configuration file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWvolu

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)`, `svcadm(1M)`, `vold(1M)`, `vold.conf(4)`, `attributes(5)`, `smf(5)`

Notes The `rpc.smserverd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/smserver
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.sprayd, sprayd – spray server

Synopsis

```
/usr/lib/netsvc/spray/rpc.sprayd
```

Description `rpc.sprayd` is a server that records the packets sent by `spray(1M)`. The `rpc.sprayd` daemon may be started by `inetd(1M)` or `listen(1M)`.

The service provided by `rpc.sprayd` is not useful as a networking benchmark as it uses unreliable connectionless transports, (udp for example). It can report a large number of packets dropped when the drops were caused by the program sending packets faster than they can be buffered locally (before the packets get to the network medium).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmds

See Also `svcs(1)`, `inetadm(1M)`, `inetd(1M)` `listen(1M)`, `pmadm(1M)`, `sacadm(1M)`, `spray(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

Notes The `rpc.sprayd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/rpc/spray:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

Name rpc.yppasswdd, yppasswdd – server for modifying NIS password file

Synopsis

```
/usr/lib/netsvc/yp/rpc.yppasswdd [-D directory]
     [-nogecos] [-noshell] [-nopw]
     [-m argument1 argument2...]

/usr/lib/netsvc/yp/rpc.yppasswdd
     [passwordfile [adjunctfile]]
     [-nogecos] [-noshell] [-nopw]
     [-m argument1 argument2...]
```

Description `rpc.yppasswdd` is a server that handles password change requests from `yppasswd(1)`. It changes a password entry in the `passwd`, `shadow`, and `security/passwd.adjunct` files. The `passwd` and `shadow` files provide the basis for the `passwd.byname` and `passwd.byuid` maps. The `passwd.adjunct` file provides the basis for the `passwd.adjunct.byname` and `passwd.adjunct.byuid` maps. Entries in the `passwd`, `shadow` or `passwd.adjunct` files are changed only if the password presented by `yppasswd(1)` matches the encrypted password of the entry. All password files are located in the `PWDIR` directory.

If the `-D` option is given, the `passwd`, `shadow`, or `passwd.adjunct` files are placed under the directory path that is the argument to `-D`.

If the `-noshell`, `-nogecos` or `-nopw` options are given, these fields cannot be changed remotely using `chfn`, `chsh`, or `passwd(1)`.

If the `-m` option is given, a `make(1S)` is performed in `/var/yp` after any of the `passwd`, `shadow`, or `passwd.adjunct` files are modified. All arguments following the flag are passed to `make`.

The second of the listed syntaxes is provided only for backward compatibility. If the second syntax is used, the `passwordfile` is the full pathname of the password file and `adjunctfile` is the full pathname of the optional `passwd.adjunct` file. If a shadow file is found in the same directory as `passwordfile`, the `shadowfile` is used as described above. Use of this syntax and the discovery of a `shadowfile` file generates diagnostic output. The daemon, however, starts normally.

The first and second syntaxes are mutually exclusive. You cannot specify the full pathname of the `passwd`, `passwd.adjunct` files and use the `-D` option at the same time.

The daemon is started automatically on the master server of the passwd map by `ypstart(1M)`, which is invoked at boot time by the `svcs:/network/nis/server:default` service.

The server does not insist on the presence of a `shadow` file unless there is no `-D` option present or the directory named with the `-D` option is `/etc`. In addition, a `passwd.adjunct` file is not necessary. If the `-D` option is given, the server attempts to find a `passwd.adjunct` file in the `security` subdirectory of the named directory. For example, in the presence of `-D /var/yp` the server checks for a `/var/yp/security/passwd.adjunct` file.

If only a `passwd` file exists, then the encrypted password is expected in the second field. If both a `passwd` and a `passwd.adjunct` file exist, the encrypted password is expected in the second field of the adjunct file with *##username* in the second field of the `passwd` file. If all three files are in use, the encrypted password is expected in the `shadow` file. Any deviation causes a password update to fail.

If you remove or add a `shadow` or `passwd.adjunct` file after `rpc.yppasswdd` has started, you must stop and restart the daemon to enable it to recognize the change. See `ypstart(1M)` for information on restarting the daemon.

The `rpc.yppasswdd` daemon considers a shell that has a name that begins with 'r' to be a restricted shell. By default, the daemon does not check whether a shell begins with an '`r`'. However, you can tell it to do so by uncommenting the `check_restricted_shell_name=1` line in `/etc/default/yppasswdd`. The result will be to restrict a user's ability to change from his default shell. See `yppasswdd(4)`.

On start up, `yppasswdd` checks for the existence of a NIS to LDAP (N2L) configuration file, `/var/yp/NISLDAPmapping`. If the configuration file is present, the daemon runs in N2L mode. If the file is not present, `yppasswdd` runs in traditional, non-N2L mode.

In N2L mode, changes are written directly to the Directory Information Tree (DIT). If the changes are written successfully, the NIS map is updated. The NIS source files, `passwd`, `shadow`, and `passwd.adjunct`, for example, are not updated. Thus, in N2L mode, the `-D` option is meaningless. In N2L mode, `yppasswdd` propagates changes by calling `yppush(1M)` instead of `ypmake(1M)`. The `-m` option is thus unused.

During an NIS-to-LDAP transition, the `yppasswdd` daemon uses the N2L-specific map, `ageing.byname`, to read and write password aging information to the DIT. If you are not using password aging, then the `ageing.byname` mapping is ignored.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu

See Also

`svcs(1)`, `make(1S)`, `passwd(1)`, `yppasswd(1)`, `inetd(1M)`, `svcadm(1M)`, `ypmake(1M)`, `yppush(1M)`, `ypstart(1M)`, `NISLDAPmapping(4)`, `passwd(4)`, `shadow(4)`, `ypfiles(4)`, `yppasswdd(4)`, `ypserv(4)`, `attributes(5)`, `smf(5)`

Notes

If `make` has not been installed and the `-m` option is given, the daemon outputs a warning and proceeds, effectively ignoring the `-m` flag.

When using the `-D` option, you should make sure that the `PWDIR` of the `/var/yp/Makefile` is set accordingly.

The second listed syntax is supplied only for backward compatibility and might be removed in a future release of this daemon.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications PLC, and cannot be used without permission.

The NIS server service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svcs:/network/nis/server:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name rpc.ypupdated, ypupdated – server for changing NIS information

Synopsis

```
/usr/lib/netsvc/yp/rpc.ypupdated [-is]
```

Description `ypupdated` is a daemon that updates information in the Network Information Service (NIS). `ypupdated` consults the `updaters(4)` file in the `/var/yp` directory to determine which NIS maps should be updated and how to change them.

By default, the daemon requires the most secure method of authentication available to it, either DES (secure) or UNIX (insecure).

On start up, `ypupdated` checks for the existence of a NIS to LDAP (N2L) configuration file, `/var/yp/NISLDAPmapping`. If the file is present, `ypupdated` generates an informational message and exits. `ypupdated` is not supported in N2L mode.

Options

`-i` Accept RPC calls with the insecure `AUTH_UNIX` credentials. This allows programmatic updating of the NIS maps in all networks.

`-s` Accept only calls authenticated using the secure RPC mechanism (`AUTH_DES` authentication). This disables programmatic updating of the NIS maps unless the network supports these calls.

Files

`/var/yp/updaters`	Configuration file for `rpc.updated` command.
`/var/yp/NISLDAPmapping`	Configuration file for N2L

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu
Interface Stability	Evolving

See Also `keyserv(1M)`, `updaters(4)`, `NISLDAPmapping(4)`, `attributes(5)`

System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)

Notes The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two services remains the same. Only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications PLC, and it must not be used without permission.

Name rpld – Network Booting RPL (Remote Program Load) Server

Synopsis

```
/usr/sbin/rpld [-fdDMblgz] interface

/usr/sbin/rpld -a [-fdDMblgz]
```

Description The RPL server provides network booting functionality to x86 clients by listening to boot requests from them according to the RPL protocol specifications. `rpld` runs on both x86 and SPARC systems. Boot requests can be generated by clients using the boot floppy supplied in the distribution. Once the request has been received, the server validates the client and adds it to its internal service list. Subsequent requests from the client to download bootfiles will result in the sending of data frames from the server to the client specifying where to load the boot program in memory. When all the bootfiles have been downloaded, the server specifies where to start execution to initiate the boot process.

In the first synopsis, the interface parameter names the network interface upon which `rpld` is to listen for requests. For example:

```
/usr/sbin/rpld /dev/eri0

/usr/sbin/rpld /dev/smc0
```

In the second synopsis, `rpld` locates all of the network interfaces present on the system and starts a daemon process for each one.

The server starts by reading the default configuration file, or an alternate configuration file if one is specified. If no configuration file can be found, internal default values will be used. Alternatively, command line options are available to override any of the values in the configuration file. After the configuration options are set, it then opens the network interface as specified in the command line and starts listening to RPL boot requests.

Network boot clients have to have information pre-configured on a server for the RPL server to validate and serve them. This involves putting configuration information in both the `ethers(4)` and the `bootparams(4)` databases. The `ethers` database contains a translation from the physical node address to the IP address of the clients and is normally used by the RARP server. The `bootparams` database stores all other information needed for booting off this client, such as the number of bootfiles and the file names of the various boot components. Both databases can be looked up by the RPL server through NIS. See the sub-section `Client Configuration` for information on how to set up these databases.

To assist in the administration and maintenance of the network boot activity, there are two run-time signals that the server will accept to change some run-time parameters and print out useful status information. See the sub-section `Signals` for details.

The RPL server is not limited to the ability to boot only clients. If properly configured, the server should be able to download any bootfiles to the clients.

Client Configuration

The following configuration information is specific to booting x86 clients.

In order to allow clients to boot x86 from across the network, the client's information has to be pre-configured in two databases: `ethers(4)` and `bootparams(4)`. Both databases can be accessed through NIS. Refer to *Solaris 10 Installation Guide: Basic Installations* for information on how to configure a diskless x86 client. The discussion contained in the rest of this section is provided for your information only and should not be performed manually.

The `ethers` database contains a translation table to convert the physical node address to the IP address of the client. Therefore, an IP address must be assigned to the client (if this has not been done already), the node address of the client must be obtained, and then this information needs to be entered in the `ethers` database.

The bulk of the configuration is done in the `bootparams` database. This is a free-format database that essentially contains a number of keyword-value string pairs. A number of keywords have been defined for specific purposes, like the `bootparams` RPC in `bootparamd(1M)`. Three more keywords have been defined for the RPL server. They are `numbootfiles`, `bootfile`, and `bootaddr`. All three keywords must be in lowercase letters with no spaces before or after the equals symbol following the keyword.

`numbootfiles` Specifies the number of files to be downloaded to the network boot client. The format of this option is:

```
numbootfiles=n
```

Always use `numbootfiles=3` to boot x86 across the network.

`bootfile` Specifies the path name of the bootfile to be downloaded and where in memory to start loading the bootfile. A complete path name should be used. For example, assuming the client's IP address is 172.16.32.15:

```
bootfile=/rplboot/172.16.32.15.hw.com:45000
bootfile=/rplboot/172.16.32.15.glue.com:35000
bootfile=/rplboot/172.16.32.15.inetboot=8000
```

The path name following the equals symbol specifies the bootfile to be downloaded, and the hex address following the colon (:) is the absolute address of the memory location to start loading that bootfile. These addresses should be in the range of 7c00 to a0000 (i.e., the base 640K range excluding the interrupt vector and BIOS data areas). Address 45000 for this `hw.com` bootfile is also a suggested value and if possible should not be changed. The address of 35000 for `glue.com` is a suggested value that, if possible, should not be changed. The address of 8000 for `inetboot` is an absolute requirement and should never be changed.

These files, when created following the procedures in the *Solaris 10 Installation Guide: Basic Installations* are actually symbolic links to to the real file to be downloaded to the client.

`hw.com` is linked to a special driver that corresponds to the network interface card of the client. `glue.com` and `inetboot` are generic to all network boot clients.

The order of these bootfile lines is not significant, but because problems have been found with certain boot PROMs, it is highly recommended that the bootfile lines be ordered in descending order of the load addresses.

`bootaddr` The absolute address in memory to start executing after all the bootfiles have been downloaded. This address should always correspond to the address where `glue.com` is being loaded. If possible, always use:

```
bootaddr=35000
```

Options The following options are supported:

`-b` *background_mode* Specify `1` to run the server in the background and relinquish the controlling terminal, or `0` to run in the foreground without relinquishing the controlling terminal. This option corresponds to the *BackGround* setting in the configuration file. If you have specified that the error or warning messages be sent to standard output in the configuration file or by using the `-D` option above, the server cannot be run in background mode. Doing so will cause the server to exit after announcing the error.

`-d` *debug_level* Specify a level of `0` if you do not want any error or warning messages to be generated, or a level from `1` to `9` for increasing amounts of messages. This option corresponds to the *DebugLevel* setting in the configuration file. The default value is `0`. Note that it is best to limit the level to `8` or below; use of level `9` may generate so many debug messages that the performance of the RPL server may be impacted.

`-D` *debug_destination* Specify `0` to send error or warning messages to standard output, `1` to syslogd, and `2` to the log file. This option corresponds to the *DebugDest* setting in the configuration file. The default value is `2`.

`-f` *config_filename* Use this to specify a configuration file name other than the system default `/etc/rpld.conf` file.

`-g` *delay_granularity* This corresponds to the *DelayGran* setting in the configuration file. If retransmission requests from clients do occur, the delay granularity factor will be used to adjust the delay count for this client upwards or downwards. If the retransmission request is caused by data overrun, the delay count will be incremented by delay granularity units to increase the delay between data frames. If the retransmission request is caused by sending data too slowly, this will be used to adjust the delay count downwards to shorten the delay. Eventually the server will settle at the delay count value that

works best with the speed of the client and no retransmission request will be needed. The default value is `2`.

`-l` *log_filename* — Specify an alternate log file name to hold the error or warning messages in connection with the `-D 2` option or the configuration file *DebugDest = 2* setting. This option corresponds to the *LogFile* setting in the configuration file. The default is `/var/spool/rpld.log`.

`-M` *maximum_clients* — Specify the maximum number of simultaneous network boot clients to be served. This option corresponds to the *MaxClients* setting in the configuration file. A value of `−1` means unlimited, and the actual number will depend on available system resources. The default value is `−1`.

`-s` *start_delay_count* — This option corresponds to the *StartDelay* setting in the configuration file. Specify the number of delay units between outgoing data frames sent to clients to avoid retransmission requests from them. Using the LLC type `1` protocol, data transfer is a one-way, best-effort delivery mechanism. The server, without any type of delay mechanism, can overrun the client by sending data frames too quickly. Therefore, a variable delay is built into the server to limit the speed of sending data to the clients, thus avoiding the clients sending back retransmission requests. This value should be machine environment specific. If you have a fast server machine but slow client machines, you may want to set a large start delay count. If you have comparable server and client machines, the delay count may be set to `1`. The delay is only approximate and should not be taken as an accurate measure of time. There is no specific correlation between the delay unit and the actual time of delay. The default value is `20`.

`-z` *frame_size* — This option corresponds to the *FrameSize* setting in the configuration file. This specifies the size of the data frames used to send data to the clients. This is limited by the underlying physical medium. For `ethernet/802.3`, the maximum physical frame size is 1500 octets. The default value is `1500`. Note that the protocol overhead of LLC1 and RPL is 32 octets, resulting in a maximum data length of 1468 octets.

Signals The RPL server accepts two signals to change run-time parameters and display status information, respectively:

`HANGUP` — This will cause the RPL server to reread the default configuration file `/etc/rpld.conf` or an alternate configuration file if one is specified when the server is started. New values of certain parameters can be used immediately, such

as *DebugLevel*, *DebugDest*, *LogFile*, *DelayGran*, and *FrameSize*. For *MaxClients*, if the server is already serving more than the new value, the server will not accept additional boot requests until the number has fallen below the *MaxClients* parameter. For *StartDelay*, this will only affect new boot requests. All the existing delay counts for the various clients in service will not be affected. Finally, the *BackGround* parameter will have no effect once the server has been running. You cannot change the mode of service without first killing the server and then restarting it.

`USR1` This signal will cause the server to dump all the parameter values and the status of each individual boot client to the destination specified by *DebugDest*.

Files

- `/usr/sbin/rpld`
- `/etc/rpld.conf`
- `/var/spool/rpld.log`
- `/etc/ethers`
- `/etc/bootparams`
- `/rplboot`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86, SPARC
Availability	SUNWbsu

See Also `bootparamd(1M)`, `in.rarpd(1M)`, `bootparams(4)`, `ethers(4)`, `nsswitch.conf(4)`, `rpld.conf(4)`, `attributes(5)`

Solaris 10 Installation Guide: Basic Installations

Name rquotad – remote quota server

Synopsis

```
/usr/lib/nfs/rquotad
```

Description `rquotad` is an `rpc(4)` server which returns quotas for a user of a local file system which is mounted by a remote machine over the NFS. The results are used by `quota(1M)` to display user quotas for remote file systems. The `rquotad` daemon is normally invoked by `inetd(1M)`.

Usage See `largefile(5)` for the description of the behavior of `rquotad` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Files

`quotas`	quota file at a UFS file system root

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfssu

See Also `svcs(1)`, `automountd(1M)`, `inetadm(1M)`, `inetd(1M)`, `mount_nfs(1M)`, `quota(1M)`, `share_nfs(1M)`, `svcadm(1M)`, `rpc(4)`, `services(4)`, `attributes(5)`, `largefile(5)`, `smf(5)`

Solaris 10 Installation Guide: Basic Installations

Notes The `rquotad` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nfs/rquota
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. Responsibility for initiating and restarting this service is delegated to `inetd(1M)`. Use `inetadm(1M)` to make configuration changes and to view configuration information for this service. The service's status can be queried using the `svcs(1)` command.

If it is disabled, it will be enabled by `mount_nfs(1M)`, `share_nfs(1M)`, and `automountd(1M)` unless its `application/auto_enable` property is set to `false`.

Name rsh, restricted_shell – restricted shell command interpreter

Synopsis

```
/usr/lib/rsh [-acefhiknprstuvx] [argument]...
```

Description `rsh` is a limiting version of the standard command interpreter `sh`, used to restrict logins to execution environments whose capabilities are more controlled than those of `sh` (see `sh(1)` for complete description and usage).

When the shell is invoked, it scans the environment for the value of the environmental variable, `SHELL`. If it is found and `rsh` is the file name part of its value, the shell becomes a restricted shell.

The actions of `rsh` are identical to those of `sh`, except that the following are disallowed:

- changing directory (see `cd(1)`),
- setting the value of `$PATH`,
- pecifying path or command names containing `/`,
- redirecting output (`>` and `>>`).

The restrictions above are enforced after *.profile* is interpreted.

A restricted shell can be invoked in one of the following ways:

1. `rsh` is the file name part of the last entry in the `/etc/passwd` file (see `passwd(4)`);
2. the environment variable `SHELL` exists and `rsh` is the file name part of its value; the environment variable `SHELL` needs to be set in the `.login` file;
3. the shell is invoked and `rsh` is the file name part of argument 0;
4. the shell is invoke with the `-r` option.

When a command to be executed is found to be a shell procedure, `rsh` invokes `sh` to execute it. Thus, it is possible to provide to the end-user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end-user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the *.profile* (see `profile(4)`) has complete control over user actions by performing guaranteed setup actions and leaving the user in an appropriate directory (probably *not* the login directory).

The system administrator often sets up a directory of commands (that is, `/usr/rbin`) that can be safely invoked by a restricted shell. Some systems also provide a restricted editor, `red`.

Exit Status Errors detected by the shell, such as syntax errors, cause the shell to return a non-zero exit status. If the shell is being used non-interactively execution of the shell file is abandoned. Otherwise, the shell returns the exit status of the last command executed.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also Intro(1), cd(1), login(1), rsh(1), sh(1), exec(2), passwd(4), profile(4), attributes(5)

Notes The restricted shell, /usr/lib/rsh, should not be confused with the remote shell, /usr/bin/rsh, which is documented in rsh(1).

Name rtc – provide all real-time clock and GMT-lag management

Synopsis

```
/usr/sbin/rtc [-c] [-z zone-name]
```

Description On x86 systems, the `rtc` command reconciles the difference in the way that time is established between UNIX and MS-DOS systems. UNIX systems utilize Greenwich Mean Time (GMT), while MS-DOS systems utilize local time.

Without arguments, `rtc` displays the currently configured time zone string. The currently configured time zone string is based on what was last recorded by `rtc -z` *zone-name*.

The `rtc` command is not normally run from a shell prompt; it is generally invoked by the system. Commands such as `date(1)` and `rdate(1M)`, which are used to set the time on a system, invoke `/usr/sbin/rtc -c` to ensure that daylight savings time (DST) is corrected for properly.

Options

`-c` This option checks for DST and makes corrections if necessary. It is normally run once a day by a `cron` job.

If there is no RTC time zone or `/etc/rtc_config` file, this option will do nothing.

`-z` *zone-name* This option, which is normally run by the system at software installation time, is used to specify the time zone in which the RTC is to be maintained. It updates the configuration file `/etc/rtc_config` with the name of the specified zone and the current GMT lag for that zone. If there is an existing `rtc_config` file, this command will update it. If not, this command will create it.

Files

`/etc/rtc_config` The data file used to record the time zone and GMT lag. This file is completely managed by `/usr/sbin/rtc`, and it is read by the kernel.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	SUNWcsu

See Also `date(1)`, `rdate(1M)`, `attributes(5)`

Name rtquery – query routing daemons for their routing tables

Synopsis

```
rtquery [-np1] [-w timeout] [-r addr] [-a secret] host...

rtquery [-t operation] host...
```

Description The `rtquery` command is used to query a RIP network routing daemon, `in.routed(1M)` or `GateD`, for its routing table by sending a request or poll command. The routing information in any routing response packets returned is displayed numerically and symbolically.

By default, `rtquery` uses the request command. When the -p option is specified, `rtquery` uses the poll command, an undocumented extension to the RIP protocol supported by `GateD`. When querying `GateD`, the poll command is preferred over the request command because the response is not subject to Split Horizon and/or Poisoned Reverse, and because some versions of `GateD` do not answer the request command. `in.routed` does not answer the poll command, but recognizes requests coming from `rtquery` and so answers completely.

The `rtquery` command is also used to turn tracing on or off in `in.routed`.

Options The following options are supported:

`-a passwd=`*XXX*
`-a md5_passwd=`*XXX*|*KeyID*
: Causes the query to be sent with the indicated cleartext or MD5 password.

`-n`
: Displays only the numeric network and host addresses instead of both numeric and symbolic names.

`-p`
: Uses the poll command to request full routing information from `GateD`. This is an undocumented extension RIP protocol supported only by `GateD`.

`-r` *addr*
: Asks about the route to destination *addr*.

`-t` *operation*
: Changes tracing, where *operation* is one of the actions listed below. Requests from processes not running with UID 0 or on distant networks are generally ignored by the daemon except for a message in the system log. `GateD` is likely to ignore these debugging requests.

  on=*tracefile*
  : Turns tracing on, directing tracing into the specified file. That file must have been specified when the daemon was started or have the name, `/var/log/in.routed.trace`.

  more
  : Increases the debugging level.

  off
  : Turns off tracing.

dump — Dumps the daemon's routing table to the current trace file.

`-w` *timeout* — Changes the delay for an answer from each host. By default, each host is given 15 seconds to respond.

`-1` — Queries using RIP version 1 instead of RIP version 2.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWroute

See Also `in.routed(1M)`, `route(1M)`, `gateways(4)`, `attributes(5)`, `icmp(7P)`, `inet(7P)`, `udp(7P)`

Routing Information Protocol, RIPv1, RFC 1058

Routing Information Protocol, RIPv2, RFC 2453, STD 0056

Name runacct – run daily accounting

Synopsis `/usr/lib/acct/runacct [`*mmdd* `[`*state*`]]`

Description `runacct` is the main daily accounting shell procedure. It is normally initiated using `cron`. `runacct` processes connect, fee, disk, and process accounting files. It also prepares summary files for `prdaily` or billing purposes. `runacct` is distributed only to source code licensees.

`runacct` takes care not to damage active accounting files or summary files in the event of errors. It records its progress by writing descriptive diagnostic messages into `active`. When an error is detected, a message is written to `/dev/console`, mail (see `mail(1)`) is sent to `root` and `adm`, and `runacct` terminates. `runacct` uses a series of lock files to protect against re-invocation. The files `lock` and `lock1` are used to prevent simultaneous invocation, and `lastdate` is used to prevent more than one invocation per day.

`runacct` breaks its processing into separate, restartable *states* using `statefile` to remember the last *state* completed. It accomplishes this by writing the *state* name into `statefile`. `runacct` then looks in `statefile` to see what it has done and to determine what to process next. *states* are executed in the following order:

`SETUP`	Move active accounting files into working files.
`WTMPFIX`	Verify integrity of `wtmpx` file, correcting date changes if necessary.
`CONNECT`	Produce connect session records in `tacct.h` format.
`PROCESS`	Convert process accounting records into `tacct.h` format.
`MERGE`	Merge the connect and process accounting records.
`FEES`	Convert output of `chargefee` into `tacct.h` format, merge with connect, and process accounting records.
`DISK`	Merge disk accounting records with connect, process, and fee accounting records.
`MERGETACCT`	Merge the daily total accounting records in `daytacct` with the summary total accounting records in `/var/adm/acct/sum/tacct`.
`CMS`	Produce command summaries.
`USEREXIT`	Any installation dependent accounting programs can be included here.
`CLEANUP`	Clean up temporary files and exit. To restart `runacct` after a failure, first check the `active` file for diagnostics, then fix any corrupted data files, such as `pacct` or `wtmpx`. The `lock`, `lock1`, and `lastdate` files must be removed before `runacct` can be restarted. The argument *mmdd* is necessary if `runacct` is being restarted. *mmdd* specifies the month and day for which `runacct` will rerun the accounting. The entry point for processing is based on the contents of `statefile`; to override this, include the desired *state* on the command line to designate where processing should begin.

Examples

EXAMPLE 1 Starting `runacct`

The following example starts `runacct`:

```
example% nohup runacct 2> /var/adm/acct/nite/fd2log &
```

EXAMPLE 2 Restarting `runacct`

The following example restarts `runacct`:

```
example% nohup runacct 0601 2>> /var/adm/acct/nite/fd2log &
```

EXAMPLE 3 Restarting `runacct` at a Specific State

The following example restarts `runacct` at a specific state:

```
example% nohup runacct 0601 MERGE 2>> /var/adm/acct/nite/fd2log &
```

Files

`/var/adm/wtmpx`	History of user access and administration information
`/var/adm/pacct`*incr*	
`/var/adm/acct/nite/active`	
`/var/adm/acct/nite/daytacct`	
`/var/adm/acct/nite/lock`	
`/var/adm/acct/nite/lock1`	
`/var/adm/acct/nite/lastdate`	
`/var/adm/acct/nite/statefile`	

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

See Also

`acctcom(1)`, `mail(1)`, `acct(1M)`, `acctcms(1M)`, `acctcon(1M)`, `acctmerg(1M)`, `acctprc(1M)`, `acctsh(1M)`, `cron(1M)`, `fwtmp(1M)`, `acct(2)`, `acct.h(3HEAD)`, `utmpx(4)`, `attributes(5)`

Notes

It is not recommended to restart `runacct` in the SETUP *state*. Run SETUP manually and restart using:

`runacct` *mmdd* WTMPFIX

If `runacct` failed in the PROCESS *state,* remove the last `ptacct` file because it will not be complete.

The `runacct` command can process a maximum of

- 6000 distinct sessions

- 1000 distinct terminal lines
- 2000 distinct login names

during a single invocation of the command. If at some point the actual number of any one of these items exceeds the maximum, the command will not succeed.

Do not invoke `runacct` at the same time as `ckpacct`, as there may be a conflict if both scripts attempt to execute `turnacct switch` simultaneously.

Name rwall – write to all users over a network

Synopsis

```
/usr/sbin/rwall hostname...
/usr/sbin/rwall -n netgroup...
/usr/sbin/rwall -h hostname -n netgroup
```

Description `rwall` reads a message from standard input until EOF. It then sends this message, preceded by the line:

```
Broadcast Message . . .
```

to all users logged in on the specified host machines. With the `-n` option, it sends to the specified network groups.

Options

`-n` *netgroup* Send the broadcast message to the specified network groups.

`-h` *hostname* Specify the `hostname`, the name of the host machine.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmdc

See Also `inetd(1M)`, `listen(1M)`, `pmadm(1M)`, `sacadm(1M)`, `wall(1M)`, `attributes(5)`

Notes The timeout is fairly short to allow transmission to a large group of machines (some of which may be down) in a reasonable amount of time. Thus the message may not get through to a heavily loaded machine.

Name sac – service access controller

Synopsis

```
sac -t sanity_interval

/usr/lib/saf/sac
```

Description The Service Access Controller (SAC) is the overseer of the server machine. It is started when the server machine enters multiuser mode. The SAC performs several important functions as explained below.

Customizing the SAC Environment

When `sac` is invoked, it first looks for the per-system configuration script `/etc/saf/_sysconfig`. `sac` interprets `_sysconfig` to customize its own environment. The modifications made to the SAC environment by `_sysconfig` are inherited by all the children of the SAC. This inherited environment may be modified by the children.

Starting Port Monitors

After it has interpreted the `_sysconfig` file, the `sac` reads its administrative file `/etc/saf/_sactab`. `_sactab` specifies which port monitors are to be started. For each port monitor to be started, `sac` forks a child (see `fork(2)`) and creates a `utmpx` entry with the `type` field set to `LOGIN_PROCESS`. Each child then interprets its per-port monitor configuration script `/etc/saf/`*pmtag*`/_config` , if the file exists. These modifications to the environment affect the port monitor and will be inherited by all its children. Finally, the child process `exec`s the port monitor, using the command found in the `_sactab` entry. (See `sacadm`; this is the command given with the `-c` option when the port monitor is added to the system.)

Polling Port Monitors to Detect Failure

The `-t` option sets the frequency with which `sac` polls the port monitors on the system. This time may also be thought of as half of the maximum latency required to detect that a port monitor has failed and that recovery action is necessary.

Administrative functions

The Service Access Controller represents the administrative point of control for port monitors. Its administrative tasks are explained below.

When queried (`sacadm` with either `-l` or `-L`), the Service Access Controller returns the status of the port monitors specified, which `sacadm` prints on the standard output. A port monitor may be in one of six states:

`ENABLED` The port monitor is currently running and is accepting connections. See `sacadm(1M)` with the `-e` option.

`DISABLED` The port monitor is currently running and is not accepting connections. See `sacadm` with the `-d` option, and see `NOTRUNNING`, below.

`STARTING` The port monitor is in the process of starting up. `STARTING` is an intermediate state on the way to `ENABLED` or `DISABLED`.

`FAILED` The port monitor was unable to start and remain running.

`STOPPING` The port monitor has been manually terminated but has not completed its shutdown procedure. `STOPPING` is an intermediate state on the way to `NOTRUNNING`.

`NOTRUNNING` The port monitor is not currently running. (See `sacadm` with `-k`.) This is the normal "not running" state. When a port monitor is killed, all ports it was monitoring are inaccessible. It is not possible for an external user to tell whether a port is not being monitored or the system is down. If the port monitor is not killed but is in the `DISABLED` state, it may be possible (depending on the port monitor being used) to write a message on the inaccessible port telling the user who is trying to access the port that it is disabled. This is the advantage of having a `DISABLED` state as well as the `NOTRUNNING` state.

When a port monitor terminates, the SAC removes the `utmpx` entry for that port monitor.

The SAC receives all requests to enable, disable, start, or stop port monitors and takes the appropriate action.

The SAC is responsible for restarting port monitors that terminate. Whether or not the SAC will restart a given port monitor depends on two things:

- The restart count specified for the port monitor when the port monitor was added by `sacadm`; this information is included in `/etc/saf/`*pmtag*`/_sactab`.
- The number of times the port monitor has already been restarted.

Security `sac` uses `pam(3PAM)` for session management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the session management module to be used for `sac`. Here is a partial `pam.conf` file with entries for `sac` using the UNIX session management module.

```
sac   session    required    pam_unix_session.so.1
```

If there are no entries for the `sac` service, then the entries for the "other" service will be used.

Options `-t` *sanity_interval* Sets the frequency (*sanity_interval*) with which `sac` polls the port monitors on the system.

Files `/etc/saf/_sactab`

`/etc/saf/_sysconfig`

`/var/adm/utmpx`

`/var/saf/_log`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also pmadm(1M), sacadm(1M), fork(2) pam(3PAM), pam.conf(4), attributes(5), pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), pam_unix_session(5)

Notes The pam_unix(5) module is no longer supported. Similar functionality is provided by pam_authtok_check(5), pam_authtok_get(5), pam_authtok_store(5), pam_dhkeys(5), pam_passwd_auth(5), pam_unix_account(5), pam_unix_auth(5), and pam_unix_session(5).

Name sacadm – service access controller administration

Synopsis

```
sacadm -a -p pmtag -t type -c cmd -v ver [-f dx] [-n count]
     [-y comment] [-z script]

sacadm -r -p pmtag

sacadm -s -p pmtag

sacadm -k -p pmtag

sacadm -e -p pmtag

sacadm -d -p pmtag

sacadm -l [-p pmtag | -t type]

sacadm -L [-p pmtag | -t type]

sacadm -g -p pmtag [-z script]

sacadm -G [-z script]

sacadm -x [-p pmtag]
```

Description `sacadm` is the administrative command for the upper level of the Service Access Facility hierarchy (port monitor administration). `sacadm` performs the following functions:

- adds or removes a port monitor
- starts or stops a port monitor
- enables or disables a port monitor
- installs or replaces a per-system configuration script
- installs or replaces a per-port monitor configuration script
- prints requested port monitor information

Requests about the status of port monitors (`-l` and `-L`) and requests to print per-port monitor and per-system configuration scripts (`-g` and `-G` without the `-z` option) may be executed by any user on the system. Other `sacadm` commands may be executed only by the super-user.

Options

`-a` Add a port monitor. When adding a port monitor, `sacadm` creates the supporting directory structure in `/etc/saf` and `/var/saf` and adds an entry for the new port monitor to `/etc/saf/_sactab`. The file `_sactab` already exists on the delivered system. Initially, it is empty except for a single line, which contains the version number of the Service Access Controller. Unless the command line that adds the new port monitor includes the `-f` option with the `-x` argument, the new port monitor will be started. Because of the complexity of the options and arguments that follow the `-a` option, it may be convenient to use a command script or the menu system to add port monitors.

`-c` *cmd* Execute the command string *cmd* to start a port monitor. The `-c` option may be used only with a `-a`. A `-a` option requires a `-c`.

`-d` Disable the port monitor *pmtag*.

`-e` Enable the port monitor *pmtag*.

`-f dx` The `-f` option specifies one or both of the following two flags which are then included in the flags field of the `_sactab` entry for the new port monitor. If the `-f` option is not included on the command line, no flags are set and the default conditions prevail. By default, a port monitor is started. A `-f` option with no following argument is illegal.

d Do not enable the new port monitor.

x Do not start the new port monitor.

`-g` The `-g` option is used to request output or to install or replace the per-port monitor configuration script `/etc/saf/`*pmtag*`/_config`. `-g` requires a `-p` option. The `-g` option with only a `-p` option prints the per-port monitor configuration script for port monitor *pmtag*. The `-g` option with a `-p` option and a `-z` option installs the file `script` as the per-port monitor configuration script for port monitor *pmtag*. Other combinations of options with `-g` are invalid.

`-G` The `-G` option is used to request output or to install or replace the per-system configuration script `/etc/saf/_sysconfig`. The `-G` option by itself prints the per-system configuration script. The `-G` option in combination with a `-z` option installs the file `script` as the per-system configuration script. Other combinations of options with a `-G` option are invalid.

`-k` Stop port monitor *pmtag*.

`-l` The `-l` option is used to request port monitor information. The `-l` by itself lists all port monitors on the system. The `-l` option in combination with the `-p` option lists only the port monitor specified by *pmtag*. A `-l` in combination with the `-t` option lists all port monitors of type `type`. Any other combination of options with the `-l` option is invalid.

`-L` The `-L` option is identical to the `-l` option except that the output appears in a condensed format.

`-n` *count* Set the restart count to *count*. If a restart count is not specified, count is set to `0`. A count of `0` indicates that the port monitor is not to be restarted if it fails.

`-p` *pmtag* Specifies the tag associated with a port monitor.

`-r` Remove port monitor *pmtag*. `sacadm` removes the port monitor entry from `/etc/saf/_sactab`. If the removed port monitor is not running, then no further action is taken. If the removed port monitor is running, the Service Access Controller (SAC) sends it `SIGTERM` to indicate that it should shut down. Note that the port monitor's directory structure remains intact.

`-s` Start a port monitor. The SAC starts the port monitor *pmtag*.

`-t type` Specifies the port monitor type.

`-v ver` Specifies the version number of the port monitor. This version number may be given as

`-v ‘`*pmspec* `-V‘`

where *pmspec* is the special administrative command for port monitor *pmtag*. This special command is `ttyadm` for `ttymon` and `nlsadmin` for `listen`. The version stamp of the port monitor is known by the command and is returned when *pmspec* is invoked with a `-V` option.

`-x` The `-x` option by itself tells the SAC to read its database file (`_sactab`). The `-x` option with the `-p` option tells port monitor *pmtag* to read its administrative file.

`-y` *comment* Include *comment* in the `_sactab` entry for port monitor *pmtag*.

`-z script` Used with the `-g` and `-G` options to specify the name of a file that contains a configuration script. With the `-g` option, `script` is a per-port monitor configuration script; with `-G` it is a per-system configuration script. Modifying a configuration script is a three-step procedure. First a copy of the existing script is made (`-g` or `-G`). Then the copy is edited. Finally, the copy is put in place over the existing script (`-g` or `-G` with `-z`).

Output If successful, `sacadm` will exit with a status of `0`. If `sacadm` fails for any reason, it will exit with a nonzero status. Options that request information will write the information on the standard output. In the condensed format (`-L`), port monitor information is printed as a sequence of colon-separated fields; empty fields are indicated by two successive colons. The standard format (`-l`) prints a header identifying the columns, and port monitor information is aligned under the appropriate headings. In this format, an empty field is indicated by a hyphen. The comment character is `#`.

Examples **EXAMPLE 1** A sample output of the `sacadm` command.

The following command line adds a port monitor. The port monitor tag is `npack`; its type is `listen`; if necessary, it will restart three times before failing; its administrative command is `nlsadmin`; and the configuration script to be read is in the file `script`:

```
sacadm -a -p npack -t listen -c /usr/lib/saf/listen npack
-v ‘nlsadmin -V‘ -n 3 -z script
```

Remove a port monitor whose tag is `pmtag`:

```
sacadm -r -p pmtag
```

Start the port monitor whose tag is `pmtag`:

EXAMPLE 1 A sample output of the `sacadm` command. *(Continued)*

```
sacadm -s -p pmtag
```

Stop the port monitor whose tag is `pmtag`:

```
sacadm -k -p pmtag
```

Enable the port monitor whose tag is `pmtag`:

```
sacadm -e -p pmtag
```

Disable the port monitor whose tag is `pmtag`:

```
sacadm -d -p pmtag
```

List status information for all port monitors:

```
sacadm -l
```

List status information for the port monitor whose tag is `pmtag`:

```
sacadm -l -p pmtag
```

List the same information in condensed format:

```
sacadm -L -p pmtag
```

List status information for all port monitors whose type is `listen`:

```
sacadm -l -t listen
```

Replace the per-port monitor configuration script associated with the port monitor whose tag is `pmtag` with the contents of the file `file.config`:

```
sacadm -g -p pmtag -z file.config
```

Files `/etc/saf/_sactab`

`/etc/saf/_sysconfig`

`/etc/saf/`*`pmtag`*`/_config`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `pmadm(1M)`, `sac(1M)`, `doconfig(3NSL)`, `attributes(5)`

Name saf – Service Access Facility

Description The SAF generalizes the procedures for service access so that login access on the local system and network access to local services are managed in similar ways. Under the SAF, systems may access services using a variety of port monitors, including ttymon, the listener, and port monitors written expressly for a user's application. The manner in which a port monitor observes and manages access ports is specific to the port monitor and not to any component of the SAF. Users may therefore extend their systems by developing and installing their own port monitors. One of the important features of the SAF is that it can be extended in this way by users.

Relative to the SAF, a service is a process that is started. There are no restrictions on the functions a service may provide. The SAF consists of a controlling process, the service access controller (SAC), and two administrative levels corresponding to two levels in the supporting directory structure. The top administrative level is concerned with port monitor administration, the lower level with service administration. The SAC is documented in the sac(1M) man page. The administrative levels and associated utilities are documented in the *System Administration Guide - Volume II*. The requirements for writing port monitors and the functions a port monitor must perform to run under the SAF and the SAC are documented here.

Port Monitors A port monitor is a process that is responsible for monitoring a set of homogeneous, incoming ports on a machine. A port monitor's major purpose is to detect incoming service requests and to dispatch them appropriately.

A port is an externally seen access point on a system. A port may be an address on a network (TSAP or PSAP), a hardwired terminal line, an incoming phone line, etc. The definition of what constitutes a port is strictly a function of the port monitor itself.

A port monitor performs certain basic functions. Some of these are required to conform to the SAF; others may be specified by the requirements and design of the port monitor itself. Port monitors have two main functions: managing ports and monitoring ports for indications of activity.

Port Management — The first function of a port monitor is to manage a port. The actual details of how a port is managed are defined by the person who defines the port monitor. A port monitor is not restricted to handling a single port; it may handle multiple ports simultaneously.

Some examples of port management are setting the line speed on incoming phone connections, binding an appropriate network address, reinitializing the port when the service terminates, outputting a prompt, etc.

Activity Monitoring — The second function of a port monitor is to monitor the port or ports for which it is responsible for indications of activity. Two types of activity may be detected.

The first is an indication to the port monitor to take some port monitor-specific action. Pressing the break key to indicate that the line speed should be cycled is an example of a port monitor activity. Not all port monitors need to recognize and respond to the same indications. The indication used to attract the attention of the port monitor is defined by the person who defines the port monitor.

The second is an incoming service request. When a service request is received, a port monitor must be able to determine which service is being requested from the port on which the request is received. The same service may be available on more than one port.

Other Port Monitor Functions

This section briefly describes other port monitor functions.

Restricting Access to the System

A port monitor must be able to restrict access to the system without disturbing services that are still running. In order to do this, a port monitor must maintain two internal states: enabled and disabled. The port monitor starts in the state indicated by the ISTATE environment variable provided by the sac. See sac(1M) for details. Enabling or disabling a port monitor affects all ports for which the port monitor is responsible. If a port monitor is responsible for a single port, only that port will be affected. If a port monitor is responsible for multiple ports, the entire collection of ports will be affected. Enabling or disabling a port monitor is a dynamic operation: it causes the port monitor to change its internal state. The effect does not persist across new invocations of the port monitor. Enabling or disabling an individual port, however, is a static operation: it causes a change to an administrative file. The effect of this change will persist across new invocations of the port monitor.

Creating `utmpx` Entries

Port monitors are responsible for creating `utmpx` entries with the type field set to USER_PROCESS for services they start. If this action has been specified, by using the `-fu` option in the `pmadm` command line that added the service, these `utmpx` entries may

	in turn be modified by the service. When the service terminates, the `utmpx` entry must be set to DEAD_PROCESS.
Port Monitor Process IDs and Lock Files	When a port monitor starts, it writes its process id into a file named `_pid` in the current directory and places an advisory lock on the file.
Changing the Service Environment: Running	`doconfig(3NSL)` Before invoking the service designated in the port monitor administrative file, `_pmtab`, a port monitor must arrange for the per-service configuration script to be run, if one exists, by calling the library function `doconfig(3NSL)`. Because the per-service configuration script may specify the execution of restricted commands, as well as for other security reasons, port monitors are invoked with root permissions. The details of how services are invoked are specified by the person who defines the port monitor.
Terminating a Port Monitor	A port monitor must terminate itself gracefully on receipt of the signal SIGTERM. The termination sequence is the following: 1. The port monitor enters the stopping state; no further service requests are accepted. 2. Any attempt to re-enable the port monitor will be ignored. 3. The port monitor yields control of all ports for which it is responsible. It must be possible for a new instantiation of the port monitor to start correctly while a previous instantiation is stopping. 4. The advisory lock on the process id file is released. Once this lock is released, the contents of the process id file are undefined and a new invocation of the port monitor may be started.

SAF Files

This section briefly covers the files used by the SAF.

The Port Monitor Administrative File

A port monitor's current directory contains an administrative file named `_pmtab`; `_pmtab` is maintained by the `pmadm` command in conjunction with a port monitor-specific administrative command.

The port monitor administrative command for a listen port monitor is `nlsadmin(1M)`; the port monitor administrative command for ttymon is `ttyadm(1M)`. Any port monitor written by a user must be provided with an administrative command specific to that port monitor to perform similar functions.

Per-Service Configuration Files

A port monitor's current directory also contains the per-service configuration scripts, if they exist. The names of the per-service configuration scripts correspond to the service tags in the `_pmtab` file.

Private Port Monitor Files

A port monitor may create private files in the directory `/var/saf/`*tag*, where *tag* is the name of the port monitor. Examples of private files are log files or temporary files.

The SAC/Port Monitor Interface

The SAC creates two environment variables for each port monitor it starts:PMTAG and ISTATE.

This variable is set to a unique port monitor tag by the SAC. The port monitor uses this tag to identify itself in response to `sac` messages. ISTATE is used to indicate to the port monitor what its initial internal state should be. ISTATE is set to “enabled” or “disabled” to indicate that the port monitor is to start in the enabled or disabled state respectively.

The SAC performs a periodic sanity poll of the port monitors. The SAC communicates with port monitors through FIFOs. A port monitor should open _pmpipe, in the current directory, to receive messages from the SAC and `../_sacpipe` to send return messages to the SAC.

Message Formats

This section describes the messages that may be sent from the SAC to a port monitor (`sac` messages), and from a port monitor to the SAC (port monitor messages). These messages are sent through FIFOs and are in the form of C structures.

`sac` Messages

The format of messages from the SAC is defined by the structure `sacmsg`:

```
struct sacmsg
{
    int sc_size; /* size of optional data portion */
    char sc_type; /* type of message */
```

```
};
```

The SAC may send four types of messages to port monitors. The type of message is indicated by setting the `sc_type` field of the `sacmsg` structure to one of the following:

SC_STATUS status request

SC_ENABLE enable message

SC_DISABLE disable message

SC_READDB message indicating that the port monitor's _pmtab file should be read

The `sc_size` field indicates the size of the optional data part of the message. See "Message Classes." For Solaris, `sc_size` should always be set to 0. A port monitor must respond to every message sent by the sac.

Port Monitor Messages

The format of messages from a port monitor to the SAC is defined by the structure `pmmsg`:

```
struct pmmsg {
    char pm_type;               /* type of message */
    unchar_t pm_state;          /* current state of port monitor */
    char pm_maxclass;           /* maximum message class this port
                                   monitor understands */
    char pm_tag[PMTAGSIZE + 1]; /* port monitor's tag */
    int pm_size;                /* size of optional data portion */
};
```

Port monitors may send two types of messages to the SAC. The type of message is indicated by setting the `pm_type` field of the `pmmsg` structure to one of the following:

PM_STATUS state information

PM_UNKNOWN negative acknowledgment

For both types of messages, the `pm_tag` field is set to the port monitor's tag and the `pm_state` field is set to the port monitor's current state. Valid states are:

PM_STARTING starting

PM_ENABLED enabled

PM_DISABLED disabled

PM_STOPPING stopping

The current state reflects any changes caused by the last message from the SAC. The status message is the normal return message. The negative acknowledgment should be sent only when the message received is not understood. `pm_size` indicates the size of the optional data part of the message. `pm_maxclass` is used to specify a message class. Both are discussed under

"Message Classes." In Solaris, always set `pm_maxclass` to 1 and `sc_size` to 0. Port monitors may never initiate messages; they may only respond to messages that they receive.

Message Classes The concept of message class has been included to accommodate possible SAF extensions. The messages described above are all class 1 messages. None of these messages contains a variable data portion; all pertinent information is contained in the message header. If new messages are added to the protocol, they will be defined as new message classes (for example, class 2). The first message the SAC sends to a port monitor will always be a class 1 message. Since all port monitors, by definition, understand class 1 messages, the first message the SAC sends is guaranteed to be understood. In its response to the SAC, the port monitor sets the `pm_maxclass` field to the maximum message class number for that port monitor. The SAC will not send messages to a port monitor from a class with a larger number than the value of `pm_maxclass`. Requests that require messages of a higher class than the port monitor can understand will fail. For Solaris, always set `pm_maxclass` to 1.

For any given port monitor, messages of class `pm_maxclass` and messages of all classes with values lower than `pm_maxclass` are valid. Thus, if the `pm_maxclass` field is set to 3, the port monitor understands messages of classes 1, 2, and 3. Port monitors may not generate messages; they may only respond to messages. A port monitor's response must be of the same class as the originating message. Since only the SAC can generate messages, this protocol will function even if the port monitor is capable of dealing with messages of a higher class than the SAC can generate. `pm_size` (an element of the pmmsg structure) and `sc_size` (an element of the `sacmsg` structure) indicate the size of the optional data part of the message. The format of this part of the message is undefined. Its definition is inherent in the type of message. For Solaris, always set both `sc_size` and `pm_size` to `0`.

Administrative Interface This section discusses the port monitor administrative files available under the SAC.

The SAC Administrative File _sactab The service access controller's administrative file contains information about all the port monitors for which the SAC is responsible. This file exists on the delivered system. Initially, it is empty except for a single comment line that contains the version number of the SAC. Port monitors are added to the system by making entries in the SAC's administrative file. These entries should be made using the administrative command `sacadm(1M)` with a `-a` option. `sacadm(1M)` is also used to remove entries from the SAC's administrative file. Each entry in the SAC's administrative file contains the following information.

`PMTAG` A unique tag that identifies a particular port monitor. The system administrator is responsible for naming a port monitor. This tag is then used by the SAC to identify the port monitor for all administrative purposes. PMTAG may consist of up to 14 alphanumeric characters.

`PMTYPE` The type of the port monitor. In addition to its unique tag, each port monitor has a type designator. The type designator identifies a group of port monitors that are different invocations of the same entity. ttymon and listen are examples of valid port monitor types. The type designator is used to facilitate the administration of

groups of related port monitors. Without a type designator, the system administrator has no way of knowing which port monitor tags correspond to port monitors of the same type. PMTYPE may consist of up to 14 alphanumeric characters.

`FLGS` The flags that are currently defined are:

d When started, do not enable the port monitor.

x Do not start the port monitor.

If no flag is specified, the default action is taken. By default a port monitor is started and enabled.

`RCNT` The number of times a port monitor may fail before being placed in a failed state. Once a port monitor enters the failed state, the SAC will not try to restart it. If a count is not specified when the entry is created, this field is set to 0. A restart count of 0 indicates that the port monitor is not to be restarted when it fails.

`COMMAND` A string representing the command that will start the port monitor. The first component of the string, the command itself, must be a full path name.

The Port Monitor Administrative File _pmtab

Each port monitor will have two directories for its exclusive use. The current directory will contain files defined by the SAF (`_pmtab`, `_pid`) and the per-service configuration scripts, if they exist. The directory `/var/saf/`*pmtag*, where *pmtag* is the tag of the port monitor, is available for the port monitor's private files. Each port monitor has its own administrative file. The `pmadm(1M)` command should be used to add, remove, or modify service entries in this file. Each time a change is made using `pmadm(1M)`, the corresponding port monitor rereads its administrative file. Each entry in a port monitor's administrative file defines how the port monitor treats a specific port and what service is to be invoked on that port. Some fields must be present for all types of port monitors. Each entry must include a service tag to identify the service uniquely and an identity to be assigned to the service when it is started (for example, root).

The combination of a service tag and a port monitor tag uniquely define an instance of a service. The same service tag may be used to identify a service under a different port monitor. The record must also contain port monitor specific data (for example, for a ttymon port monitor, this will include the prompt string which is meaningful to ttymon). Each type of port monitor must provide a command that takes the necessary port monitor-specific data as arguments and outputs these data in a form suitable for storage in the file. The `ttyadm(1M)` command does this for ttymon and `nlsadmin(1M)` does it for listen. For a user-defined port monitor, a similar administrative command must also be supplied. Each service entry in the port monitor administrative file must have the following format and contain the information listed below:

```
svctag:flgs:id:reserved:reserved:reserved:pmspecific# comment
```

SVCTAG is a unique tag that identifies a service. This tag is unique only for the port monitor through which the service is available. Other port monitors may offer the same or other services with the same tag. A service requires both a port monitor tag and a service tag to identify it uniquely. SVCTAG may consist of up to 14 alphanumeric characters. The service entries are defined as:

`FLGS` Flags with the following meanings may currently be included in this field:

- x Do not enable this port. By default the port is enabled.
- u Create a utmpx entry for this service. By default no utmpx entry is created for the service.

`ID` The identity under which the service is to be started. The identity has the form of a login name as it appears in `/etc/passwd`.

`PMSPECIFIC` Examples of port monitor information are addresses, the name of a process to execute, or the name of a STREAMS-based pipe to pass a connection through. This information will vary to meet the needs of each different type of port monitor.

`COMMENT` A comment associated with the service entry. Port monitors may ignore the *u* flag if creating a utmpx entry for the service is not appropriate to the manner in which the service is to be invoked. Some services may not start properly unless utmpx entries have been created for them (for example, login). Each port monitor administrative file must contain one special comment of the form:

VERSION=value

where *value* is an integer that represents the port monitor's version number. The version number defines the format of the port monitor administrative file. This comment line is created automatically when a port monitor is added to the system. It appears on a line by itself, before the service entries.

Monitor-Specific Administrative Command

Previously, two pieces of information included in the `_pmtab` file were described: the port monitor's version number and the port monitor part of the service entries in the port monitor's `_pmtab` file. When a new port monitor is added, the version number must be known so that the `_pmtab` file can be correctly initialized. When a new service is added, the port monitor part of the `_pmtab` entry must be formatted correctly. Each port monitor must have an administrative command to perform these two tasks. The person who defines the port monitor must also define such an administrative command and its input options. When the command is invoked with these options, the information required for the port monitor part of the service entry must be correctly formatted for inclusion in the port monitor's `_pmtab` file and must be written to the standard output. To request the version number the command must be invoked with a `-V` option; when it is invoked in this way, the port monitor's current

version number must be written to the standard output. If the command fails for any reason during the execution of either of these tasks, no data should be written to standard output.

The Port Monitor/Service Interface

The interface between a port monitor and a service is determined solely by the service. Two mechanisms for invoking a service are presented here as examples.

New Service Invocations

The first interface is for services that are started anew with each request. This interface requires the port monitor to first `fork(2)` a child process. The child will eventually become the designated service by performing an `exec(1)`. Before the `exec(1)` happens, the port monitor may take some port monitor-specific action; however, one action that must occur is the interpretation of the per-service configuration script, if one is present. This is done by calling the library routine `doconfig(3NSL)`.

Standing Service Invocations

The second interface is for invocations of services that are actively running. To use this interface, a service must have one end of a stream pipe open and be prepared to receive connections through it.

Port Monitor Requirements

To implement a port monitor, several generic requirements must be met. This section summarizes these requirements. In addition to the port monitor itself, an administrative command must be supplied.

Initial Environment

When a port monitor is started, it expects an initial execution environment in which:

- It has no file descriptors open
- It cannot be a process group leader
- It has an entry in `/etc/utmpx` of type LOGIN_PROCESS
- An environment variable, ISTATE, is set to “enabled” or “disabled” to indicate the port monitor's correct initial state
- An environment variable, PMTAG, is set to the port monitor's assigned tag
- The directory that contains the port monitor's administrative files is its current directory
- pThe port monitor is able to create private files in the directory `/var/saf/`*tag*, where *tag* is the port monitor's tag
- The port monitor is running with user id 0 (root)

Important Files

Relative to its current directory, the following key files exist for a port monitor.

`_config`	The port monitor's configuration script. The port monitor configuration script is run by the SAC. The SAC is started by `init(1M)` as a result of an entry in `/etc/inittab` that calls `sac(1M)`.
`_pid`	The file into which the port monitor writes its process id.
`_pmtab`	The port monitor's administrative file. This file contains information about the ports and services for which the port monitor is responsible.
`_pmpipe`	The FIFO through which the port monitor will receive messages from the SAC.
`svctag`	The per-service configuration script for the service with the tag *svctag*.
`../_sacpipe`	The FIFO through which the port monitor will send messages to `sac(1M)`.

Port Monitor Responsibilities

A port monitor is responsible for performing the following tasks in addition to its port monitor function:

Write its process id into the file `_pid` and place an advisory lock on the file
Terminate gracefully on receipt of the signal SIGTERM
Follow the protocol for message exchange with the SAC

A port monitor must perform the following tasks during service invocation:

Create a `utmpx` entry if the requested service has the `u` flag set in `_pmtab`

Port monitors may ignore this flag if creating a `utmpx` entry for the service does not make sense because of the manner in which the service is to be invoked. On the other hand, some services may not start properly unless utmpx entries have been created for them.

Interpret the per-service configuration script for the requested service, if it exists, by calling the `doconfig(3NSL)` library routine

Configuration Files and Scripts

The library routine `doconfig(3NSL)`, defined in `libnsl.so`, interprets the configuration scripts contained in the files `/etc/saf/_sysconfig` (the per-system configuration file), and `/etc/saf/`*pmtag*`/_config` (per-port monitor configuration files); and in `/etc/saf/`*pmtag*`/svctag` (per-service configuration files). Its syntax is:

```
#include <sac.h>
    int doconfig (int fd, char *script, long rflag);
```

`script` is the name of the configuration script; *fd* is a file descriptor that designates the stream to which stream manipulation operations are to be applied; *rflag* is a bitmask that indicates the mode in which script is to be interpreted. *rflag* may take two values, NORUN and NOASSIGN,

which may be or'd. If *rflag* is zero, all commands in the configuration script are eligible to be interpreted. If *rflag* has the NOASSIGN bit set, the assign command is considered illegal and will generate an error return. If *rflag* has the NORUN bit set, the run and runwait commands are considered illegal and will generate error returns. If a command in the script fails, the interpretation of the script ceases at that point and a positive integer is returned; this number indicates which line in the script failed. If a system error occurs, a value of −1 is returned. If a script fails, the process whose environment was being established should not be started. In the example, `doconfig(3NSL)` is used to interpret a per-service configuration script.

```
 . . .
        if ((i = doconfig (fd, svctag, 0)) != 0){
        error ("doconfig failed on line %d of script %s",i,svctag);
    }
```

The Per-System Configuration File

The per-system configuration file, `/etc/saf/_sysconfig`, is delivered empty. It may be used to customize the environment for all services on the system by writing a command script in the interpreted language described in this chapter and on the `doconfig(3NSL)` manpage. When the SAC is started, it calls the `doconfig(3NSL)` function to interpret the per-system configuration script. The SAC is started when the system enters multiuser mode.

Per-Port Monitor Configuration Files

Per-port monitor configuration scripts (`/etc/saf/`*pmtag*`/_config`) are optional. They allow the user to customize the environment for any given port monitor and for the services that are available through the ports for which that port monitor is responsible. Per-port monitor configuration scripts are written in the same language used for per-system configuration scripts. The per-port monitor configuration script is interpreted when the port monitor is started. The port monitor is started by the SAC after the SAC has itself been started and after it has run its own configuration script, `/etc/saf/_sysconfig`. The per-port monitor configuration script may override defaults provided by the per-system configuration script.

Per-Service Configuration Files

Per-service configuration files allow the user to customize the environment for a specific service. For example, a service may require special privileges that are not available to the general user.

Using the language described in the doconfig(3NSL) manpage, you can write a script that will grant or limit such special privileges to a particular service offered through a particular port monitor. The per-service configuration may override defaults provided by higher-level configuration scripts. For example, the per-service configuration script may specify a set of STREAMS modules other than the default set.

The Configuration Language

The language in which configuration scripts are written consists of a sequence of commands, each of which is interpreted separately. The following reserved keywords are defined: `assign`, `push`, `pop`, `runwait`, and `run`. The comment character is `#`. Blank lines are not significant. No line in a command script may exceed 1024 characters.

`assign` *variable*=*value*
: Used to define environment variables; *variable* is the name of the environment variable and *value* is the value to be assigned to it. The value assigned must be a string constant; no form of parameter substitution is available. *value* may be quoted. The quoting rules are those used by the shell for defining environment variables. `assign` will fail if space cannot be allocated for the new variable or if any part of the specification is invalid.

`push` *module1*[,*module2, module3*, . . .]
: Used to push STREAMS modules onto the stream designated by `fd`; *module1* is the name of the first module to be pushed, *module2* is the name of the second module to be pushed, and so on. The command will fail if any of the named modules cannot be pushed. If a module cannot be pushed, the subsequent modules on the same command line will be ignored and modules that have already been pushed will be popped.

`pop` [*module*]
: Used to pop STREAMS modules off the designated stream. If `pop` is invoked with no arguments, the top module on the stream is popped. If an argument is given, modules will be popped one at a time until the named module is at the top of the stream. If the named module is not on the designated stream, the stream is left as it was and the command fails. If *module* is the special

keyword `ALL`, then all modules on the stream will be popped. Only modules above the topmost driver are affected.

`runwait command`

The `runwait` command runs a command and waits for it to complete; `command` is the path name of the command to be run. The command is run with `/bin/sh -c` prepended to it; shell scripts may thus be executed from configuration scripts. The `runwait` command will fail if command cannot be found or cannot be executed, or if `command` exits with a nonzero status.

`run command`

The `run` command is identical to `runwait` except that it does not wait for command to complete; `command` is the path name of the command to be run. `run` will not fail unless it is unable to create achild process to execute the command. Although they are syntactically indistinguishable, some of the commands available to `run` and `runwait` are interpreter built-in commands. Interpreter built-ins are used when it is necessary to alter the state of a process within the context of that process. The `doconfig` interpreter built-in commands are similar to the shell special commands and, like these, they do not spawn another process for execution. See the sh(1) man page. The initial set of built-in commands is: `cd`, `ulimit`, `umask`.

Sample Port Monitor Code

This example shows an example of a "null" port monitor that simply responds to messages from the SAC.

```
# include <stdlib.h>
# include <stdio.h>
# include <unistd.h>
# include <fcntl.h>
# include <signal.h>
# include <sac.h>

char Scratch[BUFSIZ]; /* scratch buffer */
char Tag[PMTAGSIZE + 1]; /* port monitor's tag */
FILE *Fp; /* file pointer for log file */
FILE *Tfp; /* file pointer for pid file */
char State; /* portmonitor's current state*/

main(argc, argv)
```

```
    int argc;
    char *argv[];
{
    char *istate;
    strcpy(Tag, getenv("PMTAG"));
/*
 * open up a log file in port monitor's private directory
 */
    sprintf(Scratch, "/var/saf/%s/log", Tag);
    Fp = fopen(Scratch, "a+");
    if (Fp == (FILE *)NULL)
        exit(1);
    log(Fp, "starting");
/*
 * retrieve initial state (either "enabled" or "disabled") and set
 * State accordingly
 */
    istate = getenv("ISTATE");
    sprintf(Scratch, "ISTATE is %s", istate);
    log(Fp, Scratch);
    if (!strcmp(istate, "enabled"))
        State = PM_ENABLED;
    else if (!strcmp(istate, "disabled"))
        State = PM_DISABLED;
    else {
        log(Fp, "invalid initial state");
        exit(1);
    }
    sprintf(Scratch, "PMTAG is %s", Tag);
    log(Fp, Scratch);
/*
 * set up pid file and lock it to indicate that we are active
 */
    Tfp = fopen("_pid", "w");
    if (Tfp == (FILE *)NULL) {
        log(Fp, "couldn't open pid file");
        exit(1);
    }
    if (lockf(fileno(Tfp), F_TEST, 0) < 0) {
        log(Fp, "pid file already locked");
        exit(1);
    }

    log(Fp, "locking file");
    if (lockf(fileno(Tfp), F_LOCK, 0) < 0) {
        log(Fp, "lock failed");
        exit(1);
```

```
    }
    fprintf(Tfp, "%d", getpid());
    fflush(Tfp);

/*
 * handle poll messages from the sac ... this function never returns
 */
    handlepoll();
    pause();
    fclose(Tfp);
    fclose(Fp);
}

handlepoll()
{
    int pfd; /* file descriptor for incoming pipe */
    int sfd; /* file descriptor for outgoing pipe */
    struct sacmsg sacmsg; /* incoming message */
    struct pmmsg pmmsg; /* outgoing message */
/*
 * open pipe for incoming messages from the sac
 */
    pfd = open("_pmpipe", O_RDONLY|O_NONBLOCK);
    if (pfd < 0) {
        log(Fp, "_pmpipe open failed");
        exit(1);
    }
/*
 * open pipe for outgoing messages to the sac
 */
    sfd = open("../_sacpipe", O_WRONLY);
    if (sfd < 0) {
        log(Fp, "_sacpipe open failed");
        exit(1);
    }
/*
 * start to build a return message; we only support class 1 messages
 */
    strcpy(pmmsg.pm_tag, Tag);
    pmmsg.pm_size = 0;
    pmmsg.pm_maxclass = 1;
/*
 * keep responding to messages from the sac
 */
     for (;;) {
         if (read(pfd, &sacmsg, sizeof(sacmsg)) != sizeof(sacmsg)) {
             log(Fp, "_pmpipe read failed");
```

```
                exit(1);
            }
    /*
     * determine the message type and respond appropriately
     */
            switch (sacmsg.sc_type) {
                case SC_STATUS:
                    log(Fp, "Got SC_STATUS message");
                    pmmsg.pm_type = PM_STATUS;
                    pmmsg.pm_state = State;
                    break;
                case SC_ENABLE:
                    /*note internal state change below*/
                    log(Fp, "Got SC_ENABLE message");
                    pmmsg.pm_type = PM_STATUS;
                    State = PM_ENABLED;
                    pmmsg.pm_state = State;
                    break;
                case SC_DISABLE:
                    /*note internal state change below*/
                    log(Fp, "Got SC_DISABLE message");
                    pmmsg.pm_type = PM_STATUS;
                    State = PM_DISABLED;
                    pmmsg.pm_state = State;
                    break;
                case SC_READDB:
                    /*
                    * if this were a fully functional port
                    * monitor it would read _pmtab here
                    * and take appropriate action
                    */
                    log(Fp, "Got SC_READDB message");
                    pmmsg.pm_type = PM_STATUS;
                    pmmsg.pm_state = State;
                    break;
                default:
                    sprintf(Scratch, "Got unknown message <%d>",
                    sacmsg.sc_type);
                    log(Fp, Scratch);
                    pmmsg.pm_type = PM_UNKNOWN;
                    pmmsg.pm_state = State;
                    break;
            }
    /*
     * send back a response to the poll
     * indicating current state
     */
```

```
            if (write(sfd, &pmmsg, sizeof(pmmsg)) != sizeof(pmmsg))
                log(Fp, "sanity response failed");
        }
}
/*
 * general logging function
 */
log(fp, msg)
    FILE *fp;
    char *msg;
{
    fprintf(fp, "%d; %s\n", getpid(), msg);
    fflush(fp);
}
```

The sac.h Header File

The following example shows the sac.h header file.

```
/* length in bytes of a utmpx id */
# define IDLEN 4
/* wild character for utmpx ids */
# define SC_WILDC 0xff
/* max len in bytes for port monitor tag */
# define PMTAGSIZE 14
/*
 * values for rflag in doconfig()
 */
/* don't allow assign operations */
# define NOASSIGN 0x1
/* don't allow run or runwait operations */
# define NORUN 0x2
/*
 * message to SAC (header only). This header is forever fixed. The
 * size field (pm_size) defines the size of the data portion of the
 * message, which follows the header. The form of this optional data
 * portion is defined strictly by the message type (pm_type).
 */
struct pmmsg {
    char pm_type;               /* type of message */
    unchar_t pm_state;            /* current state of pm */
    char pm_maxclass;           /* max message class this port monitor
                                    understands */
    char pm_tag[PMTAGSIZE + 1]; /* pm's tag */
    int pm_size;                /* size of opt data portion */
};
/*
 * pm_type values
 */
# define PM_STATUS 1 /* status response */
```

```
# define PM_UNKNOWN 2 /* unknown message was received */
/*
 * pm_state values
 */
/*
 * Class 1 responses
 */
# define PM_STARTING 1   /* monitor in starting state */
# define PM_ENABLED 2    /* monitor in enabled state */
# define PM_DISABLED 3   /* monitor in disabled state */
# define PM_STOPPING 4   /* monitor in stopping state */
/*
 * message to port monitor
 */
struct sacmsg {
    int sc_size;         /* size of optional data portion */
    char sc_type;        /* type of message */
};
/*
 * sc_type values
 * These represent commands that the SAC sends to a port monitor.
 * These commands are divided into "classes" for extensibility. Each
 * subsequent "class" is a superset of the previous "classes" plus
 * the new commands defined within that "class". The header for all
 * commands is identical; however, a command may be defined such that
 * an optional data portion may be sent in addition to the header.
 * The format of this optional data piece is self-defining based on
 * the command. The first message sent by the SAC
 * will always be a class 1 message. The port monitor response
 * indicates the maximum class that it is able to understand. Another
 * note is that port monitors should only respond to a message with
 * an equivalent class response (i.e. a class 1 command causes a
 * class 1 response).
 */
/*
 * Class 1 commands (currently, there are only class 1 commands)
 */
# define SC_STATUS 1    /* status request *
# define SC_ENABLE 2    /* enable request */
# define SC_DISABLE 3   /* disable request */
# define SC_READDB 4    /* read pmtab request */
/*
 * ‘errno’ values for Saferrno, note that Saferrno is used by both
 * pmadm and sacadm and these values are shared between them
 */
# define E_BADARGS 1   /* bad args/ill-formed cmd line */
# define E_NOPRIV 2    /* user not priv for operation */
```

```
# define E_SAFERR 3    /* generic SAF error */
# define E_SYSERR 4    /* system error */
# define E_NOEXIST 5   /* invalid specification */
# define E_DUP 6       /* entry already exists */
# define E_PMRUN 7     /* port monitor is running */
# define E_PMNOTRUN 8  /* port monitor is not running */
# define E_RECOVER 9
   /* in recovery */
```

Directory Structure

This section gives a description of the SAF files and directories.

`/etc/saf/_sysconfig`	The per-system configuration script.
`/etc/saf/_sactab`	The SAC's administrative file. Contains information about the port monitors for which the SAC is responsible.
`/etc/saf/`*pmtag*	The home directory for port monitor *pmtag*.
`/etc/saf/`*pmtag*`/_config`	The per-port monitor configuration script for port monitor pmtag.
`/etc/saf/`*pmtag*`/_pmtab`	Port monitor pmtag's administrative file. Contains information about the services for which *pmtag* is responsible.
`/etc/saf/`*pmtag*`/`*svctag*	The file in which the per-service configuration script for service *svctag* (available through port monitor `pmtag`) is placed.
`/etc/saf/`*pmtag*`/_pid`	The file in which a port monitor writes its process id in the current directory and places an advisory lock on the file.
`/etc/saf/ pmtag /_pmpipe`	The file in which the port monitor receives messages from the SAC and `../_sacpipe` and sends return messages to the SAC.
`/var/saf/_log`	The SAC's log file.
`/var/saf/`*pmtag*	The directory for files created by port monitor *pmtag*, for example its log file.

List Of Commands

The following administrative commands relate to SAF.

`sacadm(1M)`	port monitor administrative command
`pmadm(1M)`	service administration command

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

See Also exec(1), sh(1), init(1M), nlsadmin(1M), pmadm(1M), sac(1M), sacadm(1M), ttyadm(1M), fork(2), doconfig(3NSL), attributes(5)

Name sar, sa1, sa2, sadc – system activity report package

Synopsis

```
/usr/lib/sa/sadc [t n] [ofile]

/usr/lib/sa/sa1 [t n]

/usr/lib/sa/sa2 [-aAbcdgkmpqruvwy] [-e time] [-f filename]
     [-i sec] [-s time]
```

Description System activity data can be accessed at the special request of a user (see `sar(1)`) and automatically, on a routine basis, as described here. The operating system contains several counters that are incremented as various system actions occur. These include counters for CPU utilization, buffer usage, disk and tape I/O activity, TTY device activity, switching and system-call activity, file-access, queue activity, inter-process communications, and paging. For more general system statistics, use `iostat(1M)`, `sar(1)`, or `vmstat(1M)`.

`sadc` and two shell procedures, `sa1` and `sa2`, are used to sample, save, and process this data.

`sadc`, the data collector, samples system data *n* times, with an interval of *t* seconds between samples, and writes in binary format to *ofile* or to standard output. The sampling interval *t* should be greater than 5 seconds; otherwise, the activity of `sadc` itself may affect the sample. If *t* and *n* are omitted, a special record is written. This facility can be used at system boot time, when booting to a multi-user state, to mark the time at which the counters restart from zero. For example, when accounting is enabled, the `svc:/system/sar:default` service writes the restart mark to the daily data file using the command entry:

```
su sys -c "/usr/lib/sa/sadc /var/adm/sa/sa'date +%d'"
```

The shell script `sa1`, a variant of `sadc`, is used to collect and store data in the binary file `/var/adm/sa/sa`*dd*, where `dd` is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The following entries in `/var/spool/cron/crontabs/sys` will produce records every 20 minutes during working hours and hourly otherwise:

```
0 * * * 0-6 /usr/lib/sa/sa1
20,40 8–17 * * 1–5 /usr/lib/sa/sa1
```

See `crontab(1)` for details.

The shell script `sa2`, a variant of `sar`, writes a daily report in the file `/var/adm/sa/sar`*dd*. See the `OPTIONS` section in `sar(1)` for an explanation of the various options. The following entry in `/var/spool/cron/crontabs/sys` will report important activities hourly during the working day:

```
5 18 * * 1–5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 1200 -A
```

Files

`/tmp/sa.adrfl`	address file
`/var/adm/sa/sadd`	Daily data file
`/var/adm/sa/sardd`	Daily report file

```
/var/spool/cron/crontabs/sys
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWaccu

See Also crontab(1), sag(1), sar(1), svcs(1), timex(1), iostat(1M), svcadm(1M), vmstat(1M), attributes(5), smf(5)

System Administration Guide: Advanced Administration

Notes The `sar` service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/sar
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name savecore – save a crash dump of the operating system

Synopsis `/usr/bin/savecore [-Lvd] [-f` *dumpfile*`] [`*directory*`]`

Description The `savecore` utility saves a crash dump of the kernel (assuming that one was made) and writes a reboot message in the shutdown log. It is invoked by the `dumpadm` service each time the system boots.

`savecore` can be configured by `dumpadm(1M)` to save crash dump data in either a compressed or uncompressed format. For the compressed format, `savecore` saves the crash dump data in the file *directory*`/vmcore.`*N*`.z`, where *N* in the pathname is replaced by a number which increments by one each time `savecore` is run in *directory*. The compressed file can be uncompressed in a separate step using the `-f` *dumpfile* option. For the uncompressed format, `savecore` saves the crash dump data in the file *directory*`/vmcore.`*N* and the kernel's namelist in *directory*`/unix.`*N*.

Before writing out a crash dump, `savecore` reads a number from the file *directory*`/minfree`. This is the minimum number of kilobytes that must remain free on the file system containing *directory*. If after saving the crash dump the file system containing *directory* would have less free space the number of kilobytes specified in `minfree`, the crash dump is not saved. if the `minfree` file does not exist, `savecore` assumes a `minfree` value of 1 megabyte.

The `savecore` utility also logs a reboot message using facility `LOG_AUTH` (see `syslog(3C)`). If the system crashed as a result of a panic, `savecore` logs the panic string too.

Options The following options are supported:

`-d` Disregard dump header valid flag. Force `savecore` to attempt to save a crash dump even if the header information stored on the dump device indicates the dump has already been saved.

`-f` *dumpfile* Save a crash dump from the specified file instead of from the system's current dump device. When given *directory*`/vmcore.`*N*`.z`, uncompress the file to `vmcore.`*N* and `unix.`*N*, where *N* is the same number as in the compressed name.

This option may also be useful if the information stored on the dump device has been copied to an on-disk file by means of the `dd(1M)` command.

`-L` Save a crash dump of the live running Solaris system, without actually rebooting or altering the system in any way. This option forces `savecore` to save a live snapshot of the system to the dump device, and then immediately to retrieve the data and to write it out to a new set of crash dump files in the specified directory. Live system crash dumps can only be performed if you have configured your system to have a dedicated dump device using `dumpadm(1M)`.

`savecore -L` does not suspend the system, so the contents of memory continue to change while the dump is saved. This means that live crash dumps are not fully self-consistent.

`-v` Verbose. Enables verbose error messages from `savecore`.

Operands The following operands are supported:

directory Save the crash dump files to the specified directory. If *directory* is not specified, `savecore` saves the crash dump files to the default `savecore` *directory*, configured by `dumpadm(1M)`.

Files

- *directory*`/vmcore.`*N*
- *directory*`/unix.`*N*
- *directory*`/bounds`
- *directory*`/minfree`
- `/var/crash/`‘`uname -n`‘ (default crash dump directory)

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs

See Also `adb(1)`, `mdb(1)`, `svcs(1)`, `dd(1M)`, `dumpadm(1M)`, `svcadm(1M)`, `syslog(3C)`, `attributes(5)`, `smf(5)`

Notes The system crash dump service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/dumpadm:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

If the dump device is also being used as a swap device, you must run `savecore` very soon after booting, before the swap space containing the crash dump is overwritten by programs currently running.

Name scadm – administer System Controller (SC)

Synopsis

```
/usr/platform/platform-name/sbin/scadm subcommand [option]
     [argument]...
```

Description The `scadm` utility administers the System Controller (SC). This utility allows the host server to interact with the SC.

The `scadm` utility *must* be run as root.

The interface, output, and location in the directory hierarchy for `scadm` are uncommitted and might change.

platform-name is the name of the platform implementation. Use the `uname -i` command to identify the platform implementation. See `uname(1)`.

The `scadm` utility has a number of subcommands. Some subcommands have specific options and arguments associated with them. See `SUBCOMMANDS`, `OPTIONS`, `OPERANDS`, and `USAGE`.

SUBCOMMANDS Subcommands immediately follow the `scadm` command on the command line, and are separated from the command by a SPACE.

The following subcommands are supported

consolehistory Display the SC's console log. The SC maintains a running log which captures all console output. This log is maintained as a first-in, first-out buffer: New console output may displace old console output if the buffer is full. By default, only the last eight kilobytes of the console log file are displayed.

The optional `-a` argument specifies that the entire console log file be displayed.

It is possible for the SC to fill this log more quickly than the `consolehistory` subcommand can read it. This means that it is possible for some log data to be lost before it can be displayed. If this happens, the `consolehistory` subcommand displays "scadm: lost <number> bytes of console log data" in the log output, to indicate that some data was lost.

The format for the `consolehistory` subcommand is:

```
scadm consolehistory [-a]
```

The `consolehistory` subcommand is not available on all platforms. If this command is used on a platform that does not support it, `scadm` prints this message:

```
scadm: command/option not supported
```

and exit with non-zero status.

date — Display the SC's time and date

The format for the `date` subcommand is:

```
scadm date
```

download — Program the SC's firmware.

There are two parts to the firmware, the boot monitor and the main image.

By default, The `scadm` command's download programs the main firmware image. The `boot` argument selects programming of the boot monitor.

The format for the `download` subcommand is:

```
scadm download [boot] file
```

fruhistory — Display the contents of the "field replacable unit" log maintained by the SC. By default, only the last eight kilobytes of the fru history log file are displayed. The data in contained this log contains snapshots of the SC's "showfru" command, taken whenever the system is reset, or a hot-plug event is detected by the SC.

The optional `-a` argument specifies that the entire fru log file be displayed.

It is possible for the SC to fill this log more quickly than the `fruhistory` subcommand can read it. This means that it is possible for some log data to be lost before it can be displayed. If this happens, the `fruhistory` subcommand displays "scadm: lost <number> bytes of fru log data" in the log output, to indicate that some data was lost.

The format for the fruhistory subcommand is:

```
scadm fruhistory [-a]
```

The `fruhistory` subcommand is not available on all platforms. If this command is used on a platform which does not support it, `scadm` prints this message:

```
scadm: command/option not supported
```

and exit with non-zero status.

help — Display a list of commands.

The format for the `help` subcommand is:

```
scadm help
```

loghistory — Display the most recent entries in the SC event log. The optional `-a` argument causes the entire event log history to be displayed. The `-a`

argument is available only on platforms which support large log files. On platforms which do not support large log files, this flag has no additional effect.

It is possible for the SC to fill this log more quickly than the `loghistory` subcommand can read it. This means that it is possible for some log data to be lost before it can be displayed. If this happens, the `loghistory` subcommand displays "scadm: lost <number> events" in the log output, to indicate that some data was lost.

The format for the `loghistory` subcommand is:

```
scadm loghistory [-a]
```

resetrsc

Reset the SC. There are two types of resets allowed, a `hard` reset and a `soft` reset.The `hard` reset is done by default. The `soft` reset can be selected by using the `-s` option.

The format for the `resetrsc` subcommand is:

```
scadm resetrsc [-s]
```

send_event

Manually send a text based event. The SC can forward the event to the SC event log. You can configure the `-c` option to send a critical warning to email, alert to logged in SC users, and `syslog`. Critical events are logged to `syslog(3C)`. There is an `80` character limit to the length of the associated text message.

The format for the `send_event` subcommand is:

```
scadm send_event [-c] "message"
```

set

Set SC configuration variables to a value.

Examples of SC configuration variables include: SC IP address `netsc_ipaddr` and SC Customer Information `sc_customerinfo`. See the output from the `scadm help` command for a complete list of SC configuration variables.

The format for the `set` subcommand is:

```
scadm set variable value
```

show

Display the current SC configuration variable settings. If no variable is specified, `scadm` shows all variable settings.

The format for the `show` subcommand is:

```
scadm show [variable]
```

shownetwork

Display the current network configuration parameters for SC.

The format for the `shownetwork` subcommand is:

```
scadm shownetwork
```

useradd — Add user accounts to the SC. The SC supports up to sixteen separate users.

The format for the `useradd` subcommand is:

```
scadm useradd username
```

userdel — Delete a user account from SC.

The format for the `userdel` subcommand is:

```
scadm userdel username
```

userpassword — Set a password for the user account specified. This password overrides any existing password currently set. There is no verification of the old password before setting the new password.

The format for the `userpassword` subcommand is:

```
scadm userpassword username
```

userperm — Set the permission level for the user.

The format for the `userperm` subcommand is:

```
scadm userperm username [aucr]
```

usershow — Display details on the specified user account. If a username is not specified, all user accounts are displayed.

The format for the `usershow` subcommand is:

```
scadm usershow username
```

version — Display the version numbers of the SC and its components.

The format for the `version` subcommand is:

```
scadm version [-v]
```

Options The `resetrsc`, `send_event`, and `version` subcommands have associated options. Options follow subcommands on the command line and are separated from the subcommand by a SPACE.

The `resetrsc` subcommand supports the following options:

`-s` Perform a soft reset instead of a hard reset. A hard reset physically resets the SC hardware. The SC software jumps to the boot firmware, simulating a reset, for a soft reset.

The `send_event` subcommand supports the following options:

`-c` Send a critical event. Without the `-c`, `-send_event` sends a warning.

The `version` subcommand supports the following options:

`-v` Display a verbose output of version numbers and associated information.

The `consolehistory`, `fruhistory`, and `loghistory` subcommands support the following option:

`-a` Display the entire log. These subcommands normally display only the most recent log data. This flag causes them to display the entire log.

Operands The `download`, `send_event`, `set`, `show`, `useradd`, `userdel`, `userperm`, `usershow`, `userpassword`, and `userperm` subcommands have associated arguments (operands).

If the subcommand has an option, the arguments follow the option on the command line and is separated from the option by a SPACE. If the subcommand does not have an option, the arguments follow the subcommand on the command line and are separated from the subcommand by a SPACE. If there are more than one arguments, they are separated from each other by a SPACE.

The `download` subcommand supports the following arguments:

boot Program the boot monitor portion of the flash. The main portion of the flash is programmed without any arguments

file Specify *file* as the path to where the boot or main firmware image resides for download.

Examples of *file* are:

`/usr/platform/`*platform_type*`/lib/image/alommainfw`

or

`/usr/platform/`*platform_type*`/lib/image/alombootfw`

The `send_event` subcommand supports the following arguments:

"*message*" Describe event using the test contained in *message*. Enclose *message* in quotation marks.

The `set` subcommand supports the following arguments:

variable Set SC configuration *variable*.

value Set SC configuration variable to *value*.

The `show` subcommand supports the following arguments:

variable Display the value of that particular variable.

The `useradd` subcommand supports the following arguments:

username Add new SC account *username*.

The `userdel` subcommand supports the following arguments:

username Remove SC account *username*.

The `userperm` subcommand supports the following arguments:

`-aucr` Set permissions for SC user accounts. If no permissions are specified, all four permissions are disabled and read only access is assigned.

The following are the definitions for permissions:

a Allow user to administer or change the SC configuration variables

u Allow user to use the user commands to modify SC accounts

c Allow user to connect to console.

r Allow user to reset SC and to power on and off the host.

username Change permissions on SC account *username*.

The `-usershow` subcommand supports the following arguments:

username Display information on SC account *username*. If *username*is not specified, all accounts are displayed.

The `userpassword` subcommand supports the following arguments:

username Set SC password for *username*.

The `userperm` subcommand supports the following arguments:

username Change SC permissions for *username*.

Examples

EXAMPLE 1 Displaying the SC's Date and Time

The following command displays the SC's date and time.

```
scadm date
```

EXAMPLE 2 Setting the SC's Configuration Variables

The following command sets the SC's configuration variable `netsc_ipaddr` to `192.168.1.2`:

```
scadm set netsc_ipaddr 192.168.1.2
```

EXAMPLE 3 Displaying the Current SC's Configuration Settings:

The following command displays the current SC configuration settings:

```
scadm show
```

EXAMPLE 4 Displaying the Current Settings for a Variable

The following command displays the current settings for the `sys_hostname` variable:

```
scadm show sys_hostname
```

EXAMPLE 5 Sending a Text-Based Critical Event

The following command sends a critical event to the SC logs, alerts the current SC users, and sends an event to `syslog(3C)`:

```
scadm send_event -c "The UPS signaled a loss in power"
```

EXAMPLE 6 Sending an Informational Text-Based Event

The following command sends an non-critical informational text based event to the SC event log:

```
scadm send_event "The disk is close to full capacity"
```

EXAMPLE 7 Adding a User To the SC

The following command adds user `rscroot` to the SC:

```
scadm useradd rscroot
```

EXAMPLE 8 Deleting a User From the SC

The following command deletes user `olduser` from the SC:

```
scadm userdel olduser
```

EXAMPLE 9 Displaying User Details

The following command displays details of all user accounts:

```
scadm usershow
```

EXAMPLE 10 Displaying Details for a Specific User

The following command displays details of user account `rscroot`:

```
scadm usershow rscroot
```

EXAMPLE 11 Setting the User Permission Level

The following command sets the full permission level for user `rscroot` to `aucr`:

```
scadm userperm rscroot aucr
```

EXAMPLE 12 Setting the User Permission Level

The following command sets only console access for user `newuser` to `c`:

```
scadm userperm newuser c
```

EXAMPLE 13 Setting the User Permission Level

The following command sets the permission level for user `newuser` to read only access:

```
scadm userperm newuser
```

EXAMPLE 14 Displaying the Current Network Parameters

The following command displays the current network configuation parameters for the SC:

```
scadm shownetwork
```

EXAMPLE 15 Viewing the Consolehistory

The following command displays the the content console in the SC event log:

```
scadm consolehistory [-a]
```

EXAMPLE 16 Viewing the Fruhistory

The following command displays the content of the "field replacable unit" in the SC event log:

```
scadm fruhistory [-a]
```

EXAMPLE 17 Viewing the Loghistory

The following command displays the most recent entries in the SC event log:

```
scadm loghistory [-a]
```

EXAMPLE 18 Displaying Verbose Information

The following command displays verbose version information on the SC and its components:

```
scadm version -v
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkvm

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

See Also uname(1), syslog(3C), attributes(5)

Name sckmd – Sun cryptographic key management daemon

Synopsis

```
/usr/platform/sun4u/lib/sckmd
```

Description `sckmd` is a server process that resides on a high-end system domain to maintain the Internet Protocol Security (IPsec) Security Associations (SAs) needed to secure communications between a Service Processor or System Controller (SC) and platform management software running within a domain. The `cvcd`(1M) and `dcs`(1M) daemons use these Security Associations. See `ipsec`(7P) for a description of Security Associations.

The `sckmd` daemon receives SAs from the Service Processor or SC and installs these SAs in a domain's Security Association Database (SADB) using `pf_key`(7P).

`sckmd` starts up at system boot time as an SMF service. The FMRI for the `sckmd` service is:

```
svc:/platform/sun4u/sckmd:default
```

A domain supports only one running `sckmd` process at a time.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsckmx.u, SUNWsckmu.u, SUNWsckmr
Interface Stability	Evolving

See Also `cvcd`(1M), `dcs`(1M), `ipsecconf`(1M), `ipsecalgs`(1M), `attributes`(5), `ipsec`(7P), `ipsecah`(7P), `ipsecesp`(7P), `pf_key`(7P)

Notes The `sckmd` service is used only on Sun Fire high-end systems and the SPARC Enterprise Server family. It provides a mechanism for exchanging IPsec keys between a domain and its System Controller (SC) or Service Processor. These platforms use IPsec to secure the communications between the SC or Service Processor and certain platform-specific daemons in the domain. Such daemons currently include `cvcd`(1M) and `dcs`(1M).

The documentation for each platform that supports `sckmd` describes how to configure its use of `IPsec` for such communications. Also, the documentation for each specific application describes how to configure its security policies and `IPsec` options in a manner appropriate for the target platform. Refer to the platform- and application-specific documentation for detailed information.

Name sconadm – register system information

Synopsis

```
/usr/sbin/sconadm register -a
     [-e softwareUpdate | -E softwareUpdate]
     [-h hostname] [-l logfile] [-N]
     [-p proxy_host[:proxy_port]]
     [-r registration_profile] [-u username]
     [-x proxy_username]

/usr/sbin/sconadm proxy [-l logfile]
     [-p proxy_host[:proxy_port]]
     [-r registration_profile] [-x proxy_username]
```

Description The `sconadm` utility is a command-line version of the Basic Registration GUI. In the first form of the command in the SYNOPSIS, `sconadm` uses the `register` subcommand to register a host with a registration server. In the second form, `sconadm` uses the `proxy` subcommand to configure all of the components for software update to use an HTTP web proxy.

The parameters specified with `-u`, `-e` (or `-E`), `-h`, `-p`, and `-x` override values specified in your registration profile. A template for this profile, owned by root, with read-only permissions, is stored in `/usr/lib/breg/data/RegistrationProfile.properties`. See `registration_profile(4)`.

For the `proxy` subcommand, the proxy password is stored in the `RegistrationProfile.properties` file, available if proxy authentication is needed. Storage in the profile prevents proxy passwords from being exposed as part of a listing of processes on a system.

Options The following options are supported:

`-a`	Accept “Terms of Use and Binary Code License”. Absence of this option means that you do not accept the license.
`-e` softwareUpdate	Enable client to be managed at the Sun-hosted Update Connection Service.
`-E` softwareUpdate	Disable client's ability to be managed at the Sun-hosted Update Connection Service.
`-h` *hostname*	Hostname of the machine you want to register.
`-l` *logfile*	Pathname of log file.
`-N`	Never register.
`-p` *proxy_host*[:*proxy_port*]	Proxy host name and optional proxy port number.
`-r` *registration_profile*	Pathname to a registration profile.
`-u` *username*	User name (a Sun Online Account).
`-x` *proxy_username*	User name on the proxy host.

Examples

Unless specified otherwise, the commands below require root privileges or privileges equivalent to root. See `privileges(5)`.

EXAMPLE 1 Registering a New System

Assume a file `registrationprofile.properties` in `/tmp` that contains the following:

```
userName=user123
password=abc123
hostName=
subscriptionKey=
portalEnabled=false
proxyHostName=
proxyPort=
proxyUserName=
proxyPassword=
```

To register a new system using the profile above, you enter:

```
/usr/sbin/sconadm register -a -r /tmp/registrationprofile.properties
```

EXAMPLE 2 Reregistering a System with a Different User

Assume a file `registrationprofile.properties` in `/tmp` with the contents shown below. Note the changed specification for `userName` and `password`.

```
userName=newuser
password=newpassword
hostName=
subscriptionKey=
portalEnabled=false
proxyHostName=
proxyPort=
proxyUserName=
proxyPassword=
```

To reregister a new system using the profile above, you enter the same command you entered to register the system:

```
/usr/sbin/sconadm register -a -r /tmp/registrationprofile.properties
```

EXAMPLE 3 Reregistering a System, Adding a Sun Subscription Key

Modify `registrationprofile.properties` as follows:

```
userName=newuser
password=newpassword
hostName=
subscriptionKey=abc12345678
portalEnabled=false
proxyHostName=
```

EXAMPLE 3 Reregistering a System, Adding a Sun Subscription Key *(Continued)*

```
proxyPort=
proxyUserName=
proxyPassword=
```

Run the command:

```
/usr/sbin/sconadm register -a -r /tmp/registrationprofile.properties
```

EXAMPLE 4 Reregistering and Enabling Access to all Update Connection Services

Modify `registrationprofile.properties` as follows:

```
userName=newuser
password=newpassword
hostName=
subscriptionKey=abc12345678
portalEnabled=false
proxyHostName=
proxyPort=
proxyUserName=
proxyPassword=
```

Note that `portalEnabled` is set to `false`. Run the command:

```
/usr/sbin/sconadm register -a -r /tmp/registrationprofile.properties \
-e softwareUpdate
```

EXAMPLE 5 Never Registering

To never register a system, enter:

```
/usr/sbin/sconadm register -N
```

EXAMPLE 6 Using a Proxy Server With Proxy Authentication

Edit `registrationprofile.properties` as follows:

```
userName=
password=
hostName=
subscriptionKey=
portalEnabled=
proxyHostName=webcache.mycompany.com
proxyPort=8080
proxyUserName=myCompanyProxyUserName
proxyPassword=myCompanyProxyPassword
```

Run the command:

EXAMPLE 6 Using a Proxy Server With Proxy Authentication *(Continued)*

```
/usr/sbin/sconadm proxy -r /tmp/registrationprofile.properties
```

EXAMPLE 7 Changing Proxy Host Settings

Edit `registrationprofile.properties` as follows:

```
userName=
password=
hostName=
subscriptionKey=
portalEnabled=
proxyHostName=webcache.mycompany.com
proxyPort=8080
proxyUserName=myCompanyProxyUserName
proxyPassword=myCompanyProxyPassword
```

Run the command:

```
/usr/sbin/sconadm proxy -r /tmp/registrationprofile.properties
```

Then, change the `proxyHostName` value by running the following command:

```
/usr/sbin/sconadm proxy -r /tmp/registrationprofile.properties \
-p newproxy.mycompany.com
```

After the preceding command all proxies use `newproxy.mycompany.com`.

EXAMPLE 8 Resetting a System Not to Use a Proxy

Edit `registrationprofile.properties` as follows:

```
userName=
password=
hostName=
subscriptionKey=
portalEnabled=
proxyHostName=
proxyPort=
proxyUserName=
proxyPassword=
```

Note that values for all proxy fields are null.

Run the command:

```
/usr/sbin/sconadm proxy -r /tmp/registrationprofile.properties
```

Exit Status

`0` Success.

`>0` An error occurred.

Files

`/usr/lib/breg/data/RegistrationProfile.properties`
Registration profile template.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbrg
Interface Stability	Stable

See Also

`registration_profile(4)`, `attributes(5)`, `privileges(5)`

Name sdpadm – SDP system configuration administration

Synopsis

```
/usr/sbin/sdpadm  status | enable | disable
```

Description The `sdpadm` command is used to display the system state of the Sockets Direct Protocol (SDP) protocol. The `sdpadm` command can optionally be used to set the state of the SDP protocol. See `sdp(7D)`.

By default the SDP protocol is disabled on the system. It can be enabled by using `sdpadm enable`.

Options The following subcommands are supported:

`status` Displays the system status of the SDP protocol

`enable` Enables the SDP protocol

`disable` Disables the SDP protocol

Usage The required privileges to change the state of the SDP protocol are controlled by the network configuration policy. If a user does not have the correct privileges to set the SDP policy, `sdpadm` returns the current state of SDP without having changed the state.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWibsdpu
Interface Stability	Uncommitted

See Also `attributes(5)`, `sdp(7D)`

Infiniband Specification Volume 1 (`http://www.infinibandta.org`)

Name sendmail – send mail over the internet

Synopsis

```
/usr/lib/sendmail [-Ac] [-Am] [-ba] [-bD] [-bd] [-bi] [-bl]
     [-bm] [-bp] [-bP] [-bs] [-bt] [-bv] [-B type] [-C file]
     [-D logfile] [-d X] [-F fullname] [-f name] [-G]
     [-h N] [-L tag] [-M xvalue] [-N notifications] [-n]
     [-Ooption =value] [-o xvalue] [-p protocol]
     [-Q [reason]] [-q [time]] [-q Xstring] [-R ret]
     [-r name] [-t] [-V envid] [-v] [-X logfile]
     [address]...
```

Description The `sendmail` utility sends a message to one or more people, routing the message over whatever networks are necessary. `sendmail` does internetwork forwarding as necessary to deliver the message to the correct place.

`sendmail` is not intended as a user interface routine. Other programs provide user-friendly front ends. `sendmail` is used only to deliver pre-formatted messages.

With no flags, `sendmail` reads its standard input up to an EOF, or a line with a single dot, and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in the local `aliases(4)` file, or in a name service as defined by the `nsswitch.conf(4)` file, and aliased appropriately. In addition, if there is a `.forward` file in a recipient's home directory, `sendmail` forwards a copy of each message to the list of recipients that file contains. Refer to the `NOTES` section for more information about `.forward` files. Aliasing can be prevented by preceding the address with a backslash.

There are several conditions under which the expected behavior is for the alias database to be either built or rebuilt. This cannot occur under any circumstances unless root owns *and* has exclusive write permission to the `/etc/mail/aliases*` files.

If a message is found to be undeliverable, it is returned to the sender with diagnostics that indicate the location and nature of the failure; or, the message is placed in a `dead.letter` file in the sender's home directory.

Service Management The `sendmail` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/smtp:sendmail
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Enabling Access to Remote Clients On an unmodified system, access to `sendmail` by remote clients is enabled and disabled through the service management facility (see `smf(5)`). In particular, remote access is determined by the value of the `local_only` SMF property:

```
svc:/network/smtp:sendmail/config/local_only = true
```

A setting of `true`, as above, disallows remote access; `false` allows remote access. The default value is `true`.

The following example shows the sequence of SMF commands used to enable `sendmail` to allow access to remote systems:

```
# svccfg -s svc:/network/smtp:sendmail setprop config/local_only = false
# svcadm refresh svc:/network/smtp:sendmail
# svcadm restart svc:/network/smtp:sendmail
```

See `svcadm`(1M) and `svccfg`(1M).

Note, however, on a system where any of the `sendmail`(4) files have been customized, setting this property might not have the intended effect. See `sendmail`(4) for details.

Automated Rebuilding of Configuration Files

See `sendmail`(4) for details on which service properties can be set to automate (re)building of configuration files when the service is started.

Restricting Host Access

`sendmail` uses TCP Wrappers to restrict access to hosts. It uses the service name of `sendmail` for `hosts_access()`. For more information on TCP Wrappers, see `tcpd`(1M) and `hosts_access`(4) in the `SUNWtcpd` package. `tcpd`(1M) and `hosts_access`(4) are not part of the Solaris man pages.

Startup Options

The `/etc/default/sendmail` file stores startup options for `sendmail` so that the options are not removed when a host is upgraded. See also `sendmail`(4) for details on which service properties can be set to automate (re)building of configuration files when the service is started.

You can use the following variables in the `/etc/default/sendmail` startup file:

CLIENTOPTIONS=*string*
: Selects additional options to be used with the client daemon, which looks in the client-only queue (`/var/spool/clientmqueue`) and acts as a client queue runner. No syntax checking is done, so be careful when making changes to this variable.

CLIENTQUEUEINTERVAL=*#*
: Similar to the `QUEUEINTERVAL` option, `CLIENTQUEUEINTERVAL` sets the time interval for mail queue runs. However, the `CLIENTQUEUEINTERVAL` option controls the functions of the client daemon, instead of the functions of the master daemon. Typically, the master daemon is able to deliver all messages to the SMTP port. However, if the message load is too high or the master daemon is not running, then messages go into the client-only queue, `/var/spool/clientmqueue`. The client daemon, which checks in the client-only queue, then acts as a client queue processor.

ETRN_HOSTS=*string*
: Enables an SMTP client and server to interact immediately without waiting for the queue run intervals, which are periodic. The server can immediately deliver the portion of its queue that goes to the specified hosts. For more information, refer to the `etrn`(1M) man page.

MODE=-bd
Selects the mode to start `sendmail` with. Use the `-bd` option or leave it undefined.

OPTIONS=*string*
Selects additional options to be used with the master daemon. No syntax checking is done, so be careful when making changes to this variable.

QUEUEINTERVAL=#
Sets the interval for mail queue runs on the master daemon. # can be a positive integer that is followed by either `s` for seconds, `m` for minutes, `h` for hours, `d` for days, or `w` for weeks. The syntax is checked before `sendmail` is started. If the interval is negative or if the entry does not end with an appropriate letter, the interval is ignored and `sendmail` starts with a queue interval of 15 minutes.

QUEUEOPTIONS=p
Enables one persistent queue runner that sleeps between queue run intervals, instead of a new queue runner for each queue run interval. You can set this option to `p`, which is the only setting available. Otherwise, this option is not set.

Mail Filter API

`sendmail` supports a mail filter API called "milter". For more information, see `/usr/include/libmilter/README` and http://www.milter.org

Options

The following options are supported:

`-Ac`
Uses `submit.cf` even if the operation mode does not indicate an initial mail submission.

`-Am`
Uses `sendmail.cf` even if the operation mode indicates an initial mail submission.

`-ba`
Goes into `ARPANET` mode. All input lines must end with a RETURN-LINEFEED, and all messages are generated with a RETURN-LINEFEED at the end. Also, the `From:` and `Sender:` fields are examined for the name of the sender.

`-bd`
Runs as a daemon in the background, waiting for incoming SMTP connections.

`-bD`
Runs as a daemon in the foreground, waiting for incoming SMTP connections.

`-bi`
Initializes the `aliases(4)` database. Root must own *and* have exclusive write permission to the `/etc/mail/aliases*` files for successful use of this option.

`-bl`
Runs as a daemon (like `-bd`) but accepts only loopback SMTP connections.

`-bm`
Delivers mail in the usual way (default).

`-bp`
Prints a summary of the mail queues.

`-bP`
Prints the number of entries in the queues. This option is only available with shared memory support.

`-bs`
Uses the SMTP protocol as described in RFC 2821. This flag implies all the operations of the `-ba` flag that are compatible with SMTP.

`-bt`
Runs in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.

`-bv`
Verifies names only. Does not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

`-B` *type*
Indicates body *type* (`7BIT` or `8BITMIME`).

`-C` *file*
Uses alternate configuration file.

`-D` *logfile*
Send debugging output to the indicated log file instead of `stdout`.

`-d` *X*
Sets debugging value to *X*.

`-f` *name*
Sets the name of the "from" person (that is, the sender of the mail).

`-F` *fullname*
Sets the full name of the sender.

`-G`
When accepting messages by way of the command line, indicates that they are for relay (gateway) submission. When this flag is set, `sendmail` might complain about syntactically invalid messages, for example, unqualified host names, rather than fixing them. `sendmail` does not do any canonicalization in this mode.

`-h` *N*
Sets the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.

`-L` *tag*
Sets the identifier used in `syslog` messages to the supplied *tag*.

`-M`*xvalue*
Sets macro *x* to the specified *value*.

`-n`
Does not do aliasing.

`-N` *notifications*
Tags all addresses being sent as wanting the indicated *notifications*, which consists of the word "NEVER" or a comma-separated list of "SUCCESS", "FAILURE", and "DELAY" for successful delivery, failure and a message that is stuck in a queue somwhere. The default is "FAILURE,DELAY".

`-o`*xvalue*
Sets option *x* to the specified *value*. `Processing Options` are described below.

`-O`*option*=*value*
Sets *option* to the specified *value* (for long from names). `Processing Options` are described below.

`-p` *protocol*
Sets the sending protocol. The *protocol* field can be in form *protocol*`:`*host* to set both the sending protocol and the sending host. For example: `-pUUCP:uunet` sets the sending *protocol* to `UUCP` and the sending host to `uunet`. Some existing programs use `-oM` to set the *r* and *s* macros; this is equivalent to using `-p`.

`-q`[*time*]
Processes saved messages in the queue at given intervals. If *time* is omitted, processes the queue once. *time* is given as a tagged number, where *s* is seconds, *m* is minutes, *h* is hours, *d* is days, and *w* is weeks. For example, `-q1h30m` or `-q90m` would both set the timeout to one hour thirty minutes.

By default, sendmail runs in the background. This option can be used safely with `-bd`.

`-qp[`*time*`-]`
Similar to `-q`[*time*], except that instead of periodically forking a child to process the queue, `sendmail` forks a single persistent child for each queue that alternates between processing the queue and sleeping. The sleep time (*time*) is specified as the argument; it defaults to `1` second. The process always sleeps at least `5` seconds if the queue was empty in the previous queue run.

`-qf`
Processes saved messages in the queue once and does not `fork(2)`, but runs in the foreground.

`-qG` *name*
Processes jobs in queue group called *name* only.

`-q[!]I` *substr*
Limits processed jobs to those containing *substr* as a substring of the queue ID or not when `!` is specified.

`-q[!]Q` *substr*
Limits processed jobs to those quarantined jobs containing *substr* as a substring of the quarantine *reason* or not when `!` is specified.

`-q[!]R` *substr*
Limits processed jobs to those containing *substr* as a substring of one of the recipients or not when `!` is specified.

`-q[!]S` *substr*
Limits processed jobs to those containing *substr* as a substring of the sender or not when `!` is specified.

`-Q`[*reason*]
Quarantines a normal queue item with the given reason or unquarantines a quarantined queue item if no reason is given. This should only be used with some sort of item matching as described above.

`-r` *name*
An alternate and obsolete form of the `-f` flag.

`-R` *ret*
Identifies the information you want returned if the message bounces. *ret* can be `HDRS` for headers only or `FULL` for headers plus body.

`-t`
Reads message for recipients. `To:`,`Cc:`, and `Bcc:` lines are scanned for people to send to. The `Bcc:` line is deleted before transmission. Any addresses in the argument list is suppressed. The `NoRecipientAction` Processing Option can be used to change the behavior when no legal recipients are included in the message.

`-v`
Goes into verbose mode. Alias expansions are announced, and so forth.

`-V` *envid*
The indicated *envid* is passed with the envelope of the message and returned if the message bounces.

`-X` *logfile*
Logs all traffic in and out of `sendmail` in the indicated *logfile* for debugging mailer problems. This produces a lot of data very quickly and should be used sparingly.

Processing Options

There are a number of "random" options that can be set from a configuration file. Options are represented by a single character or by multiple character names. The syntax for the single character names of is:

`O`*xvalue*

This sets option *x* to be *value*. Depending on the option, *value* may be a string, an integer, a boolean (with legal values `t`, `T`, `f`, or `F`; the default is `TRUE`), or a time interval.

The multiple character or long names use this syntax:

`O` *Longname=argument*

This sets the option *Longname* to be *argument*. The long names are beneficial because they are easier to interpret than the single character names.

Not all processing options have single character names associated with them. In the list below, the multiple character name is presented first followed by the single character syntax enclosed in parentheses.

`AliasFile (A`*file*`)`
Specifies possible alias files.

`AliasWait (a` *N*`)`
If set, waits up to *N* minutes for an "@:@" entry to exist in the `aliases(4)` database before starting up. If it does not appear in *N* minutes, issues a warning. Defaults to 10 minutes.

`AllowBogusHELO`
Allows a `HELO SMTP` command that does not include a host name. By default this option is disabled.

`BadRcptThrottle=`*N*
If set and more than the specified number of recipients in a single SMTP envelope are rejected, sleeps for one second after each rejected RCPT command.

`BlankSub (B`*c*`)`
Sets the blank substitution character to *c*. Unquoted spaces in addresses are replaced by this character. Defaults to SPACE (that is, no change is made).

`CACertFile`
File containing one CA cert.

`CACertPath`
Path to directory with certs of CAs.

`CheckAliases (n)`
Validates the RHS of aliases when rebuilding the `aliases(4)` database.

`CheckpointInterval (C`*N*`)`
Checkpoints the queue every *N* (default `10`) addresses sent. If your system crashes during delivery to a large list, this prevents retransmission to any but the last *N* recipients.

`ClassFactor (z`*fact*`)`
The indicated factor *fact* is multiplied by the message class (determined by the `Precedence:` field in the user header and the `P` lines in the configuration file) and subtracted from the priority. Thus, messages with a higher `Priority:` are favored. Defaults to `1800`.

`ClientCertFile`
File containing the cert of the client, that is, this cert is used when `sendmail` acts as client.

`ClientKeyFile`
File containing the private key belonging to the client cert.

`ClientPortOptions`
Sets client SMTP options. The options are key=value pairs. Known keys are:

`Addr` *Address Mask*
Address Mask defaults to `INADDR_ANY`. The address mask can be a numeric address in dot notation or a network name.

`Family`
Address family (defaults to INET).

`Listen`
Size of listen queue (defaults to 10).

`Port`
Name/number of listening port (defaults to `smtp`).

`RcvBufSize`
The size of the TCP/IP receive buffer.

`SndBufSize`
The size of the TCP/IP send buffer.

`Modifier`
Options (flags) for the daemon. Can be:

`h`
Uses name of interface for `HELO` command.

If `h` is set, the name corresponding to the outgoing interface address (whether chosen by means of the `Connection` parameter or the default) is used for the `HELO`/`EHLO` command.

`ColonOkInAddr`
If set, colons are treated as a regular character in addresses. If not set, they are treated as the introducer to the RFC 822 "group" syntax. This option is on for version 5 and lower configuration files.

`ConnectionCacheSize (k`*N*`)`
The maximum number of open connections that are to be cached at a time. The default is `1`. This delays closing the current connection until either this invocation of `sendmail` needs to connect to another host or it terminates. Setting it to `0` defaults to the old behavior, that is, connections are closed immediately.

`ConnectionCacheTimeout (K`*timeout*`)`
The maximum amount of time a cached connection is permitted to idle without activity. If this time is exceeded, the connection is immediately closed. This value should be small (on the order of ten minutes). Before `sendmail` uses a cached connection, it always sends a `NOOP` (no operation) command to check the connection. If the `NOOP` command fails, it reopens the connection. This keeps your end from failing if the other end times out. The

point of this option is to be a good network neighbor and avoid using up excessive resources on the other end. The default is five minutes.

`ConnectionRateThrottle`
The maximum number of connections permitted per second. After this many connections are accepted, further connections are delayed. If not set or <= 0, there is no limit.

`ConnectionRateWindowSize`
Define the length of the interval for which the number of incoming connections is maintained. The default is 60 seconds.

`ControlSocketName`
Name of the control socket for daemon management. A running `sendmail` daemon can be controlled through this Unix domain socket. Available commands are: `help`, `restart`, `shutdown`, and `status`. The `status` command returns the current number of daemon children, the free disk space (in blocks) of the queue directory, and the load average of the machine expressed as an integer. If not set, no control socket is available. For the sake of security, this Unix domain socket must be in a directory which is accessible only by root; `/var/spool/mqueue/.smcontrol` is recommended for the socket name.

`CRLFile`
File containing certificate revocation status, useful for X.509v3 authentication.

`DaemonPortOptions (O`*options*`)`
Sets server SMTP options. The options are *key=value* pairs. Known keys are:

`Name`
User-definable name for the daemon (defaults to "`Daemon#`"). Used for error messages and logging.

`Addr`
Address mask (defaults `INADDR_ANY`).

The address mask may be a numeric address in dot notation or a network name.

`Family`
Address family (defaults to `INET`).

`InputMailFilters`
List of input mail filters for the daemon.

`Listen`
Size of listen queue (defaults to `10`).

`Modifier`
Options (flags) for the daemon; can be a sequence (without any delimiters) of:

`a`
Requires authentication.

`b`
Binds to interface through which mail has been received.

`c`
Performs hostname canonification (`.cf`).

`f`
Requires fully qualified hostname (`.cf`).

`h`
Uses name of interface for `HELO` command.

`u`
Allows unqualified addresses (`.cf`).

`C`
Does not perform hostname canonification.

`E`
Disallows `ETRN` (see RFC 2476).

`Name`
User-definable name for the daemon (defaults to `Daemon#`). Used for error messages and logging.

`Port`
Name/number of listening port (defaults to `smtp`).

`ReceiveSize`
The size of the TCP/IP receive buffer.

`SendSize`
The size of the TCP/IP send buffer.

`children`
Maximum number of children per daemon. See `MaxDaemonChildren`.

`DeliveryMode`
Delivery mode per daemon. See `DeliveryMode`.

`refuseLA`
`RefuseLA` per daemon.

`delayLA`
`DelayLA` per daemon.

`queueLA`
`QueueLA` per daemon.

`sendmail` listens on a new socket for each occurrence of the `DaemonPortOptions` option in a configuration file.

`DataFileBufferSize`
Sets the threshold, in bytes, before a memory-bases queue data file becomes disk-based. The default is 4096 bytes.

`DeadLetterDrop`
Defines the location of the system-wide dead.letter file, formerly hard-coded to `/var/tmp/dead.letter`. If this option is not set (the default), `sendmail` does not attempt to save to a system-wide `dead.letter` file in the event it cannot bounce the mail to the user or postmaster. Instead, it renames the `qf` file as it has in the past when the `dead.letter` file could not be opened.

`DefaultCharSet`
Sets the default character set to use when converting unlabeled 8 bit input to MIME.

`DefaultUser (g`*gid*`) or (u`*uid*`)`
Sets the default group ID for mailers to run in to *gid* or set the default userid for mailers to *uid*. Defaults to `1`. The value can also be given as a symbolic group or user name.

`DelayLA=`*LA*
When the system load average exceeds *LA*, `sendmail` sleeps for one second on most SMTP commands and before accepting connections.

`DeliverByMin=`*time*
Sets minimum time for `Deliver By SMTP Service Extension` (RFC 2852). If `0`, no time is listed, if less than `0`, the extension is not offered, if greater than `0`, it is listed as minimum time for the `EHLO` keyword `DELIVERBY`.

`DeliveryMode (d`*x*`)`
Delivers in mode *x*. Legal modes are:

`i`
Delivers interactively (synchronously).

`b`
Delivers in background (asynchronously).

`d`
Deferred mode. Database lookups are deferred until the actual queue run.

`q`
Just queues the message (delivers during queue run).

Defaults to `b` if no option is specified, `i` if it is specified but given no argument (that is, `Od` is equivalent to `Odi`).

`DHParameters`
File containing the DH parameters.

`DialDelay`
If a connection fails, waits this many seconds and tries again. Zero means “do not retry”.

`DontBlameSendmail`
If set, overrides the file safety checks. This compromises system security and should not be used. See http://www.sendmail.org/tips/DontBlameSendmail.php for more information.

`DontExpandCnames`
If set, $[... $] lookups that do DNS-based lookups do not expand CNAME records.

`DontInitGroups`
If set, the `initgroups(3C)` routine is never invoked. If you set this, agents run on behalf of users only have their primary (`/etc/passwd`) group permissions.

`DontProbeInterfaces`
If set, `sendmail` does not insert the names and addresses of any local interfaces into the `$=w class`. If set, you must also include support for these addresses, otherwise mail to addresses in this list bounces with a configuration error.

`DontPruneRoutes (R)`
If set, does not prune route-addr syntax addresses to the minimum possible.

`DoubleBounceAddress`
If an error occurs when sending an error message, sends that "double bounce" error message to this address.

`EightBitMode (8)`
Uses 8–bit data handling. This option requires one of the following keys. The key can selected by using just the first character, but using the full word is better for clarity.

`mimify`
Does any necessary conversion of `8BITMIME` to 7–bit.

`pass`
Passes unlabeled 8–bit input through as is.

`strict`
Rejects unlabeled 8–bit input.

`ErrorHeader (E`*file/message*`)`
Appends error messages with the indicated message. If it begins with a slash, it is assumed to be the pathname of a file containing a message (this is the recommended setting). Otherwise, it is a literal message. The error file might contain the name, email address, and/or phone number of a local postmaster who could provide assistance to end users. If the option is missing or `NULL`, or if it names a file which does not exist or which is not readable, no message is printed.

`ErrorMode (e`*x*`)`
Disposes of errors using mode *x*. The values for *x* are:

`e`
Mails back errors and gives `0` exit status always.

`m`
Mails back errors.

`p`
Prints error messages (default).

`q`
No messages, just gives exit status.

`w`
Writes back errors (mail if user not logged in).

`FallbackMXhost (V`*fallbackhost*`)`
If specified, the *fallbackhost* acts like a very low priority MX on every host. This is intended to be used by sites with poor network connectivity.

`FallBackSmartHost`
If specified, the *fallBackSmartHost* is used in a last-ditch effort for each host. This is intended to be used by sites with "fake internal DNS". That is, a company whose DNS accurately reflects the world inside that company's domain but not outside.

`FastSplit`
If set to a value greater than zero (the default is one), it suppresses the MX lookups on addresses when they are initially sorted, that is, for the first delivery attempt. This usually results in faster envelope splitting unless the MX records are readily available in a local DNS cache. To enforce initial sorting based on MX records set `FastSplit` to zero. If the mail is submitted directly from the command line, then the value also limits the number of processes to deliver the envelopes; if more envelopes are created they are only queued up and must be taken care of by a queue run. Since the default submission method is by way of SMTP (either from a MUA or by way of the Message Submission Program [MSP]), the value of `FastSplit` is seldom used to limit the number of processes to deliver the envelopes.

`ForkEachJob (Y)`
If set, delivers each job that is run from the queue in a separate process. Use this option if you are short of memory, since the default tends to consume considerable amounts of memory while the queue is being processed.

`ForwardPath (J`*path*`)`
Sets the path for searching for users' `.forward` files. The default is `$z/.forward`. Some sites that use the automounter may prefer to change this to `/var/forward/$u` to search a file with the same name as the user in a system directory. It can also be set to a sequence of paths separated by colons; `sendmail` stops at the first file it can successfully and safely open. For example, `/var/forward/$u:$z/.forward` searches first in `/var/forward/` *username* and then in *~username*`/.forward` (but only if the first file does not exist). Refer to the NOTES section for more information.

`HeloName=`*name*
Sets the name to be used for `HELO`/`EHLO` (instead of `$j`).

`HelpFile (H`*file*`)`
Specifies the help file for SMTP.

`HoldExpensive (c)`
If an outgoing mailer is marked as being expensive, does not connect immediately.

`HostsFile`
Sets the file to use when doing "file" type access of host names.

`HostStatusDirectory`
If set, host status is kept on disk between `sendmail` runs in the named directory tree. If a full path is not used, then the path is interpreted relative to the queue directory.

`IgnoreDots (i)`
Ignores dots in incoming messages. This is always disabled (that is, dots are always accepted) when reading SMTP mail.

`LogLevel (L`*n*`)`
Sets the default log level to *n*. Defaults to `9`.

`(M`*x value*`)`
Sets the macro *x* to *value*. This is intended only for use from the command line.

`MailboxDatabase`
Type of lookup to find information about local mail boxes, defaults to `pw` which uses `getpwnam(3C)`. Other types can be introduced by adding them to the source code, see `libsm/mbdb.c` for details.

`MatchGECOS (G)`
Tries to match recipient names using the GECOS field. This allows for mail to be delivered using names defined in the GECOS field in `/etc/passwd` as well as the login name.

`MaxDaemonChildren`
The maximum number of children the daemon permits. After this number, connections are rejected. If not set or <=0, there is no limit.

`MaxHopCount (h`*N*`)`
The maximum hop count. Messages that have been processed more than *N* times are assumed to be in a loop and are rejected. Defaults to `25`.

`MaxMessageSize`
The maximum size of messages that are accepted (in bytes).

`MaxMimeHeaderLength=`*M*`[/`*N*`]`
Sets the maximum length of certain MIME header field values to *M* characters. For some of these headers which take parameters, the maximum length of each parameter is set to *N* if specified. If `/`*N* is not specified, one half of *M* is used. By default, these values are `0`, meaning no checks are done.

`MaxNOOPCommands=`*N*
Overrides the default of 20 for the number of useless commands.

`MaxQueueChildren=`*N*
When set, this limits the number of concurrent queue runner processes to *N*. This helps to control the amount of system resources used when processing the queue. When there are multiple queue groups defined and the total number of queue runners for these queue groups would exceed `MaxQueueChildren` then the queue groups are not all run concurrently. That is, some portion of the queue groups run concurrently such that `MaxQueueChildren` is not be exceeded, while the remaining queue groups are run later (in round robin order). See `MaxRunnersPerQueue`.

`MaxQueueRunSize`
If set, limits the maximum size of any given queue run to this number of entries. This stops reading the queue directory after this number of entries is reached; job priority is not used. If not set, there is no limit.

`MaxRunnersPerQueue=`*N*
This sets the default maximum number of queue runners for queue groups. Up to *N* queue runners work in parallel on a queue group's messages. This is useful where the processing of a message in the queue might delay the processing of subsequent messages. Such a delay can be the result of non-erroneous situations such as a low bandwidth connection. The can be overridden on a per queue group basis by setting the `Runners` option. The default is `1` when not set.

`MeToo (M)`
Sends to me too, even if I am in an alias expansion.

`MaxRecipientsPerMessage`
If set, allows no more than the specified number of recipients in an SMTP envelope. Further recipients receive a 452 error code and are deferred for the next delivery attempt.

`MinFreeBlocks (b`*N*`/`*M*`)`
Insists on at least *N* blocks free on the file system that holds the queue files before accepting email by way of SMTP. If there is insufficient space, `sendmail` gives a `452` response to the `MAIL` command. This invites the sender to try again later. The optional *M* is a maximum message size advertised in the `ESMTP EHLO` response. It is currently otherwise unused.

`MinQueueAge`
Specifies the amount of time a job must sit in the queue between queue runs. This allows you to set the queue run interval low for better responsiveness without trying all jobs in each run. The default value is 0.

`MustQuoteChars`
Specifies the characters to be quoted in a full name phrase. `&,;:\()[]` are quoted automatically.

`NiceQueueRun`
Specifies the priority of queue runners. See `nice(1)`.

`NoRecipientAction`
Sets action if there are no legal recipient files in the message. The legal values are:

`add-apparently-to`
Adds an `Apparently-to:` header with all the known recipients (which may expose blind recipients).

`add-bcc`
Adds an empty `Bcc:` header.

`add-to`
Adds a `To:` header with all the known recipients (which may expose blind recipients).

`add-to-undisclosed`
Adds a `To: undisclosed-recipients:` header.

`none`
Does nothing, that is, leaves the message as it is.

`OldStyleHeaders (o)`
Assumes that the headers may be in old format, that is, spaces delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it is assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.

`OperatorChars or $o`
Defines the list of characters that can be used to separate the components of an address into tokens.

`PidFile`
Specifies the filename of the `pid` file. The default is `/var/run/sendmail.pid`. The filename is macro-expanded before it is opened, and unlinked when `sendmail` exits.

`PostmasterCopy (P`*`postmaster`*`)`
If set, copies of error messages are sent to the named *postmaster*. Only the header of the failed message is sent. Since most errors are user problems, this is probably not a good idea on large sites, and arguably contains all sorts of privacy violations, but it seems to be popular with certain operating systems vendors.

`PrivacyOptions (p`*`opt,opt,...`*`)`
Sets privacy options. Privacy is really a misnomer; many of these options are just a way of insisting on stricter adherence to the SMTP protocol.

The `goaway` pseudo-flag sets all flags except `noreceipts`, `restrictmailq`, `restrictqrun`, `restrictexpand`, `noetrn`, and `nobodyreturn`. If `mailq` is restricted, only people in the same group as the queue directory can print the queue. If queue runs are restricted, only root and the owner of the queue directory can run the queue. The `restrict-expand` pseudo-flag instructs sendmail to drop privileges when the -bv option is given by users who are neither root nor the `TrustedUser` so users cannot read private aliases, forwards, or `:include:` files. It adds the `NonRootSafeAddr` to the "DontBlame-Sendmail" option to prevent misleading unsafe address warnings. It also overrides the `-v` (verbose) command line option to prevent

information leakage. Authentication Warnings add warnings about various conditions that may indicate attempts to fool the mail system, such as using an non-standard queue directory.

The options can be selected from:

`authwarnings`
Puts `X-Authentication-Warning:` headers in messages.

`goaway`
Disallows essentially all SMTP status queries.

`needexpnhelo`
Insists on `HELO` or `EHLO` command before `EXPN`.

`needmailhelo`
Insists on `HELO` or `EHLO` command before `MAIL`.

`needvrfyhelo`
Insists on `HELO` or `EHLO` command before `VRFY`.

`noactualrecipient`
Do not put an X-Actual-Recipient line in a DNS that reveals the actual account to which an address is mapped.

`noetrn`
Disallows `ETRN` entirely.

`noexpn`
Disallows `EXPN` entirely.

`noreceipts`
Prevents return receipts.

`nobodyreturn`
Does not return the body of a message with DSNs.

`novrfy`
Disallows `VRFY` entirely.

`public`
Allows open access.

`restrictexpand`
Restricts `-bv` and `-v` command line flags.

`restrictmailq`
Restricts `mailq` command.

`restrictqrun`
Restricts `-q` command line flag.

`ProcessTitlePrefix` *string*
Prefixes the process title shown on "`/usr/ucb/ps auxww`" listings with *string*. The string is macro processed.

`QueueDirectory (Q`*dir*`)`
Uses the named *dir* as the queue directory.

`QueueFactor (q`*factor*`)`
Uses *factor* as the multiplier in the map function to decide when to just queue up jobs rather than run them. This value is divided by the difference between the current load average and the load average limit (`x` flag) to determine the maximum message priority to be sent. Defaults to `600000`.

`QueueFileMode=`*mode*
Defaults permissions for queue files (octal). If not set, `sendmail` uses `0600` unless its real and effective `uid` are different in which case it uses `0644`.

`QueueLA (x`*LA*`)`
When the system load average exceeds *LA*, just queues messages (that is, does not try to send them). Defaults to eight times the number of processors online when `sendmail` starts.

`QueueSortOrder=`*algorithm*
Sets the algorithm used for sorting the queue. Only the first character of the value is used. Legal values are `host` (to order by the name of the first host name of the first recipient), `filename` (to order by the name of the queue file name), `time` (to order by the submission/creation time), `random` (to order randomly), `modification` (to order by the modification time of the `qf` file (older entries first)), `none` (to not order), and `priority` (to order by message priority). Host ordering makes better use of the connection cache, but may tend to process low priority messages that go to a single host over high priority messages that go to several hosts; it probably shouldn't be used on slow network links. Filename and modification time ordering saves the overhead of reading all of the queued items before starting the queue run. Creation (submission) time ordering is almost always a bad idea, since it allows large, bulk mail to go out before smaller, personal mail, but may have applicability on some hosts with very fast connections. Random is useful if several queue runners are started by hand which try to drain the same queue since odds are they are working on different parts of the queue at the same time. Priority ordering is the default.

`QueueTimeout (T`*rtime*`/`*wtime*`)`
Sets the queue timeout to *rtime*. After this interval, messages that have not been successfully sent are returned to the sender. Defaults to five days (`5d`). The optional *wtime* is the time after which a warning message is sent. If it is missing or `0`, then no warning messages are sent.

`RandFile`
File containing random data (use prefix `file:`) or the name of the UNIX socket if EGD is used (use prefix `egd:`). Note that Solaris supports `random(7D)`, so this does not need to be specified.

`RecipientFactor (y`*fact*`)`
The indicated factor *fact* is added to the priority (thus *lowering* the priority of the job) for each recipient, that is, this value penalizes jobs with large numbers of recipients. Defaults to `30000`.

`RefuseLA (X`*LA*`)`
When the system load average exceeds *LA*, refuses incoming SMTP connections. Defaults to 12 times the number of processors online when `sendmail` starts.

`RejectLogInterval`
Log interval when refusing connections for this long (default: 3h).

`ResolverOptions (I)`
Tunes DNS lookups.

`RetryFactor (Z`*fact*`)`
The indicated factor *fact* is added to the priority every time a job is processed. Thus, each time a job is processed, its priority is decreased by the indicated value. In most environments this should be positive, since hosts that are down are all too often down for a long time. Defaults to `90000`.

`RrtImpliesDsn`
If this option is set, a `Return-Receipt-To:` header causes the request of a DSN, which is sent to the envelope sender as required by RFC 1891, not to the address given in the header.

`RunAsUser`
If set, becomes this user when reading and delivering mail. Intended for use of firewalls where users do not have accounts.

`SafeFileEnvironment`
If set, `sendmail` does a `chroot` into this directory before writing files.

`SaveFromLine (f)`
Saves Unix-style `From` lines at the front of headers. Normally they are assumed redundant and discarded.

`SendMimeErrors (j)`
If set, sends error messages in MIME format (see RFC 2045 and RFC 1344 for details). If disabled, `sendmail` does not return the DSN keyword in response to an `EHLO` and does not do Delivery Status Notification processing as described in RFC 1891.

`ServerCertFile`
File containing the cert of the server, that is, this cert is used when `sendmail` acts as server.

`ServerKeyFile`
File containing the private key belonging to the server cert.

`ServiceSwitchFile`
Defines the path to the service-switch file. Since the service-switch file is defined in the Solaris operating environment this option is ignored.

`SevenBitInput (7)`
Strips input to seven bits for compatibility with old systems. This should not be necessary.

`SharedMemoryKey`
Specifies key to use for shared memory segment. If not set (or `0`), shared memory is not be used. If this option is set, `sendmail` can share some data between different instances. For example, the number of entries in a queue directory or the available space in a file system. This allows for more efficient program execution, since only one process needs to update the data instead of each individual process gathering the data each time it is required.

`SharedMemoryKeyFile=`*file*
If `SharedMemoryKeyFile` is set to `-1`, the automatically selected shared memory key will be stored in the specified file.

`SingleLineFromHeader`
If set, `From:` lines that have embedded newlines are unwrapped onto one line.

`SingleThreadDelivery`
If this option and the `HostStatusDirectory` option are both set, uses single thread deliveries to other hosts.

`SmtpGreetingMessage or $e`
Specifies the initial SMTP greeting message.

`SoftBounce`
If set, issue temporary errors (`4xy`) instead of permanent errors (`5xy`). This can be useful during testing of a new configuration to avoid erroneous bouncing of mail.

`StatusFile (S`*file*`)`
Logs statistics in the named *file*. By default, this is `/etc/mail/sendmail.st`. As root, you must `touch(1)` this file to enable `mailstats(1)`.

`SuperSafe (s)`
This option can be set to `True`, `False`, `Interactive`, or `PostMilter`. If set to `True`, `sendmail` is set to super-safe when running things, that is, always instantiate the queue file, even if you are going to attempt immediate delivery. `sendmail` always instantiates the queue file before returning control to the client under any circumstances. This should really always be set to `True`. The `Interactive` value has been introduced in `8.12` and can be used together with `DeliveryMode=i`. It skips some synchronization calls which are effectively doubled in the code execution path for this mode. If set to `PostMilter`, `sendmail` defers synchronizing the queue file until any milters have signaled acceptance of the message. `PostMilter` is useful only when `sendmail` is running as an SMTP server; in all other situations it acts the same as True.

`TempFileMode (F`*mode*`)`
Specifies the file mode for queue files.

`Timeout (r`*timeouts*`)`
Timeout reads after time interval. The *timeouts* argument is a list of *keyword=value* pairs. All but *command* apply to client SMTP. For backward compatibility, a timeout with no

keyword= part is set all of the longer values. The recognized timeouts and their default values, and their minimum values specified in RFC 1123 section 5.3.2 are:

`aconnect`
all connections for a single delivery attempt [0, unspecified]

`command`
command read [1h, 5m]

`connect`
initial connect [0, unspecified]

`control`
complete control socket transaction [2m, none]

`datablock`
data block read [1h, 3m]

`datafinal`
reply to final `.` in data [1h, 10m]

`datainit`
reply to `DATA` command [5m, 2m]

`fileopen`
file open [60sec, none]

`helo`
reply to `HELO` or `EHLO` command [5m, none]

`hoststatus`
host retry [30m, unspecified]

`iconnect`
first attempt to connect to a host [0, unspecified]

`ident`
`IDENT` protocol timeout [5s, none]

`initial`
wait for initial greeting message [5m, 5m]

`lhlo`
wait for reply to an LMTP LHLO command [2m, unspecified]

`mail`
reply to `MAIL` command [10m, 5m]

`misc`
reply to `NOOP` and `VERB` commands [2m, none]

`queuereturn`
undeliverable message returned [5d]

`queuewarn`
deferred warning [4h]

`quit`
reply to `QUIT` command [2m, none]

`rcpt`
reply to `RCPT` command [1h, 5m]

`resolver.retrans`
Resolver's retransmission time interval (in seconds) [varies]. Sets both `Timeout.resolver.retrans.first` and `Timeout.resolver.retrans.normal`.

`resolver.retrans.first`
Resolver's retransmission time interval (in seconds) for the first attempt to deliver a message [varies].

`resolver.retrans.normal`
Resolver's retransmission time interval (in seconds) for all look-ups except the first delivery attempt [varies].

`resolver.retry`
Number of times to retransmit a resolver query [varies]. Sets both `Timeout.resolver.retry.first` and `Timeout.resolver.retry.normal`.

`resolver.retry.first`
Number of times to retransmit a resolver query for the first attempt to deliver a message [varies].

`resolver.retry.normal`
Number of times to retransmit a resolver query for all look-ups except the first delivery attempt [varies].

`rset`
reply to `RSET` command [5m, none]

`starttls`
response to an `SMTP STARTTLS` command [1h]

`TimeZoneSpec (t`*tzinfo*`)`
Sets the local time zone info to *tzinfo*, for example, "PST8PDT". Actually, if this is not set, the `TZ` environment variable is cleared (so the system default is used); if set but null, the user's `TZ` variable is used, and if set and non-null, the `TZ` variable is set to this value.

`TLSSrvOptions`
If this option is '`V`', then no client verification is performed,that is, the server does not ask for a certificate.

`TrustedUser`
The user parameter can be a user name (looked up in the passwd map) or a numeric user id. Trusted user for file ownership and starting the daemon. If set, generated alias databases and the control socket (if configured) are automatically owned by this user.

`TryNullMXList (w)`
If you are the "best" (that is, lowest preference) MX for a given host, you should normally detect this situation and treat that condition specially, by forwarding the mail to a UUCP feed, treating it as local, or whatever. However, in some cases (such as Internet firewalls) you may want to try to connect directly to that host as though it had no MX records at all. Setting this option causes `sendmail` to try this. The downside is that errors in your configuration are likely to be diagnosed as "host unknown" or "message timed out" instead of something more meaningful. This option is deprecated.

`UnixFromLine or $l`
The "From " line used when sending to files or programs.

`UnsafeGroupWrites`
If set, group-writable :include: and .forward files are considered "unsafe", that is, programs and files cannot be directly referenced from such files.

`UseErrorsTo (l)`
If there is an `Errors-To`: header, sends error messages to the addresses listed there. They normally go to the envelope sender. Use of this option causes `sendmail` to violate RFC 1123. This option is not recommended and deprecated.

`UseMSP`
Uses as mail submission program, that is, allows group writable queue files if the group is the same as that of a set-group-id `sendmail` binary.

`UserDatabaseSpec (U)`
Defines the name and location of the file containing User Database information.

`Verbose (v)`
Runs in verbose mode. If this is set, `sendmail` adjusts the `HoldExpensive` and `DeliveryMode` options so that all mail is delivered completely in a single job so that you can see the entire delivery process. The `Verbose` option should *never* be set in the configuration file; it is intended for command line use only.

`XscriptFileBufferSize`
Sets the threshold, in bytes, before a memory-bases queue transcript file becomes disk-based. The default is 4096 bytes.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep `sendmail` from suppressing the blanks from between arguments.

If invoked as `newaliases`, `sendmail` rebuilds the alias database, so long as the `/etc/mail/aliases*` files are owned by root *and* root has exclusive write permission. If invoked as `mailq`, `sendmail` prints the contents of the mail queue.

Operands

address
address of an intended recipient of the message being sent.

Usage

See `largefile(5)` for the description of the behavior of `sendmail` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status

`sendmail` returns an exit status describing what it did. The codes are defined in `/usr/include/sysexits.h`.

`EX_OK`
Successful completion on all addresses.

`EX_NOUSER`
User name not recognized.

`EX_UNAVAILABLE`
Catchall. Necessary resources were not available.

`EX_SYNTAX`
Syntax error in address.

`EX_SOFTWARE`
Internal software error, including bad arguments.

`EX_OSERR`
Temporary operating system error, such as “cannot fork”.

`EX_NOHOST`
Host name not recognized.

`EX_TEMPFAIL`
Message could not be sent immediately, but was queued.

Environment Variables

No environment variables are used. However, sendmail's start-up script, invoked by `svcadm(1M)`, reads `/etc/default/sendmail`. In this file, if the variable `ETRN_HOSTS` is set, the start-up script parses this variable and invokes `etrn(1M)` appropriately. `ETRN_HOSTS` should be of the form:

```
"s1:c1.1,c1.2        s2:c2.1 s3:c3.1,c3.2,c3.3"
```

That is, white-space separated groups of *server:client* where *client* can be one or more comma-separated names. The *:client* part is optional. *server* is the name of the server to prod; a mail queue run is requested for each *client* name. This is comparable to running:

```
/usr/lib/sendmail -qR client
```

on the host *server*.

Files

`dead.letter`
Unmailable text

`/etc/default/sendmail`
Contains default settings. You can override some of the settings by command line options.

`/etc/mail/aliases`
Mail aliases file (ASCII)

`/etc/mail/aliases.db`
Database of mail aliases (binary)

`/etc/mail/aliases.dir`
Database of mail aliases (binary)

`/etc/mail/aliases.pag`
Database of mail aliases (binary)

`/etc/mail/sendmail.cf`
Defines environment for `sendmail`

`/etc/mail/submit.cf`
Defines environment for MSP

`/etc/mail/trusted-users`
Lists users that are "trusted", that is, able to set their envelope from address using `-f` without generating a warning message. Note that this file is consulted by the default `sendmail.cf`, but not by the default `submit.cf`, in which the line referring to `/etc/mail/trusted-users` is commented out. See `sendmail(4)` for instructions on making changes to `submit.cf` and `sendmail.cf`.

`/var/spool/clientmqueue/*`
Temporary files and queued mail

`/var/spool/mqueue/*`
Temporary files and queued mail

`~/.forward`
List of recipients for forwarding messages

`/usr/include/libmilter/README`
Describes the steps needed to compile and run a filter

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/smtp/sendmail

See Also svcs(1), biff(1B), mail(1), mailq(1), mailx(1), nice(1), check-hostname(1M), check-permissions(1M), etrn(1M), newaliases(1M), svcadm(1M), svccfg(1M), fork(2), getpwnam(3C)getusershell(3C)resolver(3RESOLV)aliases(4), hosts(4), sendmail(4), shells(4), attributes(5), largefile(5), smf(5), random(7D)

tcpd(1M), hosts_access(4) in the SUNWtcpd package.

RFC 2821 *Simple Mail Transfer Protocol*, John Klensin, April 2001.

RFC 2822 *Internet Message Format*, Pete Resnick, April 2001.

sendmail, Third Edition, Bryan Costales with Eric Allman, O'Reilly & Associates, Inc., 2003.

http://www.sendmail.org

http://www.milter.org

Notes The sendmail program requires a fully qualified host name when starting. A script has been included to help verify if the host name is defined properly (see check-hostname(1M)).

The permissions and the ownership of several directories have been changed in order to increase security. In particular, access to /etc/mail and /var/spool/mqueue has been restricted.

Security restrictions have been placed users using .forward files to pipe mail to a program or redirect mail to a file. The default shell (as listed in /etc/passwd) of these users must be listed in /etc/shells. This restriction does not affect mail that is being redirected to another alias.

Additional restrictions have been put in place on .forward and :include: files. These files and the directory structure that they are placed in cannot be group- or world-writable. See check-permissions(1M).

If you have interfaces that map to domains that have MX records that point to non-local destinations, you might need to enable the DontProbeInterfaces option to enable delivery to those destinations. In its default startup behavior, sendmail probes each interface and adds an interface's IP addresses, as well as any domains that those addresses map to, to its list of domains that are considered local. For domains thus added, being on the list of local domains is equivalent to having a 0-preference MX record, with localhost as the MX value. If this is not the result you want, enable DontProbeInterfaces.

Because of cryptographic import restrictions in some countries, symmetric key cryptographic algorithms are limited to 128-bit if the SUNWcry package is not installed. The SUNWcry package is not included with the Solaris software. This package is available instead as a separate, controlled download.

Name setuname – change machine information

Synopsis

```
setuname [-t] [-n node] [-s name]
```

Description The `setuname` utility changes the parameter value for the system name and node name. Each parameter can be changed using `setuname` and the appropriate option.

Either or both the `-s` and `-n` options must be given when invoking `setuname`.

The system architecture may place requirements on the size of the system and network node name. The command will issue a fatal warning message and an error message if the name entered is incompatible with the system requirements.

Options The following options are supported:

`-n` *node* Changes the node name. *node* specifies the new network node name and can consist of alphanumeric characters and the special characters dash, underbar, and dollar sign.

`-s` *name* Changes the system name. *name* specifies new system name and can consist of alphanumeric characters and the special characters dash, underbar, and dollar sign.

`-t` Temporary change. No attempt will be made to create a permanent change.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `attributes(5)`

Notes `setuname` attempts to change the parameter values in two places: the running kernel and, as necessary per implementation, to cross system reboots. A temporary change changes only the running kernel.

Name sf880drd – Sun Fire 880 Dynamic Reconfiguration daemon

Synopsis

```
sf880drd
```

Description The Sun Fire 880 Dynamic Reconfiguration daemon, `sf880drd`, is part of the PCI and system bus hotplug framework. `sf880drd` starts at boot time. It has no configuration options and does not report any system status.

`sf880drd` implements the Sun Fire `880` console-less system administration (per-slot pushbuttons and LED status indicators). It also manages various aspects of CPU/memory hotplug.

Files `/usr/platform/SUNW,Sun-Fire-880/lib/sf880drd`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsfdr.u

See Also `svcs(1)`, `cfgadm(1M)`, `cfgadm_pci(1M)`, `cfgadm_sbd(1M)`, `svcadm(1M)`, `attributes(5)`, `smf(5)`

Notes The `sf880drd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/platform/sun4u/sf880drd
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name sftp-server – SFTP server subsystem

Synopsis

```
/usr/lib/ssh/sftp-server
```

Description `sftp-server` implements the server side of the SSH File Transfer Protocol as defined in the IETF `draft-ietf-secsh-filexfer`.

`sftp-server` is a subsystem for `sshd(1M)` and must not be run directly. There are no options or config settings.

To enable the `sftp-server` subsystem for `sshd` add the following to `/etc/ssh/sshd_config`:

```
Subsystem   sftp     /usr/lib/ssh/sftp-server
```

See `sshd_config(4)` for a description of the format and contents of that file.

There is no relationship between the protocol used by `sftp-server` and the FTP protocol (RFC 959) provided by `in.ftpd`.

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/usr/lib/ssh/sftp-server` Server-side binary

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsshdu
Interface Stability	Evolving

See Also `sftp(1)`, `ssh(1)`, `ssh-add(1)`, `ssh-keygen(1)`, `sshd(1M)`, `sshd_config(4)`, `attributes(5)`

To view license terms, attribution, and copyright for OpenSSH, the default path is `/var/sadm/pkg/SUNWsshdr/install/copyright`. If the Solaris operating environment has been installed anywhere other than the default, modify the given path to access the file at the installed location.

Author Markus Friedl

Name share – make local resource available for mounting by remote systems

Synopsis

```
share [-F FSType] [-o specific_options] [-d description]
      [pathname]
```

Description The `share` command exports, or makes a resource available for mounting, through a remote file system of type *FSType*. If the option `-F` *FSType* is omitted, the first file system type listed in `/etc/dfs/fstypes` is used as default. For a description of NFS specific options, see `share_nfs`(1M). *pathname* is the pathname of the directory to be shared. When invoked with no arguments, `share` displays all shared file systems.

Options

`-F` *FSType* — Specify the filesystem type.

`-o` *specific_options* — The *specific_options* are used to control access of the shared resource. (See `share_nfs`(1M) for the NFS specific options.) They may be any of the following:

`rw`	*pathname* is shared read/write to all clients. This is also the default behavior.
`rw=`*client*[`:`*client*]...	*pathname* is shared read/write only to the listed clients. No other systems can access *pathname*.
`ro`	*pathname* is shared read-only to all clients.
`ro=`*client*[`:`*client*]...	*pathname* is shared read-only only to the listed clients. No other systems can access *pathname*.

Separate multiple options with commas. Separate multiple operands for an option with colons. See EXAMPLES.

`-d` *description* — The `-d` flag may be used to provide a description of the resource being shared.

Examples

EXAMPLE 1 Sharing a Read-Only Filesystem

This line will share the `/disk` file system read-only at boot time.

```
share -F nfs -o ro /disk
```

EXAMPLE 2 Invoking Multiple Options

The following command shares the filesystem `/export/manuals`, with members of the netgroup having read-only access and users on the specified host having read-write access.

```
share -F nfs -o ro=netgroup_name,rw=host1:host2:host3 /export/manuals
```

Files

`/etc/dfs/dfstab`	list of `share` commands to be executed at boot time
`/etc/dfs/fstypes`	list of file system types, NFS by default
`/etc/dfs/sharetab`	system record of shared file systems

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also mountd(1M), nfsd(1M), share_nfs(1M), shareall(1M), unshare(1M), attributes(5)

Notes Export (old terminology): file system sharing used to be called exporting on SunOS 4.x, so the `share` command used to be invoked as exportfs(1B) or `/usr/sbin/exportfs`.

If `share` commands are invoked multiple times on the same filesystem, the last `share` invocation supersedes the previous—the options set by the last `share` command replace the old options. For example, if read-write permission was given to `usera` on `/somefs`, then to give read-write permission also to `userb` on `/somefs`:

```
example% share -F nfs -o rw=usera:userb /somefs
```

This behavior is not limited to sharing the root filesystem, but applies to all filesystems.

Name shareall, unshareall – share, unshare multiple resources

Synopsis

```
shareall [-F FSType [,FSType]...] [-| file]

unshareall [-F FSType [,FSType]...]
```

Description When used with no arguments, `shareall` shares all resources from `file`, which contains a list of `share` command lines. If the operand is a hyphen (–), then the `share` command lines are obtained from the standard input. Otherwise, if neither a `file` nor a hyphen is specified, then the file `/etc/dfs/dfstab` is used as the default.

Resources may be shared by specific file system types by specifying the file systems in a comma-separated list as an argument to `-F`.

`unshareall` unshares all currently shared resources. Without a `-F` flag, it unshares resources for all distributed file system types.

Options `-F` *FSType* Specify file system type. Defaults to the first entry in `/etc/dfs/fstypes`.

Files `/etc/dfs/dfstab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `share(1M)`, `unshare(1M)`, `attributes(5)`

Name share_nfs – make local NFS file systems available for mounting by remote systems

Synopsis `share [-d` *description*`] [-F nfs] [-o` *specific_options*`]` *pathname*

Description The `share` utility makes local file systems available for mounting by remote systems. It starts the `nfsd(1M)` and `mountd(1M)` daemons if they are not already running.

If no argument is specified, then `share` displays all file systems currently shared, including NFS file systems and file systems shared through other distributed file system packages.

Options The following options are supported:

`-d` *description*
Provide a comment that describes the file system to be shared.

`-F nfs`
Share NFS file system type.

`-o` *specific_options*
Specify *specific_options* in a comma-separated list of keywords and attribute-value-assertions for interpretation by the file-system-type-specific command. If *specific_options* is not specified, then by default sharing is read-write to all clients. *specific_options* can be any combination of the following:

`aclok`
Allows the NFS server to do access control for NFS Version 2 clients (running SunOS 2.4 or earlier). When `aclok` is set on the server, maximal access is given to all clients. For example, with `aclok` set, if anyone has read permissions, then everyone does. If `aclok` is not set, minimal access is given to all clients.

`anon=`*uid*
Set *uid* to be the effective user ID of unknown users. By default, unknown users are given the effective user ID `UID_NOBODY`. If *uid* is set to −1, access is denied.

`index=file`
Load `file` rather than a listing of the directory containing this file when the directory is referenced by an NFS URL.

`log=tag`
Enables NFS server logging for the specified file system. The optional tag determines the location of the related log files. The `tag` is defined in `etc/nfs/nfslog.conf`. If no `tag` is specified, the default values associated with the *global* `tag` in `etc/nfs/nfslog.conf` is used. Support of NFS server logging is only available for NFS Version 2 and Version 3 requests.

`nosub`
Prevents clients from mounting subdirectories of shared directories. For example, if `/export` is shared with the `nosub` option on server *fooey* then a NFS client cannot do:

```
mount -F nfs fooey:/export/home/mnt
```

NFS Version 4 does not use the `MOUNT` protocol. The `nosub` option only applies to NFS Version 2 and Version 3 requests.

`nosuid`
By default, clients are allowed to create files on the shared file system with the setuid or setgid mode enabled. Specifying `nosuid` causes the server file system to silently ignore any attempt to enable the setuid or setgid mode bits.

`public`
Moves the location of the public file handle from `root` (`/`) to the exported directory for WebNFS-enabled browsers and clients. This option does not enable WebNFS service; WebNFS is always on. Only one file system per server may use this option. Any other option, including the `-ro=list` and `-rw=list` options can be included with the `public` option.

`ro`
Sharing is read-only to all clients.

`ro=`*access_list*
Sharing is read-only to the clients listed in *access_list*; overrides the `rw` suboption for the clients specified. See *access_list* below.

`root=`*access_list*
Only root users from the hosts specified in *access_list* have root access. See *access_list* below. By default, no host has root access, so root users are mapped to an anonymous user ID (see the `anon=`*uid* option described above). Netgroups can be used if the file system shared is using UNIX authentication (`AUTH_SYS`).

`rw`
Sharing is read-write to all clients.

`rw=`*access_list*
Sharing is read-write to the clients listed in *access_list*; overrides the `ro` suboption for the clients specified. See *access_list* below.

`sec=`*mode*[`:`*mode*]. . .
Sharing uses one or more of the specified security modes. The *mode* in the `sec=`*mode* option must be a node name supported on the client. If the `sec=` option is not specified, the default security mode used is `AUTH_SYS`. Multiple `sec=` options can be specified on the command line, although each mode can appear only once. The security modes are defined in `nfssec(5)`.

Each `sec=` option specifies modes that apply to any subsequent `window=, rw, ro, rw=, ro=` and `root=` options that are provided before another `sec=`option. Each additional `sec=` resets the security mode context, so that more `window=, rw, ro, rw=, ro=` and `root=` options can be supplied for additional modes.

`sec=`*none*
If the option `sec=`*none* is specified when the client uses `AUTH_NONE`, or if the client uses a security mode that is not one that the file system is shared with, then the credential of each NFS request is treated as unauthenticated. See the `anon=`*uid* option for a description of how unauthenticated requests are handled.

`secure`
This option has been deprecated in favor of the `sec=`*dh* option.

`window=`*value*
When sharing with `sec=`*dh*, set the maximum life time (in seconds) of the RPC request's credential (in the authentication header) that the NFS server allows. If a credential arrives with a life time larger than what is allowed, the NFS server rejects the request. The default value is 30000 seconds (8.3 hours).

access_list The *access_list* argument is a colon-separated list whose components may be any number of the following:

hostname
The name of a host. With a server configured for DNS or LDAP naming in the `nsswitch` "hosts" entry, any hostname must be represented as a fully qualified DNS or LDAP name.

netgroup
A netgroup contains a number of hostnames. With a server configured for DNS or LDAP naming in the `nsswitch` "hosts" entry, any hostname in a netgroup must be represented as a fully qualified DNS or LDAP name.

domain name suffix
To use domain membership the server must use DNS or LDAP to resolve hostnames to IP addresses; that is, the "hosts" entry in the `/etc/nsswitch.conf` must specify "dns" or "ldap" ahead of "nis" or "nisplus", since only DNS and LDAP return the full domain name of the host. Other name services like NIS or NIS+ cannot be used to resolve hostnames on the server because when mapping an IP address to a hostname they do not return domain information. For example,

```
NIS or NIS+   172.16.45.9 --> "myhost"
```

and

```
DNS or LDAP   172.16.45.9 -->
     "myhost.mydomain.mycompany.com"
```

The domain name suffix is distinguished from hostnames and netgroups by a prefixed dot. For example,

```
rw=.mydomain.mycompany.com
```

A single dot can be used to match a hostname with no suffix. For example,

```
rw=.
```

matches ”mydomain” but not ”mydomain.mycompany.com”. This feature can be used to match hosts resolved through NIS and NIS+ rather than DNS and LDAP.

network
The network or subnet component is preceded by an at-sign (@). It can be either a name or a dotted address. If a name, it is converted to a dotted address by `getnetbyname(3SOCKET)`. For example,

```
=@mynet
```

would be equivalent to:

`=@172.16` or `=@172.16.0.0`

The network prefix assumes an octet-aligned netmask determined from the zeroth octet in the low-order part of the address up to and including the high-order octet, if you want to specify a single IP address (see below). In the case where network prefixes are not byte-aligned, the syntax allows a mask length to be specified explicitly following a slash (/) delimiter. For example,

`=@theothernet/17` or `=@172.16.132/22`

...where the mask is the number of leftmost contiguous significant bits in the corresponding IP address.

When specifying individual IP addresses, use the same @ notation described above, without a netmask specification. For example:

```
=@172.16.132.14
```

Multiple, individual IP addresses would be specified, for example, as:

```
root=@172.16.132.20:@172.16.134.20
```

A prefixed minus sign (–) denies access to that component of *access_list*. The list is searched sequentially until a match is found that either grants or denies access, or until the end of the list is reached. For example, if host ”terra” is in the ”engineering” netgroup, then

```
rw=-terra:engineering
```

denies access to `terra` but

```
rw=engineering:-terra
```

grants access to `terra`.

Operands The following operands are supported:

pathname
The pathname of the file system to be shared.

Examples

EXAMPLE 1 Sharing A File System With Logging Enabled

The following example shows the `/export` file system shared with logging enabled:

```
example% share -o log /export
```

The default global logging parameters are used since no tag identifier is specified. The location of the log file, as well as the necessary logging work files, is specified by the global entry in `/etc/nfs/nfslog.conf`. The `nfslogd(1M)` daemon runs only if at least one file system entry in `/etc/dfs/dfstab` is shared with logging enabled upon starting or rebooting the system. Simply sharing a file system with logging enabled from the command line does not start the `nfslogd(1M)`.

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Files

`/etc/dfs/fstypes`
: list of system types, NFS by default

`/etc/dfs/sharetab`
: system record of shared file systems

`/etc/nfs/nfslogtab`
: system record of logged file systems

`/etc/nfs/nfslog.conf`
: logging configuration file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfssu

See Also

`mount(1M)`, `mountd(1M)`, `nfsd(1M)`, `nfslogd(1M)`, `share(1M)`, `unshare(1M)`, `getnetbyname(3SOCKET)`, `nfslog.conf(4)`, `netgroup(4)`, `attributes(5)`, `nfssec(5)`

Notes

If the `sec=` option is presented at least once, all uses of the `window=, rw, ro, rw=, ro=` and `root=` options must come `after` the first `sec=` option. If the `sec=` option is not presented, then `sec=`*sys* is implied.

If one or more explicit `sec=` options are presented, *sys* must appear in one of the options mode lists for accessing using the `AUTH_SYS` security mode to be allowed. For example:

```
share -F nfs /var
share -F nfs -o sec=sys /var
```

grants read-write access to any host using AUTH_SYS, but

```
share -F nfs -o sec=dh /var
```

grants no access to clients that use AUTH_SYS.

Unlike previous implementations of share_nfs, access checking for the window=, rw, ro, rw=, and ro= options is done per NFS request, instead of per mount request.

Combining multiple security modes can be a security hole in situations where the ro= and rw= options are used to control access to weaker security modes. In this example,

```
share -F nfs -o sec=dh,rw,sec=sys,rw=hosta /var
```

an intruder can forge the IP address for hosta (albeit on each NFS request) to side-step the stronger controls of AUTH_DES. Something like:

```
share -F nfs -o sec=dh,rw,sec=sys,ro /var
```

is safer, because any client (intruder or legitimate) that avoids AUTH_DES only gets read-only access. In general, multiple security modes per share command should only be used in situations where the clients using more secure modes get stronger access than clients using less secure modes.

If rw=, and ro= options are specified in the same sec= clause, and a client is in both lists, the order of the two options determines the access the client gets. If client hosta is in two netgroups - group1 and group2 - in this example, the client would get read-only access:

```
share -F nfs -o ro=group1,rw=group2 /var
```

In this example hosta would get read-write access:

```
share -F nfs -o rw=group2,ro=group1 /var
```

If within a sec= clause, both the ro and rw= options are specified, for compatibility, the order of the options rule is not enforced. All hosts would get read-only access, with the exception to those in the read-write list. Likewise, if the ro= and rw options are specified, all hosts get read-write access with the exceptions of those in the read-only list.

The ro= and rw= options are guaranteed to work over UDP and TCP but may not work over other transport providers.

The root= option with AUTH_SYS is guaranteed to work over UDP and TCP but may not work over other transport providers.

The root= option with AUTH_DES is guaranteed to work over any transport provider.

There are no interactions between the root= option and the rw, ro, rw=, and ro= options. Putting a host in the root list does not override the semantics of the other options. The access the host gets is the same as when the root= options is absent. For example, the following share command denies access to hostb:

```
share -F nfs -o ro=hosta,root=hostb /var
```

The following gives read-only permissions to `hostb`:

```
share -F nfs -o ro=hostb,root=hostb /var
```

The following gives read-write permissions to `hostb`:

```
share -F nfs -o ro=hosta,rw=hostb,root=hostb /var
```

If the file system being shared is a symbolic link to a valid pathname, the canonical path (the path which the symbolic link follows) are shared. For example, if `/export/foo` is a symbolic link to `/export/bar` (`/export/foo -> /export/bar`), the following `share` command results in `/export/bar` as the shared pathname (and not `/export/foo`).

```
example# share -F nfs /export/foo
```

An NFS mount of `server:/export/foo` results in `server:/export/bar` really being mounted.

This line in the `/etc/dfs/dfstab` file shares the `/disk` file system read-only at boot time:

```
share -F nfs -o ro /disk
```

The same command entered from the command line does not share the `/disk` file system unless there is at least one file system entry in the `/etc/dfs/dfstab` file. The `mountd(1M)` and `nfsd(1M)` daemons only run if there is a file system entry in `/etc/dfs/dfstab` when starting or rebooting the system.

The `mountd(1M)` process allows the processing of a path name the contains a symbolic link. This allows the processing of paths that are not themselves explicitly shared with `share_nfs`. For example, `/export/foo` might be a symbolic link that refers to `/export/bar` which has been specifically shared. When the client mounts `/export/foo` the `mountd` processing follows the symbolic link and responds with the `/export/bar`. The NFS Version 4 protocol does not use the `mountd` processing and the client's use of `/export/foo` does not work as it does with NFS Version 2 and Version 3 and the client receives an error when attempting to mount `/export/foo`.

Name showmount – show remote mounts

Synopsis

```
/usr/sbin/showmount [-ade] [hostname]
```

Description `showmount` lists the clients that have remotely mounted a filesystem from *host*. This information is maintained by the `mountd(1M)` server on *host*, and is saved across crashes in the file `/etc/rmtab`. The default value for *host* is the value returned by `hostname(1)`.

The `showmount` command does not display the names of NFS Version 4 clients.

Options

`-a` Print all remote mounts in the format:

```
hostname : directory
```

where `hostname` is the name of the client, and `directory` is the root of the file system that has been mounted.

`-d` List directories that have been remotely mounted by clients.

`-e` Print the list of shared file systems.

Files `/etc/rmtab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfscu

See Also `hostname(1)`, `mountd(1M)`, `attributes(5)`

Solaris 10 Installation Guide: Basic Installations

Bugs If a client crashes, its entry will not be removed from the list of remote mounts on the server.

Name showrev – show machine, software revision, and patch revision information

Synopsis

```
/usr/bin/showrev [-a] [-p | -p -R root_path] [-w]
     [-c command] [-s hostname]
```

Description `showrev` displays revision information for the current hardware and software. With no arguments, `showrev` shows the system revision information including hostname, hostid, release, kernel architecture, application architecture, hardware provider, domain, and kernel version.

If a command is supplied with the `-c` option, `showrev` shows the `PATH` and `LD_LIBRARY_PATH` and finds out all the directories within the `PATH` that contain it. For each file found, its file type, revision, permissions, library information, and checksum are printed as well.

Options The following options are supported:

`-a` Print all system revision information available. Window system and patch information are added.

`-c` *command* Print the revision information about *command*.

`-p` Print only the revision information about patches.

`-R` *root_path* Define the full path name of a directory to use as the *root_path*. By specifying the root path, `showrev` retrieves the revision information about the patch from package system information files located under a directory tree starting at *root_path*. The *root_path* can be specified when retrieving installed patch information in a client from a server, for example, `/export/root/client1`.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-s` *hostname* Perform this operation on the specified *hostname*. The `-s` operation completes correctly only when *hostname* is running Solaris 2.5 or compatible versions.

`-w` Print only the OpenWindows revision information.

Output Varies, based on flags passed. If no flags are passed, output similar to the following appears:

```
 Hostname: system1
Hostid: 7233808e
Release: 5.10
Kernel architecture: sun4u
Application architecture: sparc
Hardware provider: Sun_Microsystems
Domain: a.network.COM
Kernel version: SunOS 5.10 generic
```

Exit Status The following error values are returned:

`0` Successful completion.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmc

See Also `arch(1)`, `ldd(1)`, `mcs(1)`, `sum(1)`, `patchadd(1M)`, `attributes(5)`

Bugs For the `-s` option to work when `hostname` is running a version of Solaris prior to 2.5, the Solstice AdminSuite must be installed on `hostname`.

Name shutdown – shut down system, change system state

Synopsis

```
/usr/sbin/shutdown [-y] [-g grace-period] [-i init-state]
     [message]
```

Description `shutdown` is executed by the super user to change the state of the machine. In most cases, it is used to change from the multi-user state (state 2) to another state.

By default, `shutdown` brings the system to a state where only the console has access to the operating system. This state is called single-user.

Before starting to shut down daemons and killing processes, `shutdown` sends a warning message and, by default, a final message asking for confirmation. *message* is a string that is sent out following the standard warning message "The system will be shut down in . . ." If the string contains more than one word, it should be contained within single (') or double (") quotation marks.

The warning message and the user provided *message* are output when there are 7200, 3600, 1800, 1200, 600, 300, 120, 60, and 30 seconds remaining before `shutdown` begins. See `EXAMPLES`.

System state definitions are:

state 0 — Stop the operating system.

state 1 — State 1 is referred to as the administrative state. In state 1 file systems required for multi-user operations are mounted, and logins requiring access to multi-user file systems can be used. When the system comes up from firmware mode into state 1, only the console is active and other multi-user (state 2) services are unavailable. Note that not all user processes are stopped when transitioning from multi-user state to state 1.

state s, S — State s (or S) is referred to as the single-user state. All user processes are stopped on transitions to this state. In the single-user state, file systems required for multi-user logins are unmounted and the system can only be accessed through the console. Logins requiring access to multi-user file systems cannot be used.

state 5 — Shut the machine down so that it is safe to remove the power. Have the machine remove power, if possible. The `rc0` procedure is called to perform this task.

state 6 — Stop the operating system and reboot to the state defined by the `initdefault` entry in `/etc/inittab`. The `rc6` procedure is called to perform this task.

Options

`-y` — Pre-answer the confirmation question so the command can be run without user intervention.

`-g` *grace-period* — Allow the super user to change the number of seconds from the 60-second default.

`-i` *init-state* If there are warnings, *init-state* specifies the state `init` is to be in. By default, system state 's' is used.

Examples

EXAMPLE 1 Using `shutdown`

In the following example, `shutdown` is being executed on host `foo` and is scheduled in 120 seconds. The warning message is output 2 minutes, 1 minute, and 30 seconds before the final confirmation message.

```
example# shutdown -i S -g 120 "===== disk replacement ====="
Shutdown started.   Tue Jun   7  14:51:40 PDT  1994

Broadcast Message from root (pts/1) on foo Tue Jun  7 14:51:41. . .
The system will be shut down in 2 minutes
===== disk replacement =====
Broadcast Message from root (pts/1) on foo Tue Jun  7 14:52:41. . .
The system will be shut down in 1 minutes
===== disk replacement =====
Broadcast Message from root (pts/1) on foo Tue Jun  7 14:53:41. . .
The system will be shut down in 30 seconds
===== disk replacement =====
Do you want to continue? (y or n):
```

Files

`/etc/inittab` controls process dispatching by `init`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`boot(1M)`, `halt(1M)`, `init(1M)`, `killall(1M)`, `reboot(1M)`, `ufsdump(1M)`, `init.d(4)`, `inittab(4)`, `nologin(4)`, `attributes(5)`

Notes

When a system transitions down to the `S` or `s` state, the `/etc/nologin` file (see `nologin(4)`) is created. Upon subsequent transition to state 2 (multi-user state), this file is removed by a script in the `/etc/rc2.d` directory.

Name slpd – Service Location Protocol Daemon

Synopsis `/usr/lib/inet/slpd [-f` *configuration-file*`]`

Description The `slpd` daemon provides common server functionality for the Service Location Protocol ("SLP") versions 1 and 2, as defined by IETF in *RFC 2165* and *RFC 2608*. SLP provides a scalable framework for the discovery and selection of network services.

`slpd` provides the following framework services:

Directory Agent	This service automatically caches service advertisements from service agents to provide them to user agents, and makes directory agent advertisements of its services. This service is optional. `slpd` does not provide directory agent service by default. Directory agents are not databases, and they do not need to be maintained.
Service Agent Server	All service agents on the local host register and deregister with this server. This service responds to all requests for services, and forwards registrations to directory agents. By default, `slpd` is a service agent server.
Passive Directory Agent Discovery	This service listens for directory agent advertisements and maintains a table of active directory agents. When a user agent wishes to discover a directory agent, it can simply query `slpd`, obviating the need to perform discovery by means of multicast. By default, `slpd` performs this service.
Proxy Registration	This service can act as a proxy service agent for services that cannot register themselves. `slpd` reads the proxy registration file for information on services it is to proxy. By default, no services are registered by proxy.

All configuration options are available from the configuration file. `slpd` reads its configuration file upon startup.

Stop and start the slpd daemon using `svcadm(1M)`. Use the command `svcadm enable network/slp` to start the slpd daemon. Use the command `svcadm disable network/slp` to stop it.

The file `/etc/inet/slp.conf` must exist before the `slp` service can start the daemon. Only the example file `/etc/inet/slp.conf.example` is present by default. To enable SLP, copy `/etc/inet/slp.conf.example` to `/etc/inet/slp.conf`.

Options The following options are supported:

`-f` *configuration-file* Specify an alternate configuration file

Examples **EXAMPLE 1** Stopping the `slpd` daemon

The following command stops the `slpd` daemon:

```
example# svcadm disable network/slp
```

EXAMPLE 2 Restarting the `slpd` daemon

The following command restarts the `slpd` daemon:

```
example# svcadm restart network/slp
```

Files `/etc/inet/slp.conf` The default configuration file

`slpd.reg` The proxy registration file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWslpu, SUNWslpr
CSI	Enabled
Interface Stability	Committed

See Also `svcs(1)`, `svcadm(1M)`, `slp_api(3SLP)`, `slp.conf(4)`, `slpd.reg(4)`, `attributes(5)`, `smf(5)`, `slp(7P)`

System Administration Guide: Network Services

Guttman, E., Perkins, C., Veizades, J., and Day, M., *RFC 2608, Service Location Protocol, Version 2*, The Internet Society, June 1999.

Notes The `slpd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/slp
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name smartcard – configure and administer a smart card

Synopsis

```
smartcard -c admin [-a application] [propertyname]...

smartcard -c admin [-a application]
     [-x { add|delete|modify } propertyname=value...]

smartcard -c admin -t service -j classname -x
     { add|delete|modify}

smartcard -c admin -t terminal
     { -j classname | -H libraryname } -d device -r userfriendlyreadername -n readername -x
     { add|delete|modify }
     [-R]

smartcard -c admin -t debug -j classname -l level -x
     { add|delete|modify}

smartcard -c admin -t override -x { add|delete|modify} propertyname=value

smartcard -c admin -I -k keytype -i filename

smartcard -c admin -E -k keytype -o filename

smartcard -c load -A aid [-r userfriendlyreadername] -P pin
     [-s slot] [-i inputfile] [-p propfile] [-v]
     [propertyname=value]...

smartcard -c load -u -P pin [-A aid]
     [-r userfriendlyreadername] [-s slot] [-v]

smartcard -c bin2capx -T cardname [-i inputfile]
     [-o outputfile] [-p propfile] [-I anothercapxfile]
     [-v] [propertyname=value]...

smartcard -c init -A aid [-r readername] [-s slot] -L

smartcard -c init -A aid [-r readername] -P pin [-s slot]
     [propertyname=value]...

smartcard -c enable

smartcard -c disable
```

Description The `smartcard` utility is used for all configurations related to a smart card. It comprises the following subcommands:

1. Administration of OCF properties. (`-c admin`)

   This subcommand is used to list and modify any of the OCF properties. With no arguments it will list all the current properties. It can only be executed by root. Some OCF properies are:

   `defaultcard`
   # default card for an application

```
defaultreader
    # default reader for an application

authmechanism
    # authentication mechanism

validcards
    # list of cards valid for an application
```

A complete listing can be obtained by using the `smartcard` utility as described in the EXAMPLES section.

2. Loading and Unloading of applets from the smart card (`-c load`) and performing initial configuration of a non-Java card.

   This subcommand administers the applets or properties on a smartcard. It can be used to load or unload applets and/or properties to and from a smart card. The applet is a Java class file that has been run through a converter to make the byte code JavaCard-compliant. This command can be used to load both an applet file in the standard format or a file converted to the `capx` format. If no `-r` option is specified, the loader tries to load to any connected reader, provided it has already been inserted using the `smartcard -c admin` command.

3. Converting card applets or properties to the `capx` format (`-c bin2capx`)

   This subcommand is used to convert a Java card applet or properties into a new format called `capx` before downloading it onto the smart card. Converting to this format enables the applet developer to add applet-specific information that is useful during the downloading process and identifies the applet.

   In the following example,

   ```
   smartcard -c bin2capx -i cyberflex.bin \
   -T CyberFlex aidto-000102030405060708090A0B0C0D0E0F fileID=2222 \
   instanceID=2223 and more.
   ```

   if no output file is specified, a default file with the name *input_filename*.`capx` is created in the current directory. The mandatory `-T` option requires the user to specify the card name for which the `capx` file is being generated.

   The following example

   ```
   smartcard -c bin2capx -T IButton
   ```

   tells the loader that the `capx` file contains the binary for `IButton`. A single `capx` file can hold binaries for multiple cards (1 per card.) Users can, for example, hold binary files for both CyberFlex and IButton in the same capx file as follows:

   ```
   smartcard -c bin2capx -T IButton -i IButton.jib -o file.capx
   ```

   In the following example,

   ```
   smartcard -c bin2capx -T CyberFlex -i cyberflex.bin \
   -l file.capx -o file.capx
   ```

the `-l` option is used to provide an already-generated `capx` file. The output is directed to the same `capx` file, resulting in `capx` file holding binaries for both cards.

4. Personalizing a smart card (`-c init`)

   This subcommand is used to set user-specific information required by an applet on a smart card. For example, the Sun applet requires a user name to be set on the card. This subcommand is also used to personalize information for non-Java cars.

5. Enabling and disabling the smart card desktop login (`-c {enable | disable}`)

Options The following options are supported:

`-a` *application*
Specify application name for the configuration parameter. Parameters may differ depending on the application. If no application name is specified, then `ocf` is the default application.

`-A` *aid*
Specify a unique alphanumeric string that identifies the applet. The *aid* argument must be a minimum of 5 characters and can be a maximum of 16 characters in length. If an applet with an identical *aid* already exists on the card, a load will result in an error.

`-c`
Specify subcommand name. Valid options are: `admin`, `load`, `bin2capx`, `init`, `enable`, and `disable`.

`-d` *device*
Specify device on which the reader is connected (for example, `/dev/cua/a`).

`-D`
Disable a system from using smart cards.

`-E`
Export the keys to a file.

`-H` *libraryname*
Specify the full path of the IFD handler library for the reader.

`-i` *filename*
Specify input file name.

`-I`
Import from a file.

`-j` *classname*
Specify fully-qualified class name.

`-k` *keytype*
Specify type of key (for example, `challenge_response`, `pki`.)

`-l`
Specify debug level (0–9), signifying level of debug information displayed.

`-L`
List all properties configurable in an applet.

`-n` *readername*
Specify reader name as required by the driver.

`-o` *filename*
Specify output file name.

`-p` *propfile*
Specify properties file name. This file could contain a list of property names and value pairs, in the format *propertyname=value*.

`-P` *pin*
Specify pin used to validate to the card.

`-r` *userfriendlyreadername*
Specify user-defined reader name where the card to be initialized is inserted.

`-R`
Restart the `ocf` server.

`-s` *slot*
Specify slot number. If a reader has multiple slots, this option specifies which slot to use for initialization. If a reader has only one slot, this option is not required. If no slot number is specified, by default the first slot of the reader is used.

`-t`
Specify type of property being updated. The valid values are:

`service`
Updating a card service provider details.

`terminal`
Updating a card reader provider details.

`debug`
OCF trace level.

`override`
Override a system property of the same name.

`-T` *cardname*
Specify card name.

`-u`
Unload the applet specified by the application ID from the card. If no application ID is specified, all applets are unloaded from the card.

`-v`
Verbose mode (displays helpful messages).

`-x`
Specify action to be taken. Valid values are: `add`, `delete`, or `modify`.

Examples

EXAMPLE 1 Viewing the Values of All Properties

Enter the following command to view the values of all the properties that are set:

```
% smartcard -c admin
```

EXAMPLE 2 Viewing the Values of Specific Properties

Enter the following command to view the values of specific properties:

```
% smartcard -c admin language country
```

EXAMPLE 3 Adding a Card Service

Enter the following command to add a card service factory for a CyberFlex card, available in the package `com.sun.services.cyberflex`, to the properties:

```
% smartcard -c admin -t service \
     -j com.sun.services.cyberflex.CyberFlexCardServiceFactory -x add
```

EXAMPLE 4 Adding a Reader

Enter the following command to add the IFD handler for the internal reader:

```
% smartcard -c admin -t terminal \
     -H /usr/lib/smartcard/ifdh_scmi2c.so -x add \
     -d /dev/scmi2c0 -r MyInternalReader -n SunISCRI
```

EXAMPLE 5 Deleting a Reader

Enter the following command to delete the SCM reader, added in the previous example, from the properties:

```
% smartcard -c admin -t terminal -r SCM -x delete
```

EXAMPLE 6 Changing the Debug Level

Enter the following command to change the debug level for all of the com.sun package to 9:

```
% smartcard -c admin -t debug -j com.sun -l 9  -x modify
```

EXAMPLE 7 Setting the Default Card for an Application

Enter one of the following commands to set the default card for an application (`dtlogin`) to be CyberFlex.

If the property default card does not exist, enter the following command:

```
% smartcard -c admin -a dtlogin -x add defaultcard=CyberFlex
```

If the property default card exists, enter the following command:

EXAMPLE 7 Setting the Default Card for an Application *(Continued)*

```
% smartcard -c admin -a dtlogin -x modify defaultcard=CyberFlex
```

EXAMPLE 8 Exporting Keys for a User into a File

Enter the following command to export the challenge-response keys for a user into a file:

```
% smartcard -c admin -k challenge_response -E -o /tmp/mykeys
```

EXAMPLE 9 Importing Keys from a File

Enter the following command to import the challenge-response keys for a user from a file:

```
% smartcard -c admin -k challenge_response -I -i /tmp/mykeys
```

EXAMPLE 10 Downloading an Applet into a Java Card

Enter the following command to download an applet into a Java card or to configure a PayFlex (non-Java) card inserted into an SCM reader for the capx file supplied in the `/usr/share/lib/smartcard` directory:

```
% smartcard -c load -r SCM \
     -i /usr/share/lib/smartcard/SolarisAuthApplet.capx
```

EXAMPLE 11 Downloading an Applet Binary

Enter the following command to download an applet binary from some place other that the capx file supplied with Solaris 8 into an IButton (the aid and input file are mandatory, the remaining parameters are optional):

```
% smartcard -c load -A A000000062030400  -i newapplet.jib
```

EXAMPLE 12 Downloading an Applet on a CyberFlex Access Card

On a CyberFlex Access Card, enter the following command to download an applet newapplet.bin at fileID 2222, instanceID 3333 using the specified verifyKey and a heap size of 2000 bytes:

```
% smartcard -c load -A newaid -i newapplet.bin \
     fileID=2222 instanceID=3333 verifyKey=newKey \
     MAC=newMAC heapsize=2000
```

EXAMPLE 13 Configuring a PayFlex Card

Enter the following command to configure a PayFlex (non-Java) card with specific aid, transport key, and initial pin:

```
% smartcard -c load -A A00000006203400 \
     pin=242424246A617661 transportKey=4746584932567840
```

EXAMPLE 14 Unloading an Applet from a Card

Enter the following command to unload an applet from iButton:

```
% smartcard -c load -u
```

EXAMPLE 15 Displaying Usage of `smartcard -c load`

Enter the following command to display the usage of the `smartcard -c load` command:

```
% smartcard -c load
```

EXAMPLE 16 Displaying All Configurable Parameters for an Applet

Enter the following command to display all the configurable parameters for an applet with aid 123456 residing on a card inserted into an SCM reader:

```
% smartcard -c init -r SM -A 123456 -L
```

EXAMPLE 17 Changing the PIN

Enter the following command to change the pin for the SolarisAuthApplet residing on a card or to change the PIN for a PayFlex (non-Java) card inserted into an SCM reader:

```
% smartcard -c init -A A000000062030400 -P oldpin pin=newpin
```

EXAMPLE 18 Displaying All Configurable Parameters for the SolarisAuthApplet.

Enter the following command to display all the configurable parameters for the SolarisAuthApplet residing on a card inserted into an SCM reader:

```
% smartcard -c init -A A000000062030400 -L
```

EXAMPLE 19 Setting a Property to a Value on a smart card

Enter the following command to set properties called `user` to the value `james` and `application` to the value `login` on a card inserted into an SCM reader that has a pin `testpin`:

```
% smartcard -c init -A A000000062030400 -r CyberFlex -P testpin \
     application=login user=james
```

EXAMPLE 20 Converting an Applet for the CyberFlex Card into `capx` Format.

Enter the following command to convert an applet for the CyberFlex card into the `capx` format required for downloading the applet into the card:

```
% smartcard -c bin2capx \
     -i /usr/share/lib/smartcard/SolarisAuthApplet.bin \
     -T CyberFlex -o /home/CorporateCard.capx -v memory=128 heapsize=12
```

EXAMPLE 21 Converting an Applet for the IButton Card into `capx` Format

Enter the following command to convert an applet for the IButton card into the `capx` format required for downloading the applet into the button:

EXAMPLE 21 Converting an Applet for the IButton Card into capx Format *(Continued)*

```
% smartcard -c bin2capx \
     -i /usr/share/lib/smartcard/SolarisAuthApplet.jib \
     -T IButton -o /home/CorporateCard.capx -v
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`1`
An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWocf
Interface Stability	Stable

See Also

`ocfserv(1M)`, `attributes(5)`, `smartcard(5)`

Notes

The command line options contain only alphanumeric input.

Name smattrpop – populate security attribute databases in a name service

Synopsis

```
smattrpop [-c ] [-f] [-m] [-p policy] [-r] -s scope -t scope
     [-v] database
```

Description The `smattrpop` command updates the `auth_attr(4)`, `exec_attr(4)`, `prof_attr(4)`, and `user_attr(4)` role-based access control databases in a target NIS, NIS+, LDAP, or local `/etc` files name service from the corresponding databases in a source name service or files.

This command processes the table entries from the source database and merges each source entry field into the same field in the corresponding table entry in the target database. If a source entry does not exist in the target database, the entry is created. If the source entry exists in the target database, the fields are merged or replaced according to the command options.

Any errors encountered while updating the target entry are reported to `stdout`, and the command continues with the next source database entry.

Options The following options are supported:

`-c` Performs cross-table checking. If you specify this option and a check error occurs, a message identifying the check error is written to `stdout`.

The target entry values are checked against entries in related databases:

- `auths` values — Each value must exist as the name of an authorization in the `auth_attr(4)` database.
- `profiles` values — Each value must exist as a name of a profile in the `prof_attr(4)` database.
- `roles` values — Each value must exist as the name of a role identity in the `user_attr(4)` database.
- For each `exec_attr(4)` entry in the source database, the name must exist as the name of a `profile` in the `prof_attr`(4) database.

`-f` Specifies that the value in each field in the source entry replaces the value in the corresponding field in the target entry, if the source entry field has a non-empty value.

`-m` For the `auths`, `profiles`, and `roles` attributes, specifies that the values in each field in the source entry are merged with the values in the corresponding target entry field. If a source value does not exist in the target field, the value is appended to the set of target values. If the target field is empty, the source values replace the target field. The attribute values that merge depend on the database being updated:

- `prof_attr`(4) — the `auths` and `profiles` attribute values are merged.
- `user_attr`(4) — the `auths`, `profiles`, and `roles` attribute values are merged.
- `exec_attr`(4) — the `uid`, `gid`, `euid`, and `egid` values are merged.

`-p` *policy* Specifies the value of the policy field in the `exec_attr`(4) database. Valid values are `suser` (standard Solaris superuser) and `tsol` (Trusted Solaris). If you specify this option, only the entries in the source `exec_attr` database with the specified policy are processed. If you omit this option, all entries in the source `exec_attr` database are processed.

`-r` Specifies that role identities in the `user_attr`(4) database in the source name service are processed. If you omit this option, only the normal user entries in the `user_attr` source database are processed.

`-s` *scope* Specifies the source name service or local file directory for database updates, using the following syntax:

type`:`*/server*/*domain*

where *type* indicates the type of name service. Valid values for *type* are:

- `file` — local files
- `nis` — NIS name service
- `nisplus` — NIS+ name service
- `ldap` — LDAP name service

server indicates the local host name of the Solaris system on which the `smattrpop` command is executed, and on which both the source and target databases exist.

domain specifies the management domain name for the name service.

You can use two special cases of *scope* values:

- To indicate the databases in the `/etc/security` local system directory, use the scope `file`:*/server*, where *server* is the name of the local system.
- To load from databases in an arbitrary directory on the Solaris server, use the scope `file`:*/server*/*pathname*, where where *server* is the name of the local system and *pathname* is the fully-qualified directory path name to the database files.

`-t` *scope* Specifies the target name service or local file directory for database updates, using the following syntax:

type`:`*/server*/*domain*

where *type* indicates the type of name service. Valid values for *type* are:

- `file` — local files
- `nis` — NIS name service
- `nisplus` — NIS+ name service
- `ldap` — LDAP name service

server indicates the local host name of the Solaris system on which the `smattrpop` command is executed, and on which both the source and target databases exist.

domain specifies the management domain name for the name service.

You can use two special cases of *scope* values:

- To indicate the databases in the `/etc/security` local system directory, use the scope `file:`/*server*, where *server* is the name of the local system.
- To update to databases in an arbitrary directory on the Solaris server, use the scope `file:`/*server*/*pathname*, where where *server* is the name of the local system and *pathname* is the fully-qualified directory path name to the database files.

`-v` Specifies that verbose messages are written. A message is written to `stdout` for each entry processed.

Operands The following operands are supported:

database Populates one or all databases. You can specify either the name of the database you want to process (for example, `auth_attr`), or `all` to process all databases. If you specify `all`, the databases are processed in the following order:

1. `auth_attr`(4)
2. `prof_attr`(4)
3. `exec_attr`(4)
4. `user_attr`(4)

Examples **EXAMPLE 1** Populating all tables in the NIS name service

The following example merges the values from all four attribute databases in the `/etc/security` directory of the local system into the corresponding tables in the NIS domain, `east.example.com`. The command is executed on the master server, `hoosier`, for the NIS domain and the source files are in the `/etc` and `/etc/security` directories on the NIS master server. No cross-table checking is performed. A summary message indicating the number of entries processed and updated for each table is written to `stdout`.

```
/usr/sadm/bin/smattrpop -s file:/hoosier \
          -t nis:/hoosier/east.example.com all
```

EXAMPLE 2 Updating the authorization table in the NIS+ name service

This example merges new authorization data from a local system file in the `auth_attr` text format into the existing `auth_attr` database in the NIS+ domain, `east.example.com`. The command is executed on the NIS+ master server, `foobar`. Values from the source `auth_attr` file replace the corresponding field values in the `NIS+` tables for each entry. A message is written to `stdout` for each entry processed. Database cross-checking is performed and any check error is written to `stdout`. A summary message indicating the number of entries processed and updated for the `auth_attr` database is written to `stdout`.

```
/usr/sadm/bin/smattrpop -c -f -v -s file:/foobar/var/temp \
          -t nisplus:/foobar/East.Sun.COM auth_attr
```

Environment Variables See environ(5) for a description of the JAVA_HOME environment variable, which affects the execution of the smattrpop command. If this environment variable is not specified, the /usr/java location is used. See smc(1M).

Exit Status Any errors encountered while updating the target entry are reported to stdout. The following exit values are returned:

0 The specified tables were updated. Individual entries may have encountered checking errors.

1 A syntax error occurred in the command line.

2 A fatal error occurred and the tables were not completely processed. Some entries may have been updated before the failure.

Files

/etc/security/auth_attr	Authorization description database. See auth_attr(4).
/etc/security/exec_attr	Execution profiles database. See exec_attr(4).
/etc/security/prof_attr	Profile description database. See prof_attr(4).
/etc/user_attr	Extended user attribute database. See user_attr(4).

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga

See Also smc(1M), smexec(1M), smprofile(1M), auth_attr(4), exec_attr(4), prof_attr(4), user_attr(4), attributes(5), environ(5)

Name smbios – display the contents of a System Management BIOS image

Synopsis `smbios [-BeOsx] [-i id] [-t type] [-w file] [file]`

Description The `smbios` utility displays the contents of the System Management BIOS (SMBIOS) image exported by the current system or stored in a file. SMBIOS is an industry-standard mechanism for low-level system software to export hardware configuration information to higher-level system management software. The SMBIOS data format itself is defined by the Distributed Management Task Force (DMTF). Refer to http://www.dmtf.org for more information about SMBIOS and to obtain a copy of the SMBIOS specification and implementation guidelines.

The SMBIOS image consists of a table of structures, each describing some aspect of the system software or hardware configuration. By default, `smbios` displays the entire contents of the current SMBIOS image. If the `-s` option is specified, `smbios` displays a summary of the structures that are present in the image. If the `-w` option is specified, `smbios` writes a copy of the SMBIOS image to the specified file. `smbios` can then be applied to the resulting file to display its content.

`smbios` attempts to display each structure and its content in a human- readable fashion. If `smbios` does not recognize a structure's type or content, the raw hexadecimal data for the structure is displayed.

Options The following options are supported:

`-B` Disable header validation for broken BIOSes.

By default, `smbios` attempts to validate the SMBIOS header by verifying the anchor strings, header checksums, and version number. This option might be necessary when a BIOS has a non-compliant header.

`-e` Display the contents of the SMBIOS entry point rather than the contents of the SMBIOS structure table.

`-i` *id* Display only the specified structure, named by its integer id.

`-O` Display obsolete structure types.

By default, `smbios` elides output for structures whose type is marked as obsolete in the DMTF SMBIOS specification.

`-s` Display only a summary listing of the structure identifiers and types, instead of the content of each selected structure.

`-t` *type* Display only those structures whose type matches the specified integer type, as defined the DMTF SMBIOS specification.

`-w` *file* Write a copy of the SMBIOS image to the specified file and exit.

The SMBIOS entry point is written to the start of the file with its structure table address set to the file offset of the structure table, and a new entry point checksum is computed.

`-x` Display raw hexadecimal data for the selected structures in addition to human-readable output.

By default, hexadecimal data is only displayed if `smbios` cannot display human-readable output for the selected structures.

Operands The following operands are supported:

file Specifies an alternate SMBIOS image to display instead of the current system's SMBIOS image.

Exit Status The following exit values are returned:

`0` Successful completion. All structures in the SMBIOS image were examined successfully.

`1` A fatal error occurred, such as failure to open the specified file or device, or corruption in the image.

`2` Invalid command-line options were specified.

Files `/dev/smbios` Kernel SMBIOS image device. This device special file is used to export a snapshot of the current system SMBIOS image.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	See below.

The command-line options are Evolving. The human-readable output is Unstable.

See Also `prtdiag(1M)`, `attributes(5)`, `smbios(7D)`

System Management BIOS Reference Specification (see http://www.dmtf.org)

Notes The implementation of a System Management BIOS image is entirely at the discretion of the system and BIOS vendors. Not all systems export an SMBIOS. The SMBIOS structure content varies widely between systems and BIOS vendors and frequently does not comply with the guidelines included in the specification. Some structure fields might not be filled in by the BIOS at all, and others might be filled inwith non-conforming values.

Name smc – start the Solaris Management Console

Synopsis

```
smc [ subcommand] [ args]

smc [ subcommand] [ args] -T tool_name
     [ - – tool_args]
```

Description The `smc` command starts the Solaris Management Console. The Solaris Management Console is a graphical user interface that provides access to Solaris system administration tools. It relies on Solaris Management Console servers running on one or more computers to perform modifications and report data. Each of these servers is a repository for code which the console can retrieve after the user of the console has authenticated himself or herself to the server.

The console can also retrieve toolboxes from the server. These toolboxes are descriptions of organized collections of tools available on that and possibly other servers. Once one of these toolboxes is loaded, the console will display it and the tools referenced in it.

The console can also run in a terminal (non-graphically), for use over remote connections or non-interactively from a script.

For information on the use of the graphical console, and for more detailed explanations of authentication, tools, and toolboxes, please refer to the Solaris Management Console online help available under the "Help" menu in the Solaris Management Console. To enable an NIS/NIS+ map to be managed from the Solaris Management Console, you must use the `smc edit` command to create a new toolbox for that map and enter the information about your NIS/NIS+ server where necessary. For instructions on creating a new toolbox, in the Solaris Management Console Help menu, select "Contents," then "About the Solaris Management Console Editor," then "To Create a Toolbox."

Subcommands The `smc` *subcommand*s are:

`open` The default subcommand for the Solaris Management Console is `open`. This will launch the console and allow you to run tools from the toolboxes you load. It does not need to be specified explicitly on the command line.

`edit` The `edit` subcommand will also launch the console, like the `open` subcommand. However, after loading a toolbox, you will not be able to run the referenced tools. Instead, you will be able to edit that toolbox, that is, add, remove, or modify any tools or folders in that toolbox.

SMF Administration The Solaris Management Console is implemented as a method that is managed by the service management facility (SMF) (see `smf(5)`), under the fault management resource identifier (FMRI):

```
svc:/application/management/wbem:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`.

The configuration properties of this service can be modified with svccfg(1M).

Through svcadm, the Solaris Management Console supports the following actions:

`start` Starts the CIM Object Manager (CIMOM) and Solaris Management Console server on the local host.

`stop` Stops the CIMOM and Solaris Management Console server on the local host.

`status` Gets the status of the CIMOM and Solaris Management Console server on the local host.

Controlling Remote Access

The Solaris Management Console supports an SMF property that controls remote access to WBEM-based applications, which include the Solaris Management Console. The property, `options/tcp_listen`, has default value of `false`, which disallows remote access. The value `true` allows remote access. See EXAMPLES.

Options

The following options are supported. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- -domain` with the *domain* argument.

If *tool_args* are specified, they must be preceded by the `- -` option and separated from the double dashes by a space.

`- -auth-data` *13;file*

Specifies a file which the console can read to collect authentication data. When running the Solaris Management Console non-interactively, the console will still need to authenticate itself with the server to retrieve tools. This data can either be passed on the command line using the `-u`, `-p`, `-r`, and `-l` options (which is insecure, because any user can see this data), or it can be placed in a file for the console to read. For security reasons, this file should be readable only by the user running the console, although the console does not enforce this restriction.

The format of *file* is:

`hostname=`*host name*
`username=`*user name*
`password=`*password for user name*
`rolename=`*role name*
`rolepassword=`*password for role name*

Only one set of
`hostname-username-password-rolename-rolepassword`

may be specified in any one file. If the rolename is not specified, no role will be assumed.

`-B | - —toolbox` *13;toolbox*

Loads the specified toolbox. *toolbox* can be either a fully-qualified URL or a filename. If you specify an HTTP URL as, for example,

`http://`*host_name*`:`*port*`/. . .`

it must point to a *host_name* and *port* on which an Solaris Management Console server is running. If you omit *port*, the default port, `898`, is used. This option overrides the `-H` option.

`-D | - —domain` *13;domain*

Specifies the default domain that you want to manage. The syntax of *domain* is *type*:/*host_name*/*domain_name*, where *type* is `nis`, `nisplus`, `dns`, `ldap`, or `file`; *host_name* is the name of the machine that serves the domain; and *domain_name* is the name of the domain you want to manage. (*Note:* Do *not* use `nis+` for `nisplus`.) This option applies only to a single tool run in the terminal console.

If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.

`-h | - —help`

Prints a usage statement about the `smc` command and its subcommands to the terminal window. To print a usage statement for one of the subcommands, enter `-h` after the subcommand.

`-H | - —hostname` *13;host_name:port*

Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name*:*port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `-B` option (see above), or set your console preferences to load a “home toolbox” by default.

`-J`*java_option*	Specifies an option that can be passed directly to the Java runtime (see `java`(1). Do not enter a space between `-J` and the argument. This option is most useful for developers.
`-l|- –rolepassword` *13;role_password*	Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-p|- –password` *13;password*	Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-r|- –rolename` *13;role_name*	Specifies a role name for authentication. If you are running the Solaris Management Console in a terminal and you do not specify this option, no role is assumed. The GUI console may prompt you for a role name, although you may not need to assume a role.
`-s|- –silent`	Disables informational messages printed to the terminal.
`-t`	Runs the Solaris Management Console in terminal mode. If this option is not given, the Solaris Management Console will automatically run in terminal mode if it cannot find a graphical display.
`- –trust`	Trusts all downloaded code implicitly. Use this option when running the terminal console non-interactively and you cannot let the console wait for user input.
`-T|- –tool` *13;tool_name*	Runs the tool with the Java class name that corresponds to *tool_name*. If you do not specify this option and the Solaris Management Console is running in terminal mode, the system prompts you. If the Solaris Management Console is running in graphical mode, the system either loads a toolbox or prompts you for one (see options `-H` and `-B`).

`-u	- –username` *13;user_name*	Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.
`-v	- –version`	Prints the version of the Solaris Management Console to the terminal. In the graphical console, this information can be found in the `About` box, available from the `Help` menu.
`-y	- –yes`	Answers `yes` to all yes/no questions. Use this option when running the terminal console non-interactively and you cannot let the console wait for user input.

Examples

EXAMPLE 1 Printing a Usage Statement

The following prints a usage statement about the `smc` command to the terminal window:

```
smc --help
```

EXAMPLE 2 Using SMF Property to Allow Remote Access

The following sequence of commands allows remote access to WBEM-based applications, including the Solaris Management Console.

```
# svccfg -s svc:/application/management/wbem \
    setprop options/tcp_listen = true
# svcadm refresh svc:/application/management/wbem
```

EXAMPLE 3 Passing an Option to Java

The following passes an option through to the Java VM, which sets the `com.example.boolean` system property to `true`. This system property is only an example; the Solaris Management Console does not use it.

```
smc -J-Dcom.example.boolean=true
```

Environment Variables

See `environ(5)` for a description of the following environment variable that affects the execution of the `smc` command:

`JAVA_HOME` If you do not specify this environment variable, your `PATH` is searched for a suitable java. Otherwise, the `/usr/j2se` location is used.

Exit Status

The following exit values are returned. Other error codes may be returned if you specify a tool (using `-T` *tool_name*) that has its own error codes. See the documentation for the appropriate tool.

`0` Successful completion.

`1` An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmcc

See Also auths(1), java(1), profiles(1), roles(1), smcconf(1M), svcadm(1M), svccfg(1M), attributes(5), environ(5), smf(5), X(7)

Name smccompile – build class list and compile Solaris Management Console service beans for remote use

Synopsis

```
/usr/sadm/bin/smccompile -c beanname

/usr/sadm/bin/smccompile -j tool | service [-n altjarname] jarfile

/usr/sadm/bin/smccompile -j library [-n altjarname] ALLTOOL | ALLSERVICE | ALL | attachedBeanna
```

Description The `smccompile` command is used by developers of tools, services, and libraries for the Solaris Management Console. For information regarding the Solaris Management Console, see `smc(1M)`.

`smccompile` compiles service class files given by the bean name for use with the Solaris Management Console. This step builds the extra proxy and stub classes for services to be used with Solaris Management Console tools. Solaris Management Console requires running `smccompile -c` before creating service jar files, and `smccompile -j` after creating tool, service, and library jars.

`smccompile`, in conjunction with `smcregister(1M)`, is intended to replace the `smcconf` command as the preferred interface for managing the Solaris Management Console repository as well as toolboxes from within scripts, due to significant performance enhancements over `smcconf`.

Options The following options are supported:

ALL	Specify that the library being registered to or unregistered from the repository is for use by all tools and services.
ALLSERVICE	Specify that the library being registered to or unregistered from the repository is for use by all services.
ALLTOOL	Specify that the library being registered to or unregistered from the repository is for use by all tools.
attachedBeaname	Specify the name of a registered jar to which the library jarfile should be attached to (or detached from). This is typically the same as *altjarname* (if provided) or *jarfile* used to register the jar to which this library is being attached or detached. An attached library means the library is only available for use by the tool or service to which it is being attached.
beanname	The full package path of the bean name to be compiled. An example bean name is: `com.mycompany.myproduct.MyService`.
`-c`	Compile and build service class files for the specified bean name. This step builds the extra proxy and stub classes for services to be used with Solaris Management Console tools. You must run `smccompile` with this option before creating service type jar files.

`-j`	Build a list of classes in text format, suitable as input to `smcregister` for registration with the Solaris Management Console repository. The output is written to standard out and should be redirected to a file. You must run `smccompile` with this option after creating any tool, service, or library jar.
jarfile	Specify the full path to the jar file to be registered. The name must be in the form *beanname*`.jar`, where *beanname* is the package path to the bean. If it is not, an alternate name must be given in that form using the `-n` option.
`-n` *altjarname*	Rename the jarfile in the repository to *altjarname*. Typically. this is the full bean name. For example, if the jarfile was `MyTool.jar`, then *altjarname* might be `com.mycompany.myproduct.MyTool.jar`. It is recommended that an *altjarname* containing the full package path be used. You must use this same name when registering the jar with `smcregister`.

Examples

EXAMPLE 1 Compiling a Service

The following command takes a Solaris Management Console service and builds its proxy and stub classes to make the service usable by Solaris Management Console tools:

```
/usr/sadm/bin/smccompile -c com.mycompany.myproject.MyServiceImpl
```

EXAMPLE 2 Building a Class List for a Service

The following command builds the class list file (`classlist.txt`) for a service suitable for use with the `smcregister(1M)` command:

```
/usr/sadm/bin/smccompile -j service \
     -n com.mycompany.myproject.MyServiceImpl.jar \
     ${HOME}/workarea/MyServiceImpl.jar > classlist.txt
```

The following command does the same thing without specifying an alternate name:

```
/usr/sadm/bin/smccompile -j service \
     ${HOME}/workarea/com.mycompany.myproject.MyServiceImpl.jar > classlist.txt
```

EXAMPLE 3 Building a Class List for a Tool

The following command builds the class list file (`classlist.txt`) for a tool suitable for use with the `smcregister`(1M) command:

```
/usr/sadm/bin/smccompile -j tool \
     -n com.mycompany.myproject.MyTool.jar \
     ${HOME}/workarea/MyTool.jar > classlist.txt
```

The following command does the same thing without specifying an alternate name:

EXAMPLE 3 Building a Class List for a Tool *(Continued)*

```
/usr/sadm/bin/smccompile -j tool \
     ${HOME}/workarea/com.mycompany.myproject.MyTool.jar > classlist.txt
```

EXAMPLE 4 Building a Class List for a Library Attached to All Tools

The following command builds the class list file (`classlist.txt`) for a library suitable for use with the `smcregister`(1M) command, and is attached to all tools:

```
/usr/sadm/bin/smccompile -j library \
     -n com.mycompany.myproject.MyLibrary.jar \
     ALLTOOL ${HOME}/workarea/MyLibrary.jar > classlist.txt
```

The following command does the same thing without specifying an alternate name:

```
/usr/sadm/bin/smccompile -j library \
    ALLTOOL \
    ${HOME}/workarea/com.mycompany.myproject.MyLibrary.jar > classlist.txt
```

EXAMPLE 5 Building a Class List for a Library Attached to a Specific Tool

The following command builds the class list file (`classlist.txt`) for a library suitable for use with the `smcregister`(1M) command, and is attached to a specific tool:

```
/usr/sadm/bin/smccompile -j library \
     -n com.mycompany.myproject.MyLibrary.jar \
     com.mycompany.myproject.MyTool.jar \
     ${HOME}/workarea/MyLibrary.jar > classlist.txt
```

The following command does the same thing without specifying an alternate name:

```
/usr/sadm/bin/smccompile -j library \
     com.mycompany.myproject.MyTool.jar \
     ${HOME}/workarea/com.mycompany.myproject.MyLibrary.jar > classlist.txt
```

Environment Variables

See `environ`(5) for descriptions of the following environment variables that affect the execution of `smccompile`:

JAVA_HOME — If you do not specify this environment variable, your PATH is searched for a suitable java. Otherwise, the `/usr/j2se` location is used.

Exit Status

The following exit values are returned:

`0` — Successful completion.

`1` — An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWMc

See Also smc(1M), smcconf(1M), smcregister(1M), attributes(5), environ(5)

Notes All standard shell quoting rules apply.

Name smcconf – configure the Solaris Management Console

Synopsis

```
/usr/sadm/bin/smcconf [-h] [-v] toolbox [action] [target]
     [parameters] [options]

/usr/sadm/bin/smcconf [-h] [-v] repository [action]
     [target] [parameters] [options]
```

Description The `smcconf` command configures the Solaris Management Console. See `smc(1M)`. This command enables you to add to, remove from, and list the contents of the toolboxes and bean repository.

Using `smcconf` to edit toolboxes is not as feature-rich as using the graphical editor in Solaris Management Console. The command line interface is intended for use in packaging scripts that do not require user interaction. To edit all the properties of a toolbox or to modify the hierarchy of folders in a toolbox, you must use the specialized graphical editor, that is, `smc edit`. See `smc(1M)`.

`smcregister` is intended to replace the `smcconf` command as the preferred interface for managing the Solaris Management Console repository as well as toolboxes from within scripts, due to significant performance enhancements over `smcconf`. See `smcregister(1M)`, `smccompile(1M)`, and the `Solaris Management Console SDK Guide` at `/usr/sadm/lib/smc/docs/sdkguide/index.html` for details.

Options The following options are supported:

`-h` Prints out a usage summary for the command.

`-v` Verbose option. Displays the debugging output at any time.

toolbox configuration

action Legal values are:

`add` Adds a *target* to the toolbox. Specify the path to the toolbox using the `-B` *toolboxpath* option and, optionally, provide locale information with the `-L` *locale* option.

`remove` Removes a *target* from the toolbox. Specify the path to the toolbox using the `-B` *toolboxpath* option and, as an alternative, provide locale information with the `-L` *locale* option.

`create` Creates a new toolbox with no tools in it. The only *target* recognized is `toolbox`.

`list` Lists the contents of the toolbox. No *target* is recognized. If you specify a *parameter*, it is taken as the path to a toolbox and the contents of that toolbox are listed. If you do not specify a *parameter*, the contents of the default toolbox are listed.

target Legal values are:

`tool` If the *action* is specified as `add`, this target adds a native Solaris Management Console tool from the toolbox. The required *parameter* is the full Java classname of the tool you are adding. If you specify a folder name with the `-F` option, the tool is placed inside that folder (the folder will not be created if it does not already exist). Otherwise, the tool is appended to the end of the toolbox and not placed inside any folder.

If the *action* is specified as `remove`, this target removes a native Solaris Management Console tool from the toolbox. The required *parameter* is the full Java classname of the tool you want to remove. If you specify a folder name with the `-F` option, any tool with the given name in that folder will be removed. If no folder name is specified, all tools with the given name in the toolbox are removed.

For the tool to appear in the console, the tool must also be registered in the repository. See the `repository configuration` section below for more information. If a tool is referenced in a toolbox but is not registered, it will not appear in the console when the toolbox is loaded.

Removing a tool from a toolbox does not remove the tool from the server repository.

`tbxURL` If the *action* is specified as `add` or `remove`, this target adds to or removes from the toolbox a link to another toolbox. The required *parameter* is the URL to the other toolbox.

The properties of addition and removal are the same as for the `tool` *target*.

`toolbox` If the *action* is specified as `create`, this target creates a skeleton toolbox with no tools. The required *parameter*s are: the toolbox name, description, and small and large icon paths. These must be followed by the `-B` *toolboxpath* and `-D` *scope* options.

`legacy` If the *action* is specified as `add` or `remove`, this target adds or removes legacy applications (command-line, X-windows, and web-based) to or from the toolbox. The `-N`, `-T`, `-E`, and `-B` options are required. The `-A` option is optional. Placement in the toolbox with the `-F` option follows the same rules as for the `tool` and `tbxURL` targets. See `NOTES` for more information about legacy applications.

`folder` If the *action* is specified as `add`, this target adds a folder to the toolbox. The required *parameter*s are: the folder name, description, and small and large icon paths.

If the *action* is specified as `remove`, this target removes a folder from the toolbox. If the folder to be removed is itself inside a folder, the containing folder must be specified with the `-F` option.

parameters Specifies values that might be required, depending on the combination of *action* and *target*.

options Supported options for various *action* and *target* combinations for the `toolbox` configuration are:

`-A` *parameters* Specifies the parameters to pass to the legacy application. This option is available only for the `legacy` target.

`-B` *toolboxpath* Specifies the path of the toolbox that is being modified. If this option is not given, the modifications will be performed on the default toolbox, "`This Computer`".

`-D` *scope* Specifies the scope (domain) in which the tool should be run. The legal values for *scope* are `file`, `nis`, `nisplus`, `dns`, and `ldap`. This can also be specified for a folder or a toolbox.

In the former case, all tools in that folder and its subfolders are run in that scope; in the latter, all tools in the toolbox are run in that scope.

`-E` *appPath* Specifies the absolute executable path of the legacy application. This option is available only for the `legacy` target.

`-F` *folder* Specifies the full path of the container folder. If this option is not given, the default folder is the 'root' folder of the toolbox.

`-H` [*host_name*][`:`*port*] Specifies the host and port from which a tool should be loaded. If *host_name* is not given, the default host is used. The default host is `localhost`, if the toolbox is loaded from the local file system, or the host from which the toolbox is loaded if loaded from a remote Solaris Management Console server. If `:port` is not given, the default port will be used. If

this option is not given at all, both the default host and the default port are used.

`-L` *locale* — Specifies the locale of the toolbox that is being modified. The default is the `C locale`.

`-N` *appName* — Specifies the name of the legacy application being registered. This is the name that appears in the console. This option is available only for the `legacy` target.

`-P` *key*:*value* — Specifies the key/value pairs that define parameters to a tool. Multiple key/value pairs can be specified at a time.

`-T` *appType* — Specifies the legacy application type. Legal values are `CLI`, `XAPP`, or `HTML`. This option is available only for the `legacy` target.

repository configuration

The Solaris Management Console repository stores information about the registered tools and services, as well as libraries (for instance, resource jars) and properties attached to tools or services.

action Legal values are:

`add` Adds information to the repository. If the `-f` option is given to `add`, the information overwrites any information of the same name already in the repository. If the `-f` option is not given, an error might be returned if the information is already in the repository.

`remove` Removes information from the repository.

`list` Lists the contents of the repository:

- All registered tools
- All registered services
- All libraries attached to all tools
- All libraries attached to all services
- All libraries attached to all tools and services

target Legal values are:

`bean` If the *action* is specified as `add`, this target will add a tool or service bean (which kind is determined by the contents of the bean) to the repository. The required *parameter* is the path to the jar file that contains the bean to be added.

If the *action* is specified as `remove`, this target will remove a tool or service bean from the repository. The required *parameter* is the full Java classname of the desired bean.

`library` If the *action* is specified as `add`, this target adds a "library" jar file to a tool or service bean. The two required *parameter*s are the full Java classname of the desired bean and the path to the jar file to be attached. The bean name can also be one of the "pseudo-beans," `ALL`, `ALLTOOL`, or `ALLSERVICE`, in which case the library is attached, respectively, to all beans, all tools, or all services in the repository.

If the *action* is specified as `remove`, this target detaches a "library" jar file from a tool or service bean. The two required *parameter*s are the full Java classname of the desired bean and the name of the jar file that is attached. As with the `add` action, the three "pseudo-beans" `ALL`, `ALLTOOL`, or `ALLSERVICE` can be used.

`property` If the *action* is specified as `add`, this target defines a property on a tool or service. One or more key/value pairs must be specified in the form,

`-P` *key*=*value*

Following this property list is a "pseudo-bean name," *pseudoBeanName*, as defined for the `library` target, on which the properties are defined. Optionally, a library name can follow the "pseudo-bean" name, in which case the properties are defined on the library that is attached to the named bean.

If the *action* is specified as `remove`, this target undefines a property on a tool or service. The key/value pairs, "pseudo-bean" name, and optional library are specified for the `add` action.

Examples

EXAMPLE 1 Adding Legacy Applications to a Toolbox

The following command adds to the default toolbox the command line interface (CLI) application, `/usr/bin/ls`, with arguments `-al -R`, giving it the name, `Directory Listing`:

```
/usr/sadm/bin/smcconf toolbox add legacy -N "Directory Listing" \
     -T CLI -E /usr/bin/ls -A "-al -R"
```

EXAMPLE 2 Adding a Folder to a Toolbox

The following command adds to the standard `Management Tools` toolbox a folder with the name, `New Folder`, the description, `This is a new folder`, and the small and large icons, `folder_s.gif` and `folder_l.gif`:

```
/usr/sadm/bin/smcconf toolbox add folder "New Folder" \
     "This is a new folder" folder_s.gif folder _l.gif \
     -B /var/sadm/smc/toolboxes/smc/smc.tbx
```

EXAMPLE 3 Adding a Native Solaris Management Console Tool to a Toolbox

The following command adds a native Solaris Management Console tool to the default toolbox. The Java classname of the tool is `HelloWorld.client.HelloTool` (the name, description, and icons visible in the console are provided by the tool itself). When loaded, it is run in the NIS domain, `syrinx`, which is hosted by the machine, `temple`, and is retrieved from port `2112` on the machine from which the toolbox was loaded:

```
/usr/sadm/bin/smcconf toolbox add tool HelloWorld.client.HelloTool \
     -D nis:/temple/syrinx -H :2112
```

EXAMPLE 4 Adding an Solaris Management Console Tool to the Repository

The following command adds the Java bean found in `HelloWorld.jar` to the repository. The jar file contains information that the bean is a tool:

```
/usr/sadm/bin/smcconf repository add bean HelloWorld.jar
```

EXAMPLE 5 Removing an Solaris Management Console Service from the repository

The following command removes a Java bean from the repository. Although the name of the bean implies that it is a service, that is merely a convention; the repository knows whether a particular registered bean is a tool or a service:

```
/usr/sadm/bin/smcconf repository remove bean \
     HelloWorld.server.HelloService
```

EXAMPLE 6 Attaching a Library to a Tool

The following command adds the library jar file, `HelloWorld_fr.jar` (probably a French localized version of the `HelloTool`'s resources) to the bean, `HelloWorld.client.HelloTool`:

```
/usr/sadm/bin/smcconf repository add library \
     HelloWorld.client.HelloTool HelloWorld_fr.jar
```

EXAMPLE 7 Attaching a Library to all Tools

The following command adds the library jar file, `widgets.jar`, to all tools in the repository. The library probably contains a widget set that might be useful to any registered tools:

```
/usr/sadm/bin/smcconf repository add library ALLTOOL widgets.jar
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of the `smcconf` command:

`JAVA_HOME` If you do not specify this environment variable, your PATH is searched for a suitable `java`. Otherwise, the `/usr/j2se` location is used.

`DISPLAY` If you do not set this environment variable, set it to null, or set it to an X(7) display to which you are not authorized to connect, the Solaris Management Console starts in terminal mode instead of graphical mode.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmc

See Also `smc(1M)`, `smccompile(1M)`, `smcregister(1M)`, `attributes(5)`, `environ(5)`

Notes All standard shell quoting rules apply.

Legacy applications (X-windows, command-line, and web-based applications) are handled differently from "native" Solaris Management Console tools. Legacy tools are handled by an instantiation of a native Solaris Management Console tool, `LegacyAppLauncher`, which, through the toolbox, is given the necessary information to run the legacy application: path, options, and so forth. Thus, you do not register a legacy application into the repository as you would a native Solaris Management Console tool. Instead, legacy applications appear only in toolboxes.

Name smcregister – configure the Solaris Management Console

Synopsis

```
/usr/sadm/bin/smcregister [-h] tool [-n altjarname] jarfile classlistfile xmlfile

/usr/sadm/bin/smcregister [-h] service [-n altjarname] jarfile classlistfile xmlfile
     [native_lib_list]

/usr/sadm/bin/smcregister [-h] library [-n altjarname] jarfile classlistfile | none ALLTOOL | ALLSERVICE |

/usr/sadm/bin/smcregister [-h] tool | service -u jarfile

/usr/sadm/bin/smcregister [-h] library -u jarfile ALL | ALLTOOL | ALLSERVICE | Attachedbeanname

/usr/sadm/bin/smcregister [-h] toolbox [-D] [action] [-f]
     [target] [parameters] [options]

/usr/sadm/bin/smcregister [-h] property key value ALL | ALLTOOL | ALLSERVICE | Attachedbeanname

/usr/sadm/bin/smcregister [-h] property -u key ALL | ALLTOOL | ALLSERVICE | Attachedbeanname

/usr/sadm/bin/smcregister [-h] repository list

/usr/sadm/bin/smcregister [-h] scripts regscript unregscript
```

Description The `smcregister` command configures the Solaris Management Console. For information regarding the Solaris Management Console, see smc(1M). This command enables you to add to, remove from, and list the contents of toolboxes and the Solaris Management Console repository.

`smcregister` also allows you to register scripts to perform registrations and unregistrations. Typically, a package containing one or more tools or services posts tool and service registrations immediately after installation. On Solaris, this is by way of invocations of `smcregister` from within a package post-install script. Similarly, unregistrations would be posted from within a package pre-remove script. These are per-machine registrations - that is, registration requests must be posted on each machine on which the Solaris Management Console server will be running. However, due to the way that diskless clients are installed, registration requests cannot be made at install time. Therefore, packages should include and install registration and unregistration scripts, and then register these scripts during installation by way of the `scripts` subcommand. These scripts should contain tool, toolbox, service, library or property configurations in any of its forms as listed in this man page. While these scripts function very much like package post-install and pre-remove scripts, do not assume the normal package environment is available. However, `PATH` can assumed to be `/usr/sbin:/usr/bin`

Using `smcregister` to edit toolboxes is not as feature-rich as using the Solaris Management Console's graphical editor. The command line interface is intended for use in packaging scripts that do not require user interaction. To edit all the properties of a toolbox or to modify the hierarchy of folders in a toolbox, you must use the specialized graphical editor, that is, `smc edit`. See `smc`(1M).

`smcregister` is intended to replace the `smcconf` command as the preferred interface for managing the Solaris Management Console repository as well as toolboxes from within scripts, due to significant performance enhancements over `smcconf`.

Options The following options are supported:

`-h` Prints out a usage summary for the command.

Scripts Configuration

regscript — The full path of a script containing registration commands. The script is executed upon the next restart of the Solaris Management Console server after the package containing the script is installed.

unregscript — The full path of a script containing unregistration commands. The script is executed upon the next restart of the Solaris Management Console server after the package containing the script is removed.

Toolbox Configuration

action — Legal values are:

- add — Adds a target to the toolbox. Specify the path to the toolbox using the `-B toolboxpath` option and, optionally, provide locale information with the `-L locale` option.
- create — Creates a new toolbox with no tools in it. The only target recognized is `toolbox`.
- list — Lists the contents of the toolbox. No target is recognized. If you specify a parameter, it is taken as the path to a toolbox and the contents of that toolbox are listed. If you do not specify a parameter, the contents of the default toolbox are listed.
- remove — Removes a target from the toolbox. Specify the path to the toolbox using the `-B toolboxpath` option and, optionally, provide locale information with the `-L locale` option.

`-D` — Defers execution of the `toolbox` command until the Solaris Management Console server is restarted. This is a convenient option for use in packaging scripts during install and un-install. Additionally, the command runs much faster than if run interactively (without `-D`).

target — Legal values are:

- folder — If the action is specified as `add`, this target adds a folder to the toolbox. There are four required parameters: the folder name, description, and small and large icon paths. If the action is specified as remove, this target removes a folder from the toolbox. If the folder to be removed is itself inside a folder, the containing folder must be specified with the `-F` option.
- legacy — If the action is specified as `add` or `remove`, this target adds or removes legacy applications (command line, `X-windows`, and

web-based) to or from the toolbox. The `-N`, `-T`, `-E`, and `-B` options are required, and the `-A` option is optional. Placement in the toolbox with the `-F` option follows the same rules as for the `tool` and `tbxURL` targets.See `NOTES` for more information about legacy applications.

tbxURL — If the action is specified as `add` or `remove`, this target adds to or removes from the toolbox a link to another toolbox. The required parameter is the URL to the other toolbox. The properties of addition and removal are the same as for the tool target.

tool — If the action is specified as `add`, this target adds a native Solaris Management Console tool from the toolbox. The required parameter is the full Java classname of the tool you are adding. If you specify a folder name with the `-F` option, the tool is placed inside that folder (the folder will not be created if it does not already exist). Otherwise, the tool is appended to the end of the toolbox and not placed inside any folder. If the action is specified as remove, this target removes a native Solaris Management Console tool from the toolbox. The required parameter is the full Java classname of the tool you wish to remove. If you specify a folder name with the `-F` option, any tool with the given name in that folder will be removed. If no folder name is specified, all tools with the given name in the toolbox will be removed. For the tool to show up in the console, the tool must also be registered in the repository. See the repository configuration section below for more information. If a tool is referenced in a toolbox but is not registered, it will not appear in the console when the toolbox is loaded. Removing a tool from a toolbox does not remove the tool from the server repository.

toolbox — If the action is specified as `create`, this target creates a skeleton toolbox with no tools. There are four required parameters: the toolbox name, description, and small and large icon paths. These must be followed by the `-B` *toolboxpath* and `-D` *scope* options.

parameters — Specifies values that may be required depending on the combination of action and target.

options — Supported options for various action and target combinations for the toolbox configuration are:

`-A` — Specifies the parameters to pass to the legacy application. This option is available only for the legacy target.

`-B`	Specifies the path of the toolbox that is being modified. If this option is not given, the modifications will be performed on the default toolbox, `This Computer`.
`-D`	Specifies the scope (domain) in which the tool should be run. The legal values for scope are `file`, `nis`, `nisplus`, `dns`, and `ldap`. This may also be specified for a folder or a toolbox. In the former case, all tools in that folder and its subfolders will be run in that scope; in the latter, all tools in the toolbox will be run in that scope.
`-E`	Specifies the absolute executable path of the legacy application. This option is available only for the legacy target.
`-f`	If the `-f` option is given to add, the information will overwrite any information of the same name already in the toolbox. If the `-f` option is not given, an error may be returned if the information is already in the toolbox.
`-F` *folder*	Specifies the full path of the container folder. If this option is not given, the default folder is the *root* folder of the toolbox.
`-H` [*host_name*][:*port*]	Specifies the host and port from which a tool should be loaded. If host_name is not given, the default host (localhost, if the toolbox is loaded from the local filesystem, or the host from which the toolbox is loaded if loaded from a remote Solaris Management Console server) will be used. If :*port* is not given, the default port will be used. If this option is not given at all, both the default host and the default port will be used.
`-L` *locale*	Specifies the locale of the toolbox which is being modified. The default is the `C` locale.
`-N` *appName*	Specifies the name of the legacy application being registered. This is the name that will appear in the console. This option is available only for the `legacy` target.

-P *key:value*	Specifies the key/value pairs that define parameters to a tool. Multiple key/value pairs can be specified at a time.
-T *appType*	Specifies the legacy application type. Legal values are `CLI`, `XAPP`, or `HTML`. This option is available only for the legacy target.

Tool, Service, and Library Configuration

See `NOTES` for more information about registration and unregistration of tools, services, and libraries.

ALL	Specify that the library being registered to or unregistered from the repository is for use by all tools and services.
ALLSERVICE	Specify that the library being registered to or unregistered from the repository is for use by all services.
ALLTOOL	Specify that the library being registered to or unregistered from the repository is for use by all tools.
attachedBeanname	The name of a registered jar to which the library jarfile should be attached to (or detached from). This is typically the same as *altjarname* (if provided) or *jarfile* used to register the jar to which this library is being attached or detached. An attached library means the library is only available for use by the tool or service to which it is being attached.
classlistfile	The classlist text file generated from the smccompile(1M) command. Library registration does not require that a classlist file be specified. Instead, you can substitute the keyword `none` in place of the classlist path argument to `smcregister`, in which case one will be generated automatically. Generating the classlist automatically during server startup will cause the next server restart to take longer, so it is strongly suggested that developers always provide a classlist file with their libraries. Auto-generation is more appropriately used to register 3rd-party library jars.
jarfile	The full path to the jar file to be registered/unregistered. The name must be in the form *beanname*`.jar`, where *beanname* is the package path to the bean. If it is not, an alternate name must be given in that form using the -n option.
-n *altjarname*	Rename the jarfile in the repository to *altjarname*. This would typically be the full bean name. For example, if the jarfile was `MyTool.jar`, then *altjarname* might be com.mycompany.myproduct.MyTool.jar. It is recommended that an *altjarname* containing the full package path be used.

native_lib_list List of up to 4 native libraries that can be associated with a service bean.

`-u` The operation will be to un-register the jar with the Solaris Management Console repository. The jarfile argument must be identical to the *altjarname* used to register the jar (if provided), or jarfile.

xmlfile The `xml` descriptor file that describes this jarfile. Every tool or services must have one. See the `Solaris Management Console SDK Guide` located at `/usr/sadm/lib/smc/docs/sdkguide/index.html`.

Repository Configuration The Solaris Management Console repository stores information about the registered tools and services, as well as libraries (for instance, resource jars) and properties attached to tools or services.

list Lists the contents of the repository:

- All registered tools
- All registered services
- All libraries attached to all tools
- All libraries attached to all services
- All libraries attached to all tools and services

Property Configuration See `NOTES` for more information about registration and unregistration of properties. If registering a property, this defines a property on a tool or service. Only one key value pair at a time can be registered.

beanname The name of a registered jar on which the properties will be defined. Optionally, a library name may follow the bean name, in which case the properties are defined on the library that is attached to the named bean.

If unregistering a property, this undefines a property from a tool or service. Only one key value pair at a time can be registered. The key, beanname, and optional library are specified as for registering a property.

Examples

EXAMPLE 1 Adding Legacy Applications to a Toolbox

The following command adds to the default toolbox the Command Line Interface (CLI) application, `/usr/bin/ls` with arguments `-al -R`, giving it the name, Directory Listing:

```
/usr/sadm/bin/smcregister toolbox add legacy -N "Directory Listing" \
     -T CLI -E /usr/bin/ls -A "-al -R"
```

Use this variation to defer execution of this command until the Solaris Management Console server is restarted:

```
/usr/sadm/bin/smcregister toolbox -D add legacy -N "Directory Listing" \
     -T CLI -E /usr/bin/ls -A "-al -R"
```

EXAMPLE 2 Adding a Folder to a Toolbox

The following command adds to the standard `Management Tools` toolbox a folder with the name, New Folder, the description, This is a new folder, and the small and large icons, `folder_s.gif` and `folder_l.gif`:

```
/usr/sadm/bin/smcregister toolbox add folder "New Folder" \
     "This is a new folder" folder_s.gif folder _l.gif \
     -B /var/sadm/smc/toolboxes/smc/smc.tbx
```

EXAMPLE 3 Adding a Native Solaris Management Console Tool to a Toolbox

The following command adds a native Solaris Management Console tool to the default toolbox. The Java classname of the tool is `com.mycompany.myproject.client.MyTool` (the name, description, and icons visible in the console are provided by the tool itself). When loaded, it will be run in the NIS domain, syrinx, which is hosted by the machine, temple, and will be retrieved from port 2112 on the machine from which the toolbox was loaded.

```
/usr/sadm/bin/smcregister toolbox add tool \
     com.mycompany.myproject.client.MyTool \
     -D nis:/temple/syrinx -H :2112
```

EXAMPLE 4 Adding an Solaris Management Console Tool to the Repository

The following command adds the Java bean found in MyTool.jar to the repository. The xml file contains information about the tool. The classlist file would have been generated by `smccompile -j`:

```
/usr/sadm/bin/smcregister tool -n com.mycompany.myproject.client.MyTool.jar \
     ${HOME}/workarea/MyTool.jar \
     ${HOME}/workarea/MyTool_classlist.txt \
     ${HOME}/workarea/MyTool.xml
```

Use this variation to add an Solaris Management Console tool to the repository without specifying an alternate name:

```
/usr/sadm/bin/smcregister tool  \
     ${HOME}/workarea/com.mycompany.myproject.client.MyTool.jar \
     ${HOME}/workarea/MyTool_classlist.txt \
     ${HOME}/workarea/MyTool.xml
```

EXAMPLE 5 Adding an Solaris Management Console Service to the Repository

The following command adds the Java bean found in MyServiceImpl.jar to the repository. The xml file contains information about the service. The classlist file would have been generated by `smccompile -j`. The extra proxy and stub classes included in the jar would have been generated by `smccompile -c`:

```
/usr/sadm/bin/smcregister service \
     -n com.mycompany.myproject.server.MyServiceImpl.jar \
```

EXAMPLE 5 Adding an Solaris Management Console Service to the Repository *(Continued)*

```
     ${HOME}/workarea/MyServiceImpl.jar \
     ${HOME}/workarea/MyServiceImpl_classlist.txt \
     ${HOME}/workarea/MyServiceImpl.xml
```

Use this variation to add a Solaris Management Console service to the repository without specifying an alternate name:

```
/usr/sadm/bin/smcregister service  \
     ${HOME}/workarea/com.mycompany.myproject.server.MyServiceImpl.jar \
     ${HOME}/workarea/MyServiceImpl_classlist.txt \
     ${HOME}/workarea/MyServiceImpl.xml
```

EXAMPLE 6 Removing an Solaris Management Console Tool From the Repository

The following command removes a Java tool bean from the repository:

```
/usr/sadm/bin/smcregister tool \
     -u com.mycompany.myproject.client.MyTool.jar
```

EXAMPLE 7 Removing an Solaris Management Console Service From the Repository

The following command removes a Java service bean from the repository:

```
/usr/sadm/bin/smcregister service \
     -u com.mycompany.myproject.server.MyServiceImpl.jar
```

EXAMPLE 8 Attaching a Library to a Specific Tool

The following command adds the library jar file, MyTool_fr.jar (probably a French localized version of the MyTool's resources) to the bean, com.mycompany.myproject.client.MyTool:

```
/usr/sadm/bin/smcregister library  \
     -n MyTool_fr.jar \
     ${HOME}/workarea/MyTool_fr.jar \
     ${HOME}/workarea/MyTool_fr_classlist.txt \
     com.mycompany.myproject.client.MyTool
```

EXAMPLE 9 Attaching a Library to All Tools

The following command adds the library jar file, widgets.jar, to all tools in the repository. The library probably contains a widget set which might be useful to any registered tools. The classlist file would have been generated by `smccompile -j`.

```
/usr/sadm/bin/smcregister library \
     ${HOME}/workarea/lib/widgets.jar\
     ${HOME}/workarea/lib/widgets_classlist.txt \
     ALLTOOL
```

Alternatively, to add a 3rd-party library jar to all tools, replace the classlist file with `none`:

EXAMPLE 9 Attaching a Library to All Tools *(Continued)*

```
/usr/sadm/bin/smcregister library \
     /opt/lib/XYZwidgets.jar none ALLTOOL
```

EXAMPLE 10 Detaching a Library from All Tools

The following command removes the Java library bean from the repository:

```
/usr/sadm/bin/smcregister library -u MyTool_fr.jar ALLTOOL
```

EXAMPLE 11 Detaching a Library from a Specific Tool

The following command detaches the library jar file, MyTool_fr.jar (probably a French localized version of the MyTool's resources) from the bean `com.mycompany.myproject.client.MyTool`, and removes it from the repository:

```
/usr/sadm/bin/smcregister library -u MyTool_fr.jar \
     com.mycompany.myproject.client.MyTool
```

EXAMPLE 12 Registering Scripts

The following command registers the following scripts containing registration and unregistration commands. MyProduct_reg.sh will be executed upon the next server restart after the file is installed by the owning package. MyProduct_unreg.sh will be executed upon the next server restart after the file is removed by the owning package:

```
/usr/sadm/bin/smcregister scripts \
     /usr/sadm/lib/myProduct/MyProduct_reg.sh \
     /usr/sadm/lib/myProduct/MyProduct_unreg.sh
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `smcregister`:

JAVA_HOME — If you do not specify this environment variable, your `PATH` is searched for a suitable java. Otherwise, the `/usr/j2se` location is used.

Exit Status

The following exit values are returned:

`0` — Successful completion.

`1` — An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmc

See Also smc(1M), smcconf(1M), smccompile(1M), attributes(5), environ(5)

Notes All standard shell quoting rules apply.

Legacy applications (`X-windows`, command-line, and web-based applications) are handled differently from native Solaris Management Console tools. Legacy tools are handled by an instantiation of a native Solaris Management Console tool, `LegacyAppLauncher`, which, through the toolbox, is given the necessary information to run the legacy application: path, options, and so forth. Thus, you do not register a legacy application into the repository as you would a native Solaris Management Console tool. Instead, legacy applications appear only in toolboxes.

Registration and unregistration of tools, services, libraries, and properties do not take effect until the Solaris Management Console server is restarted. Run `/etc/init.d/init.wbem stop` followed by `/etc/init.d/init.wbem start`

Name smcron – manage jobs in the crontab database

Synopsis

```
/usr/sadm/bin/smcron subcommand [ auth_args] - —
     [subcommand_args]
```

Description The `smcron` command manages jobs in the `crontab(1)` database.

subcommands `smcron` *subcommand*s are:

`add` Adds a job to the `crontab`(1) database. To add a job, the administrator must have the `solaris.jobs.user` authorization. To add a job to another user's `crontab` file, the administrator must have the `solaris.jobs.admin` authorization.

`delete` Deletes a job from the `crontab`(1) database. To delete a job, the administrator must have the `solaris.jobs.user` authorization. To delete a job from another user's `crontab` file, the administrator must have the `solaris.jobs.admin` authorization.

`list` Lists one or more jobs in the `crontab`(1) database. To list all jobs, the administrator must have the `solaris.jobs.user` authorization. To list a job in another user's `crontab` file, the administrator must have the `solaris.jobs.admin` authorization. No authorization is needed to list a user's own jobs.

`modify` Modifies a job in the `crontab`(1) database. To modify a job, the administrator must have the `solaris.jobs.user` authorization. To modify a job in another user's `crontab` file, the administrator must have the `solaris.jobs.admin` authorization.

Options The `smcron` authentication arguments, *auth_args*, are derived from the `smc(1M)` arg set and are the same regardless of which subcommand you use. The `smcron` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- —` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- —domain` with the *domain* argument.

`-D | - —domain` *13;domain* Specifies the default domain that you want to manage. `smcron` accepts only `file` for this option. `file` is also the default value.

The `file` default domain means that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.

`-H | - –hostname` *13;host_name:port*
: Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc(1M) -B` option, or set your console preferences to load a "home toolbox" by default.

`-l | - –rolepassword` *13;role_password*
: Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-p | - –password` *13;password*
: Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-r | - –rolename` *13;role_name*
: Specifies a role name for authentication. If you do not specify this option, no role is assumed.

`-u | - –username` *13;user_name*
: Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

`- –`
: This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- –` option.

subcommand_args
: For the time-related subcommands described below, `-m`, `-M`, `-t`, and `-w`, you can enter multiple arguments, separated only by commas. `smcron` will construct `crontab` entries appropriate for your arguments. See EXAMPLES.

Note: Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

- For subcommand `add`:

  `-c` *command* — Specifies the command that you want to run.

  `-h` — (Optional) Displays the command's usage statement.

  `-m` *day_of_month* — (Optional) Specifies the day of the month you want to run the job. Valid values are 1–31. If you specify both `-t` and `-m` options, the job executes one day per month at the time specified by `-t`.

  `-M` *month* — (Optional) Specifies the month that you want to run the job. Valid values are 1–12. If you specify both `-t` and `-M` options, the job executes during the specified month at the time specified by `-t`.

  `-n` *name* — Specifies the unique name of the job.

  `-o` *owner* — (Optional) Specifies the user name that is the owner of the job. If you do not specify this option, the user name specified by the `-U` option is assumed.

  `-t` *time_of_day* — Specifies the time (in *hh*:*mm*) that you want to execute the command. If no other time-related options are specified (`-m`, `-M`, or `-w`), the job executes every day at the time specified by `-t`. If you specify both `-t` and `-w` options, the job executes one day per week at the time specified by `-t`. If you specify both `-t` and `-m` options, the job executes one day per month at the time specified by `-t`. If you specify both `-t` and `-M` options, the job executes each day during the specified month at the time specified by `-t`.

  `-w` *day_of_week* — (Optional) Specifies the day of the week you want to execute the command. Valid values are as follows:

  - `0`=Sunday
  - `1`=Monday
  - `2`=Tuesday
  - `3`=Wednesday
  - `4`=Thursday
  - `5`=Friday
  - `6`=Saturday

  If you specify both `-t` and `-w` options, the job executes one day per week at the time specified by `-t`.

- For subcommand `delete`:

  `-h` — (Optional) Displays the command's usage statement.

  `-n` *name* — Specifies the unique name of the job.

  `-o` *owner* — (Optional) Specifies the user name that is the owner of the job. If you do not specify this option, the user name specified by the `-U` option is assumed.

- For subcommand `list`:

  `-f` *n*|*s*|*v* (Optional) Specifies the format of the output. See EXAMPLES for examples of each output type.

  - *n* — Displays the data in native format, as it appears in the `crontab(1)` database.
  - *s* — Default format. Displays the data in summary format.
  - *v* — Displays the data in verbose format.

  `-h` (Optional) Displays the command's usage statement.

  `-o` *owner* (Optional) Lists all jobs for the specified owner (user name). If you do not specify this option, all jobs in the `crontab`(1) database are listed.

- For subcommand `modify`:

  `-c` *command* (Optional) Specifies the command that you want to run.

  `-h` (Optional) Displays the command's usage statement.

  `-m` *day_of_month* (Optional) Specifies the day of the month you want to run the job. Valid values are 1–31. If you specify both `-t` and `-m` options, the job executes one day per month at the time specified by `-t`.

  `-M` *month* (Optional) Specifies the month that you want to run the job. Valid values are 1–12. If you specify both `-t` and `-M` options, the job executes during the specified month at the time specified by `-t`.

  `-n` *name* Specifies the current unique name of the job.

  `-N` *new_name* (Optional) Specifies the new unique name of the job.

  `-o` *owner* (Optional) Specifies the user name that is the owner of the job. If you do not specify this option, the user name specified by the `-U` option is assumed.

  `-O` *new_owner* (Optional) Specifies the new owner of the job.

  `-t` *time_of_day* (Optional) Specifies the time (in *hh*:*mm*) that you want to execute the command. If no other time-related options are specified (`-m`, `-M`, or `-w`), then the job executes every day at the time specified by `-t`. If you specify both `-t` and `-w` options, the job executes one day per week at the time specified by `-t`. If you specify both `-t` and `-m` options, the job executes one day per month at the time specified by `-t`. If you specify both `-t` and `-M`, then the job executes each day during the specified month at the time specified by `-t`.

  `-w` *day_of_week* (Optional) Specifies the day of the week you want to execute the command. Valid values are as follows:

  - `0`=Sunday

- 1=Monday
- 2=Tuesday
- 3=Wednesday
- 4=Thursday
- 5=Friday
- 6=Saturday

If you specify both `-t` and `-w` options, the job executes one day per week at the time specified by `-t`.

Examples

EXAMPLE 1 Adding a Job

The following adds a new job, owned by `root`, that removes the old log files from `/tmp` daily at 1:30 AM.

```
./smcron add -H myhost -u root -p mypassword -- -n "Remove old logs" \
             -t 1:30 -c "rm /tmp/*.log" -o root
```

EXAMPLE 2 Deleting a Job

The following deletes the job `Remove old logs` owned by `root`:

```
./smcron delete -H myhost -u root -p mypassword -- \
          -n "Remove old logs" -o root
```

EXAMPLE 3 Listing Jobs in Native Format

The following lists all jobs in native, or `crontab`(1), format:

```
./smcron list -H myhost -u root -p mypassword -- -f n
MINUTE HOUR DATE MONTH DAY COMMAND

10 3 * * * /usr/sbin/logadm
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
```

EXAMPLE 4 Listing Jobs in Standard Format

The following lists all jobs owned by `lp` in standard format:

```
./smcron list -H myhost -u root -p mypassword -- -f s -o lp
NAME::OWNER::SCHEDULE::COMMAND

NoName_1765663371::lp::Weekly on Sundays at 3:13 AM::cd /var/lp/logs;
   if [ -f requests ]; then if [ -f requests.1 ]; then /bin/mv requests.1
   requests.2; fi; /usr/bin/cp requests requests.1; > requests; fi
NoName_512822673::lp::Weekly on Sundays at 4:15 AM::cd /var/lp/logs;
   if [ -f lpsched ]; then if [ -f lpsched.1 ]; then /bin/mv lpsched.1
   lpsched.2; fi; /usr/bin/cp lpsched lpsched.1; >lpsched; fi
```

EXAMPLE 5 Listing jobs in verbose format

The following lists all jobs in verbose format:

```
./smcron list -H myhost -u root -p mypassword -- -f v
NAME::OWNER::SCHEDULE::NEXT_RUN::STATUS::COMMAND

NoName_1075488942::root::Advanced::::Finished on Feb 10 3:10 with code 1
::/etc/cron.d/logchecker
databackup::root::Weekly on Sundays at 3:10 AM::3/19/00 3:10 AM
::Finished on Sep 19 3:10::/usr/lib/newsyslog
runlog::root::Daily at 2:01 AM::3/14/00 2:01 AM::Finished on Feb 11
 2:01 AM::/usr/sbin/rtc
```

EXAMPLE 6 Changing a Job

The following modifies the job `Remove old logs` owned by `root` to execute daily at 2:00 AM:

```
./smcron modify -H myhost -u root -p mypassword -- -n "Remove old logs" \
          -o root -t 2:00
```

EXAMPLE 7 Specifying Multiple Time Arguments

`smcron` allows you to specify a range of times for all of its time-related subcommands, `-m`, `-M`, `-t`, and `-w`. For example, the following command:

```
# smcron add -u root -p xxxx -- -n cronjob1 -w 1-4,5 \
-t 12:00,13:15,14:30 -c ls
```

...creates the following entry in `crontab`:

```
0,15,30 12,13,14 * * 1,2,3,4,6 ls #cronjob1
```

This job would run on Monday through Thursday and Saturday at the following times:

```
12:00 12:15 12:30
13:00 13:15 13:30
14:00 14:15 14:30
```

Environment Variables

See `environ(5)` for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smcron` command. If this environment variable is not specified, the `/usr/java` location is used. See `smc(1M)`.

Exit Status

The following exit values are returned:

`0` Successful completion.

1 Invalid command syntax. A usage message displays.

2 An error occurred while executing the command. An error message displays.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga

See Also crontab(1), cron(1M), smc(1M), attributes(5), environ(5)

Notes The timezone of the `cron` daemon sets the system-wide timezone for `cron` entries. This, in turn, is by set by default system-wide using `/etc/default/init`.

If some form of *daylight savings* or *summer/winter time* is in effect, then jobs scheduled during the switchover period could be executed once, twice, or not at all.

Name smcwebserver – manage the server for the Sun Web Console

Synopsis `/usr/sbin/smcwebserver` *subcommand options*

Description The `smcwebserver` utility manages the Sun Web Console server. Sun Web Console is a browser-based interface that performs systems management. System administrators can manage systems, devices and services from the console.

When the console webserver is running, you can view the console by opening a browser and pointing to:

`https://`*host*`:6789`

host is the machine where the console has been installed and the console server is running.

The `smcwebserver` service is managed by the service management facility (SMF) (see `smf(5)`), under the fault management resource identifier (FMRI):

`svc:/system/webconsole`

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. `smcwebserver` can only be started by the superuser or someone in the Primary Administrator role.

The configuration properties of this service can be modified with `svccfg(1M)`.

`smcwebserver` supports an SMF property that controls remote access to the Sun Web Console server. The property, `options/tcp_listen`, has a default value of `false`, which disallows remote access. The value `true` allows remote access. See EXAMPLES.

Subcommands The following subcommands are supported:

disable — Disable automatic start or stop during system boot or shutdown. Until the administrator reruns the script with the `smcwebserver enable` subcommand the webserver can be started/stopped only when the administrator executes the script manually using the following command:

```
# /usr/sbin/smcwebserver [start | stop]
```

`enable` — Enable the webserver to startup automatically during subsequent system boot and gracefully stop during system shutdown.

`restart` — Stop and subsequently start the console webserver.

The format of the restart subcommand is:

`restart` [`-U` *username*]

`start` — Start the console webserver.

The format of the start subcommand is:

`start` [`-U` *username*]

`stop` Stop the console webserver.

`status` Display status of the console webserver.

The format of the status subcommand is:

`status` [`-p`]

Options The following options are supported:

`-U` *username* | `--username` *username* — The user identity to run the server as. Once the server has successfuly started under the specified identity, all subsequent starts will automatically be done under that identity until you change it via this option, or by changing the com.sun.web.console.user configuration property via the smreg(1M) command. The default is to run the server under the `noaccess` identity.

`-p` | `--parseable` — Display non-localized output suitable for programmatic parsing. If the server is running, the output will be:

running=yes

If the server is not running, the output will be:

running=no

`-h` | `--help` | `-?` — Display the usage statement.

`-V` | `--version` — Display console version information.

Examples **EXAMPLE 1** Displaying the Usage Statement

The following command displays the `smcwebserver` usage statement:

```
% smcwebserver --help
```

EXAMPLE 2 Determining if the Server is Running

The following shell command will start the server if it is not already running.

```
ans=‘smcwebserver -p | grep running | cut -d"=" -f2‘
if [ "$ans" = "no" ]; then
    smcwebserver start
fi
```

EXAMPLE 3 Using SMF Property to Allow Remote Access

The following sequence of commands allows remote access to the Sun Web Console server.

```
# svccfg -s svc:/system/webconsole \
    setprop options/tcp_listen = true
# svcadm refresh svc:/system/webconsole
```

Environment Variables

See `environ(5)` for descriptions of the following environment variables that affect the execution of `smcwebserver`:

`JAVA_HOME` If you do not specify this environment variable, your `PATH` is searched for a suitable java. Otherwise, depending on the OS, the following default locations are used:

Solaris: `/usr/j2se`

Linux: `/usr/java/j2sdk1.4*`

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmcon
Interface Stability	Stable

See Also

`smreg(1M)`, `svcadm(1M)`, `svccfg(1M)`, `attributes(5)`, `environ(5)`, `smf(5)`

Name smdiskless – manage diskless client support for a server

Synopsis `/usr/sadm/bin/smdiskless` *subcommand* `[` *auth_args*`] - –` `[`*subcommand_args*`]`

Description The `smdiskless` command manages diskless client support for a server.

`smdiskless` *subcommand*s are:

`add` Adds a new diskless client to a server. There are two usages for this command. The user can either specify all the optional arguments directly on the command line, or provide a `sysidcfg(4)` formatted file as input. A future enhancement will allow specifying both a `sysidcfg`(4) formatted file and optional arguments, which will override the values in the `sysidcfg`(4) file.

`delete` Deletes an existing diskless client from the system databases and removes any server support associated with the host, depending on the *os_server* type.

`list` Lists existing diskless clients served by *os_server*.

`modify` Modifies the specified attributes of the diskless client *os_server*.

Options The `smdiskless` authentication arguments, *auth_args*, are derived from the `smc(1M)` arg set and are the same regardless of which subcommand you use. The `smdiskless` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- –` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- –domain`.

Note – `smdiskless` supports the `-–auth-data` *file* option, which enables you to specify a file the console can read to collect authentication data. See `smc(1M)` for a description of this option.

`-D` | `- –domain` *13;domain* Specifies the default domain that you want to manage. The syntax of *domain* is *type*:/*host_name*/*domain_name*, where *type* is `nis`, `nis+`, `dns`, `ldap`, or `file`; *host_name* is the name of the machine that serves the domain; and *domain_name* is the name of the domain you want to manage. (*Note:* Do *not* use `nis+` for `nisplus`.)

If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.

`-H | - —hostname` *13;host_name:port*
: Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc(1M) -B` option, or set your console preferences to load a "home toolbox" by default.

`-l | - —rolepassword` *13;role_password*
: Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-p | - —password` *13;password*
: Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-r | - —rolename` *13;role_name*
: Specifies a role name for authentication. If you do not specify this option, no role is assumed.

`-u | - —username` *13;user_name*
: Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

`- —`
: This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- —` option.

subcommand_args
: *Note:* Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

- For subcommand `add`:

  `-h` (Optional) Displays the command's usage statement.

  `-i` *IP_address* Specifies the IP address for the host in the form of `172.16.200.1`.

  `-e` *ethernet_addr* Specifies the Ethernet address.

  `-n` *host* Specifies the client name.

  `-o` *os_server* (optional) Specifies the name of the host where the OS service filesystems reside. If this option is not specified, the host will be the same as that specified in the `smc(1M)` `-D` option. This option is useful in the event that the name service server and the OS server are not the same machine.

  `-x` os=*platform* Specifies the operating system. The syntax for *platform* is as follows:

  instruction_set.*implementation*.Solaris_*version*

  where

  - *instruction_set* is one of `sparc` or `i386`
  - *implementation* is the implementation architecture, that is, `i86pc` and `sun4u`.
  - *version* is the Solaris version number. The supported *version* numbers are `2.6`, `2.7` (for Solaris 7), `8`, `9`, and `10`. Examples are:

    ```
    sparc.sun4u.Solaris_8
    ```

  `-x` root=*pathname* (Optional) Specifies the absolute path of the directory in which to create the root directory for diskless clients. The default (and recommended) *pathname* is `/export/root/`*client_name*.

  `-x` swap=*pathname* (Optional) Specifies the absolute path of the directory in which to create the swap file for diskless clients. The default (and recommended) *pathname* is `/export/swap/`*client_name*.

  `-x` swapsize=*size* (Optional) Specifies the size, in megabytes, of the swap file for diskless clients. The default swap size is `24M`.

  `-x` dump=*pathname* (Optional) Specifies the absolute path of the dump directory for diskless clients. The default (and recommended) *pathname* is `/export/dump/`*client_name*.

  `-x` dumpsize=*size* (Optional) Specifies the size, in megabytes, of the dump file for diskless clients. The default swap size is `24M`.

  `-x` pw=*Y* (Optional) Prompts for the system's root password. The default is not to prompt.

The following options are used to configure workstations on first boot by sysidtool(1M). They can either be specified on the command line, or in a sysidcfg(4) formatted file. *Note:* Use the sysidcfg(4) file to:

- Add a DNS client.
- Specify use of the LDAP name service.
- Specify a security policy.

The keywords and functions supported by sysidtool and sysidcfg vary among Solaris releases. Consult the man pages for your operating system release (uname -r) to determine the level of support available.

-x tz=*timezone*	(Optional) Specifies the path of a timezone file, relative to /usr/share/lib/zoneinfo. The default is the server's timezone.
-x ns=NIS | NIS+ | NONE	(Optional) Specifies the client's nameservice. This is one of NIS, NIS+, or NONE . Use a sysidcfg(4) file to specify DNS or LDAP. The default ns value is NONE, which results in the use of the files source in nsswitch.conf. See nsswitch.conf(4) for a description of the files source.
-x nameserver=*hostname*	(Optional) Specifies the nameserver's hostname. The default is the server's nameserver.
-x domain=*domain*	(Optional) Specifies the client's domain. The default is the server's domain.
-x nameserver_ipaddress=*ip_address*	(Optional) Specifies the nameserver's IP address.
-x netmask=*ip_address*	(Optional) Specifies the client's IP address netmask. The default is the server's netmask.
-x locale=*locale*	(Optional) Specifies the client's system locale. The default is the C locale.
-x terminal=*term*	(Optional) Specifies the workstation's terminal type, typically, sun or xterms.
-x passwd=*root_password*	(Optional) Specifies the system's root password. The default is no password.
-x sysidcfg=*path_to_sysidcfg_file*	(Optional) Specifies the file to be placed in the /etc directory of the diskless client. On first boot, /etc/.UNCONFIGURED exists and sysidtool(1M) will run. If a file called

`/etc/sysidcfg` exists, `sysidtool`(1M) reads this file and uses the information for system configuration.

- For subcommand `delete`:

  `-h` (Optional) Displays the command's usage statement.

  `-n` *host* Specifies the hostname of the diskless client to delete. This host is deleted from relevant tables and OS Services for this client are deleted.

  `-o` *os_server* (Optional) Specifies the name of the host where the OS service filesystems reside. If this option is not specified, the host will be the same as that specified in the `smc`(1M) `-D` option. This option is useful in the event that the name service server and the OS server are not the same machine.

- For subcommand `list`:

  `-h` (Optional) Displays the command's usage statement.

  `-o` *os_server* (Optional) Specifies the name of the host where the OS service filesystems reside. If this option is not specified, the host will be the same as that specified in the `smc`(1M) `-D` option. This option is useful in the event that the name service server and the OS server are not the same machine.

- For subcommand `modify`:

  `-e` *ethernet_addr* Changes the specified diskless client's ethernet address to *ethernet_addr*.

  `-h` (Optional) Displays the command's usage statement.

  `-n` *host* Specifies the host name of the diskless client to modify.

  `-o` *os_server* (Optional) Specifies the name of the host where the OS service filesystems reside. If this option is not specified, the host will be the same as that specified in the `smc`(1M) `-D` option. This option is useful in the event that the name service server and the OS server are not the same machine.

  `-x` tz=*timezone* (Optional) Changes the specified diskless client's timezone.

Examples

EXAMPLE 1 Creating a new diskless client

The following command adds a new diskless client named `client1` which will run Solaris 10 on a `sun4u` machine:

```
example% /usr/sadm/bin/smdiskless add -- -i 172.16.200.1 \
            -e 8:0:11:12:13:14 -n client1 -x os=sparc.sun4u.Solaris_10 \
            -x root=/export/root/client1 -x swap=/export/swap/client1 \
            -x swapsize=32 -x tz=US/Eastern -x locale=en_US
```

EXAMPLE 2 Deleting an existing diskless client

The following command deletes the diskless client named `client1` from the OS server named `osserver`, where the OS server is using NIS+ and the NIS+ server is `nisplusserve`:

```
example% /usr/sadm/bin/smdiskless delete \
            -D nisplus:/nisplusserver/my.domain.com -- \
            -o osserver -n client1
```

EXAMPLE 3 Listing the diskless clients served by a host

The following command lists the diskless clients running on the OS server, `osserver`:

```
example% /usr/sadm/bin/smdiskless list -D file:/osserver/osserver -- \
            -o osserver
```

EXAMPLE 4 Modifying the attributes of the diskless client host

The following command modifies the ethernet address for the client named `client1` on the OS server, `osserver`, to be `8:0:11:12:13:15`:

```
example% /usr/sadm/bin/smdiskless modify -D file:/osserver/osserver -- \
            -o osserver -n client1 -e 8:0:11:12:13:15
```

Environment Variables

See `environ(5)` for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smdiskless` command. If this environment variable is not specified, the `/usr/java1.2` location is used. See `smc(1M)`.

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` Invalid command syntax. A usage message displays.

`2` An error occurred while executing the command. An error message displays.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdclnt

See Also

`smc(1M)`, `smosservice(1M)`, `sysidtool(1M)`, `nsswitch.conf(4)`, `sysidcfg(4)`, `attributes(5)`, `environ(5)`

Name smexec – manage entries in the exec_attr database

Synopsis `/usr/sadm/bin/smexec` *subcommand* `[` *auth_args*`] - –` `[`*subcommand_args*`]`

Description The `smexec` command manages an entry in the `exec_attr(4)` database in the local `/etc` files name service or a NIS or NIS+ name service.

subcommands `smexec` *subcommand*s are:

`add` Adds a new entry to the `exec_attr(4)` database. To add an entry to the `exec_attr` database, the administrator must have the `solaris.profmgr.execattr.write` authorization.

`delete` Deletes an entry from the `exec_attr(4)` database. To delete an entry from the `exec_attr` database, the administrator must have the `solaris.profmgr.execattr.write` authorization.

`modify` Modifies an entry in the `exec_attr(4)` database. To modify an entry in the `exec_attr` database, the administrator must have the `solaris.profmgr.execattr.write` authorization.

Options The `smexec` authentication arguments, *auth_args*, are derived from the `smc(1M)` arg set and are the same regardless of which subcommand you use. The `smexec` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- –` option.

auth_args The *auth_args* `-D`, `-H`, `-l`, `-p`, `-r`, and `-u` are described below. They are all optional. These options are a subset of the full complement of supported options described in `smc(1M)`.

If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- –domain` with the *domain* argument.

`-D` | `- –domain` *13;domain* Specifies the default domain that you want to manage. The syntax of *domain* is *type*:/*host_name*/*domain_name*, where *type* is `nis`, `nisplus`, `dns`, `ldap`, or `file`; *host_name* is the name of the machine that serves the domain; and *domain_name* is the name of the domain you want to manage. (*Note:* Do *not* use `nis+` for `nisplus`.)

If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.

`-H | - —hostname` *13;host_name:port*
: Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc(1M) -B` option, or set your console preferences to load a "home toolbox" by default.

`-l | - —rolepassword` *13;role_password*
: Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-p | - —password` *13;password*
: Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-r | - —rolename` *13;role_name*
: Specifies a role name for authentication. If you do not specify this option, no role is assumed.

`-u | - —username` *13;user_name*
: Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

`- —`
: This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- —` option.

subcommand_args
: *Note:* Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

To add or change privileges, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.

- For subcommand `add`:

  `-c` *command_path*|*CDE_action* — Specifies the full path to the command or CDE action associated with the new `exec_attr` entry. Specifying a CDE action is available only if the system is configured with Solaris Trusted Extensions. See "Using Options that Require Solaris Trusted Extensions," below.

  `-g` *egid* — (Optional) Specifies the effective group ID that executes with the command.

  `-G` *gid* — (Optional) Specifies the real group ID that executes with the command.

  `-h` — (Optional) Displays the command's usage statement.

  `-n` *profile_name* — Specifies the name of the profile associated with the new `exec_attr` entry.

  `-t` *type* — Specifies the type for the command. Currently, the only acceptable value for *type* is `cmd`.

  `-u` *euid* — (Optional) Specifies the effective user ID that executes with the command.

  `-U` *uid* — (Optional) Specifies the real user ID that executes with the command.

  `-M` *limit_privs* — Specifies the privilege name(s) to add to the new `exec_attr(4)` entry. The default is `all` for *limit* privilege.

  To add or change privileges, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.

  `-I` *inheritable_privs* — Specifies the inheritable privilege name(s) to add to the new `exec_attr(4)` entry.

- For subcommand `delete`:

  `-c` *command_path*|*CDE_action* — Specifies the full path to the command or CDE action associated with the `exec_attr` entry. Specifying a CDE action is available only if the system is

configured with Solaris Trusted Extensions. See "Using Options that Require Solaris Trusted Extensions," below.

`-h` — (Optional) Displays the command's usage statement.

`-n` *profile_name* — Specifies the name of the profile associated with the `exec_attr` entry.

`-t` *type* — Specifies the type `cmd` for command. Currently, the only acceptable value for *type* is `cmd`.

- For subcommand `modify`:

  `-c` *command_path*|*CDE_action* — Specifies the full path to the command or CDE action associated with the `exec_attr` entry you want to modify. Specifying a CDE action is available only if the system is configured with Solaris Trusted Extensions. See "Using Options that Require Solaris Trusted Extensions," below.

  `-g` *egid* — (Optional) Specifies the new effective group ID that executes with the command.

  `-G` *gid* — (Optional) Specifies the new real group ID that executes with the command.

  `-h` — (Optional) Displays the command's usage statement.

  `-n` *profile_name* — Specifies the name of the profile associated with the `exec_attr` entry.

  `-t` *type* — Specifies the type `cmd` for command. Currently, the only acceptable value for *type* is `cmd`.

  `-u` *euid* — (Optional) Specifies the new effective user ID that executes with the command.

  `-U` *uid* — (Optional) Specifies the new real user ID that executes with the command.

  `-M` *limit_privs* — Specifies the privilege name(s) to modify in an `exec_attr(4)` entry. The default is `all` for *limit* privilege.

  To add or change privileges, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.

  `-I` *inheritable_privs* — Specifies the inheritable privilege name(s) to modify in an`exec_attr(4)` entry.

Using Options that Require Solaris Trusted Extensions

To use an option that requires the Solaris Trusted Extensions feature, you must use the `-B` *toolbox* option to specify a toolbox that contains support for Trusted Extensions. For example:

```
# smexec -add -c <CDE action> -n "User Manager" \
-B http://<server>/toolboxes/tsol_files.tbx
```

In the command above, <*server*> is the name of the machine running the Solaris Management Console. See smc(1M) for a description of the `-B` option.

Examples

EXAMPLE 1 Creating an `exec_attr` Database Entry

The following creates a new `exec_attr` entry for the `User Manager` profile on the local file system. The entry type is `cmd` for the command `/usr/bin/cp`. The command has an effective user ID of `0` and an effective group ID of `0`.

```
./smexec add -H myhost -p mypasswd -u root -- -n "User Manager" \
          -t cmd -c /usr/bin/cp -u 0 -g 0
```

EXAMPLE 2 Deleting an `exec_attr` Database Entry

The following example deletes an `exec_attr` database entry for the `User Manager` profile from the local file system. The entry designated for the command `/usr/bin/cp` is deleted.

```
./smexec delete -H myhost -p mypasswd -u root -- -n "User Manager" \
          -t cmd -c /usr/bin/cp
```

EXAMPLE 3 Modifying an `exec_attr` Database Entry

The following modifies the attributes of the `exec_attr` database entry for the `User Manager` profile on the local file system. The `/usr/bin/cp` entry is modified to execute with the real user ID of `0` and the real group ID of `0`.

```
./smexec modify -H myhost -p mypasswd -u root -- -n "User Manager" \
          -t cmd -c /usr/bin/cp -U 0 -G 0
```

Environment Variables

See environ(5) for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smexec` command. If this environment variable is not specified, the `/usr/java` location is used. See smc(1M).

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` Invalid command syntax. A usage message displays.

`2` An error occurred while executing the command. An error message displays.

Files

The following file is used by the `smexec` command:

`/etc/security/exec_attr` Rights profiles database. See exec_attr(4).

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga
Interface Stability	Evolving

See Also smc(1M), exec_attr(4), attributes(5), environ(5)

Name smgroup – manage group entries

Synopsis `/usr/sadm/bin/smgroup` *subcommand* `[` *auth_args*`] - –` `[`*subcommand_args*`]`

Description The `smgroup` command manages one or more group definitions in the group database for the appropriate files in the local `/etc` files name service or a NIS or NIS+ name service.

The following `smgroup` *subcommand*s are supported

`add` Adds a new group entry. To add an entry, the administrator must have the `solaris.admin.usermgr.write` authorization.

`delete` Deletes a group entry. You can delete only one entry at a time. To delete an entry, the administrator must have the `solaris.admin.usermgr.write` authorization. *Note:* You cannot delete the system groups with IDs less than 100, or the groups 60001, 60002, or 65534.

`list` Lists one or more group entries in the form of a three-column list, containing the group name, group ID, and group members, separated by colons (`:`). To list entries, the administrator must have the `solaris.admin.usermgr.read` authorization.

`modify` Modifies a group entry. To modify an entry, the administrator must have the `solaris.admin.usermgr.write` authorization.

Options The `smgroup` authentication arguments, *auth_args*, are derived from the `smc(1M)` arg set and are the same regardless of which subcommand you use. The `smgroup` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- –` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- –domain`.

The following *auth_args* are supported:

`-D` | `- –domain` *13;domain* Specifies the default domain that you want to manage. The syntax of *domain* is *type*:/*host_name*/*domain_name*, where *type* is `nis`, `nisplus`, `dns`, `ldap` or `file`; *host_name* is the name of the machine that serves the domain;

	and *domain_name* is the name of the domain you want to manage. (*Note:* Do *not* use `nis+` for `nisplus`.) If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.
`-H | - —hostname` *13;host_name:port*	Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name*:*port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc(1M) -B` option, or set your console preferences to load a "home toolbox" by default.
`-l | - —rolepassword` *13;role_password*	Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-p | - —password` *13;password*	Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-r | - —rolename` *13;role_name*	Specifies a role name for authentication. If you do not specify this option, no role is assumed.
`-u | - —username` *13;user_name*	Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.
`- —`	This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- —` option.

subcommand_args Descriptions and other argument options that contain white spaces must be enclosed in double quotes.

The `add` subcommand supports the following *subcommand_args*:

`-g` *gid*	(Optional) Specifies the group ID for the new group. The group ID must be a non-negative decimal integer with a maximum value of 2MB (2,147,483,647). Group IDs 0–99 are reserved for the system and should be used with care. If you do not specify a *gid*, the system automatically assigns the next available *gid*. To maximize interoperability and compatibility, administrators are recommended to assign groups using the range of GIDs below 60000 where possible.
`-h`	(Optional) Displays the command's usage statement.
`-m` *group_member1* `-m` *group_member2* . . .	(Optional) Specifies the new members to add to the group.
`-n` *group_name*	Specifies the name of the new group. The group name must be unique within a domain, contain 2–32 alphanumeric characters, begin with a letter, and contain at least one lowercase letter.

The `delete` subcommand supports the following *subcommand_args*:

`-h`	(Optional) Displays the command's usage statement.
`-n` *group_name*	Specifies the name of the group you want to delete.

The `list` subcommand supports the following *subcommand_args*

`-h`	(Optional) Displays the command's usage statement.
`-n` *group_name*	(Optional) Specifies the name of the group you want to list. If you do not specify a group name, all groups are listed.

The `modify` subcommand supports the following *subcommand_args*

`-h`	(Optional) Displays the command's usage statement.

`-m` *group_member1* `-m` *group_member2* . . .	(Optional) Specifies the new members to add to the group. Note that *group_member* overwrites the existing member list in the group file.
`-n` *group_name*	Specifies the name of the group you want to modify.
`-N` *new_group*	(Optional) Specifies the new group name. The group name must be unique within a domain, contain 2–32 alphanumeric characters, begin with a letter, and contain at least one lowercase letter.

Examples

EXAMPLE 1 Creating a Test Group

The following creates the `test_group` group entry with a group ID of `123` and adds `test_member1` and `test_member2` to the group:

```
./smgroup add -H myhost -p mypasswd -u root -- -n test_group \
          -m test_member1 -m test_member2 -g 123
```

EXAMPLE 2 Deleting a Group

The following deletes `test_group`:

```
./smgroup delete -H myhost -p mypasswd -u root -- -n test_group
```

EXAMPLE 3 Displaying All Groups

The following displays all groups in a three-column list showing the group name, group ID, and group members:

```
./smgroup list -H myhost -p mypasswd -u root --
```

EXAMPLE 4 Displaying a Group

The following displays the `group_1` data in a three-column list showing the group name, group ID, and group members:

```
./smgroup list -H myhost -p mypasswd -u root -- -n group_1
```

EXAMPLE 5 Renaming a Group

The following renames a group from `finance` to `accounting`:

```
./smgroup modify -H myhost -p mypasswd -u root  -- \
          -n finance -N accounting
```

Environment Variables See environ(5) for a description of the JAVA_HOME environment variable, which affects the execution of the smgroup command. If this environment variable is not specified, the /usr/java location is used. See smc(1M).

Exit Status The following exit values are returned:

0 Successful completion.

1 Invalid command syntax. A usage message displays.

2 An error occurred while executing the command. An error message displays.

Files The following files are used by the smgroup command:

/etc/group Group file. See group(4).

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga

See Also smc(1M), group(4), attributes(5), environ(5)

Name smlog – manage and view WBEM log files

Synopsis

```
/usr/sadm/bin/smlog subcommand [auth_args] --
     [subcommand_args]
```

Description The `smlog` command manages WBEM log files and allows a user to view WBEM log file records.

Subcommands The `smlog` command supports the following subcommands:

backup — Backs up the entries in the current WBEM log file. The backup command then creates a new log file and makes this log file the current log file.

delete — Deletes an existing (backed up) WBEM log file.

list — Lists the names of all the WBEM log files available for viewing.

view — Allows the user to view the contents of the specified WBEM log file.

Options The `smlog` authentication arguments, *auth_args*, are derived from the `smc(1M)` `arg` set and are the same regardless of which subcommand you use. The `smlog` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- —` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- —domain` with the *domain* argument.

`-D | - —domain` *13;domain* — Specifies the default domain that you want to manage. `smlog` accepts only `file` for this option. `file` is also the default value.

The `file` default domain means that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.

`-H | - —hostname` *13;host_name:port* — Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a

toolbox to load into the console. To override this behavior, use the `smc -B` option (see `smc`(1M)), or set your console preferences to load a "home toolbox" by default.

`-l|- –rolepassword` *13;role_password*
: Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-p|- –password` *13;password*
: Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-r|- –rolename` *13;role_name*
: Specifies a role name for authentication. If you do not specify this option, no role is assumed.

`-u|- –username` *13;user_name*
: Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

`- –`
: This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- –` option.

subcommand_args
: Descriptions and other `arg` options that contain white spaces must be enclosed in double quotes.

The `backup` subcommand supports the following *subcommand_args*:

`-h`
: Displays the command's usage statement.

  This *subcommand_arg* is optional.

The `delete` subcommand supports the following *subcommand_args*

`-h`
: Displays the command's usage statement.

  This *subcommand_arg* is optional.

`-n` *name*
: Specifies the name of the log file you want to delete.

The `list` subcommand supports the following *subcommand_args*:

`-h` Displays the command's usage statement.

This *subcommand_arg* is optional.

The `view` subcommand supports the following *subcommand_args*:

`-h` Displays the command's usage statement.

This *subcommand_arg* is optional.

`-n` *name* Specifies the name of the log file you want to view.

`-v` Displays the data in verbose format.

This *subcommand_arg* is optional.

Examples

EXAMPLE 1 Listing WBEM Log Files

The following command lists all available WBEM log files:

```
./smlog list -H myhost -p mypasswd -u root --

Log.01/03/2001.14:38:29
Log.01/04/2001.16:34:59
Log.01/08/2001.14:13:33
Log.01/11/2001.18:39:53
Log.01/12/2001.10:31:31
Log.12/21/2000.17:41:11
```

EXAMPLE 2 Displaying a WBEM Log File

The following command displays the contents of a log file:

```
./smlog view -H myhost -p mypasswd -u root -- -n Log.01/04/2001.16:34:59

Date and Time       Client     User   Source            Severity       Cate
gory        Summary
1/5/01 5:22:47 PM   hostname1  root   Solaris_OsService Informational  Appl
ication log No services found.
1/5/01 5:21:46 PM   hostname1  root   Solaris_OsService Informational  Appl
ication log No services found.
```

The `smlog` output wraps when it exceeds `80` characters.

Environment Variables

See `environ(5)` for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smlog` command. If this environment variable is not specified, the `/usr/java1.2` location is used. See `smc(1M)`.

Exit Status

The following exit values are returned:

`0` Successful completion.

1 Invalid command syntax. A usage message displays.

2 An error occurred while executing the command. An error message displays.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga

See Also smc(1M), attributes(5), environ(5)

Name smmaillist – manage email alias entries

Synopsis `/usr/sadm/bin/smmaillist` *subcommand* `[` *auth_args*`] - –` `[`*subcommand_args*`]`

Description The `smmaillist` command manages one or more email alias entries for the appropriate files in the local `/etc` files name service or a NIS or NIS+ name service.

subcommands `smmaillist` *subcommand*s are:

`add` Creates a new email alias definition and adds it to the appropriate files. To add an entry, the administrator must have the `solaris.admin.usermgr.write` authorization.

`delete` Deletes an email alias entry. You can delete only one entry at a time. To delete an entry, the administrator must have the `solaris.admin.usermgr.write` authorization. *Note:* You cannot delete Postmaster or Mailer-Daemon aliases.

`list` Lists one or more email alias entries. To list an entry, the administrator must have the `solaris.admin.usermgr.read` authorization.

`modify` Modifies an email alias entry. To modify an entry, the administrator must have the `solaris.admin.usermgr.write` authorization.

Options The `smmaillist` authentication arguments, *auth_args*, are derived from the `smc`(1M) arg set and are the same regardless of which subcommand you use. The `smmaillist` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc`(1M)). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- –` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- –domain` with the *domain* argument.

`-D` | `- –domain` *13;domain* Specifies the default domain that you want to manage. The syntax of *domain* is *type:/host_name/domain_name*, where *type* is `nis`, `nisplus`, `dns`, `ldap`, or `file`; *host_name* is the name of the machine that serves the domain; and *domain_name* is the name of the domain you want to manage. (*Note:* Do *not* use `nis+` for `nisplus`.)

	If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.
`-H | - —hostname` *13;host_name:port*	Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc(1M) -B` option, or set your console preferences to load a "home toolbox" by default.
`-l | - —rolepassword` *13;role_password*	Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-p | - —password` *13;password*	Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-r | - —rolename` *13;role_name*	Specifies a role name for authentication. If you do not specify this option, no role is assumed.
`-u | - —username` *13;user_name*	Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.
`- —`	This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- —` option.

subcommand_args *Note:* Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

- For subcommand `add`:

  `-a` *address1* `-a` *address2* . . . (Optional) Specifies the new email address. See `sendmail(1M)`.

  `-h` (Optional) Displays the command's usage statement.

  `-n` *alias_name* Specifies the name of the alias you want to add. See `sendmail`(1M).

- For subcommand `delete`:

  `-h` (Optional) Displays the command's usage statement.

  `-n` *alias_name* Specifies the alias you want to delete.

- For subcommand `list`:

  `-h` (Optional) Displays the command's usage statement.

  `-n` *alias_name* (Optional) Specifies the name of the alias you want to display. If you do not specify an alias, all aliases are listed.

- For subcommand `modify`:

  `-a` *address1* `-a` *address2* . . . (Optional) Specifies new email address(es) to replace the existing one(s). See `sendmail(1M)`.

  `-h` (Optional) Displays the command's usage statement.

  `-n` *alias_name* (Optional) Specifies the name of the alias you want to modify.

  `-N` *new_alias_name* Specifies the new alias name. Use only when renaming an alias. See `sendmail`(1M).

Examples

EXAMPLE 1 Creating an alias

The following creates the `coworkers` alias and adds the following member list: `bill@machine1`, `sue@machine2`, and `me@machine3` to the alias.

```
./smmaillist add -H myhost -p mypasswd -u root -- -n coworkers \
          -a bill@machine1 -a sue@machine2 -a me@machine3
```

EXAMPLE 2 Deleting a mail alias

The following deletes the `my_alias` alias:

```
./smmaillist delete -H myhost -p mypasswd -u root -- -n my_alias
```

EXAMPLE 3 Displaying members of a mail alias

The following displays the list of members belonging to the `my_alias` alias:

```
./smmaillist list -H myhost -p mypasswd -u root -- -n my_alias
```

EXAMPLE 4 Displaying members of all mail aliases

The following displays the list of members belonging to all mail aliases:

```
./smmaillist list -H myhost -p mypasswd -u root --
```

EXAMPLE 5 Renaming a mail alias

The following renames the `current_name` mail alias to `new_name`:

```
./smmaillist modify -H myhost -p mypasswd -u root -- \
          -n current_name -N new_name
```

EXAMPLE 6 Redefining an address list

The following changes the recipients of the alias `my_alias` to `bill@machine1`. Any previous recipients are deleted from the alias.

```
./smmaillist modify -H myhost -p mypasswd -u root -- \
          -n my_alias -a bill@machine1
```

Environment Variables

See `environ(5)` for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smmaillist` command. If this environment variable is not specified, the `/usr/java` location is used. See `smc(1M)`.

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` Invalid command syntax. A usage message displays.

`2` An error occurred while executing the command. An error message displays.

Files

The following files are used by the `smmaillist` command:

`/var/mail/aliases` Aliases for `sendmail(1M)`. See `aliases(4)`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga

See Also

`sendmail(1M)`, `smc(1M)`, `aliases(4)`, `attributes(5)`, `environ(5)`

Name smmultiuser – manage bulk operations on user accounts

Synopsis

```
/usr/sadm/bin/smmultiuser subcommand [ auth_args] - —
    [subcommand_args]
```

Description The `smmultiuser` command allows bulk operations on user entries in the local `/etc` filesystem or a NIS or NIS+ name service, using either an input file or piped input. *Note:* Both input files and piped input contain a cleartext (non-encrypted) password for each new user entry.

subcommands `smmultiuser` *subcommand*s are:

`add` Adds multiple user entries to the appropriate files. To add an entry, the administrator must have the `solaris.admin.usermgr.write` authorization.

`delete` Deletes one or more user entries from the appropriate files. To delete an entry, the administrator must have the `solaris.admin.usermgr.write` authorization.

`modify` Modifies existing user entries in the user account database. To modify an entry, the administrator must have the `solaris.admin.usermgr.write` authorization. Here is the list of what can be modified using the `modify` subcommand:

1. UserName (only under certain conditions; see Note 2 in NOTES).
2. Password (only under certain conditions; see Note 3 in NOTES). To modify a password, the administrator must have the `solaris.admin.usermgr.pswd` authorization.
3. Description.
4. Primary Group ID.
5. Shell type.
6. FullName.

Options The `smmultiuser` authentication arguments, *auth_args*, are derived from the `smc`(1M) arg set and are the same regardless of which subcommand you use. The `smmultiuser` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc`(1M)). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- —` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, `- —trust`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- —domain`.

`-D` | `- —domain` *13;domain* Specifies the default domain that you want to manage. The syntax of *domain* is

	type:/*host_name*/*domain_name*, where *type* is `nis`, `nisplus`, `dns`, `ldap`, or `file`; *host_name* is the name of the machine that serves the domain; and *domain_name* is the name of the domain you want to manage. (*Note:* Do *not* use `nis+` for `nisplus`.) If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.
`-H | - —hostname` *13;host_name:port*	Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc(1M)` `-B` option, or set your console preferences to load a "home toolbox" by default.
`-l | - —rolepassword` *13;role_password*	Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-p | - —password` *13;password*	Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-r | - —rolename` *13;role_name*	Specifies a role name for authentication. If you do not specify this option, no role is assumed.
`- —trust`	Trusts all downloaded code implicitly. Use this option when running the terminal console non-interactively and you cannot let the console wait for user input.

If using piped input into any of the `smmultiuser` subcommands, it will now be necessary to use the `- —trust` option with the `-L` *logfile* option. See EXAMPLES.

`-u | - —username` *13;user_name* — Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

`- —` — This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- —` option.

subcommand_args *Note:* Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

- For subcommand `add`:

  `-h` — (Optional) Displays the command's usage statement.

  `-i` *input_file* — Specifies the input file containing the user account information. After the command is executed, the input file is removed. The input file must follow the `/etc/passwd` file format. If you do not specify the `-i` *input_file* option, you must include a *piped_input* operand immediately before the command. See EXAMPLES.

  `-L` *logfile* — (Optional) Specifies the full pathname to the text file that stores the command's success/failure data. *Note:* This text file is an ASCII—formatted log file; it is different from and unrelated to the output of the normal logging mechanism that also occurs within the Log Viewer tool. The `-L` *logfile* option is used to dump additional logging information to a text file.

- For subcommand `delete`:

  `-h` — (Optional) Displays the command's usage statement.

  `-i` *input_file* — Specifies the input file containing the user account information. After the command is executed, the input file is removed. The input file must follow the `/etc/passwd` file format. If you do not specify the `-i` *input_file* option, you must include a *piped_input* operand immediately before the command. See EXAMPLES.

  `-L` *logfile* — (Optional) Specifies the full pathname to the text file that stores the command's success/failure data.

- For subcommand `modify`:

  `-h` — (Optional) Displays the command's usage statement.

`-i` *input_file* Specifies the input file containing the user account information. After the command is executed, the input file is removed. The input file must follow the `/etc/passwd` file format. If you do not specify the `-i` *input_file* option, you must include a *piped_input* operand immediately before the command. See EXAMPLES. *Note:* When modifying passwords, use the piped input, since it is more secure than keeping passwords in a file. See Note 1 in NOTES.

`-L` *logfile* (Optional) Specifies the full pathname to the text file that stores the command's success/failure data.

Operands The following operands are supported:

piped_input You must include *piped_input* if you do not specify an *input_file*. Include the piped input immediately before the command. The piped input must follow the `/etc/passwd` file format. See EXAMPLES. *Note:* Use the `- –trust` option when using piped input with the `-L` *logfile* option to avoid the user prompt from the Security Alert Manager, which normally asks the user whether the log file should be created. Without the `- –trust` option, the piped input is improperly taken as the answer to the prompt before the user can answer "`Y`" or "`N`", and the logging operation will probably fail.

Examples **EXAMPLE 1** Creating multiple user accounts

The following reads in user account data from the `/tmp/foo` file and creates new user accounts on the local file system. The input file is formatted in the `/etc/passwd` format.

```
./smmultiuser add -H myhost -p mypasswd -u root --  -i /tmp/foo
```

EXAMPLE 2 Deleting multiple user accounts

The following reads in user account data from the `/tmp/foo` file and deletes the named user accounts from the local file system:

```
./smmultiuser delete -H myhost -p mypasswd -u root --  -i /tmp/foo
```

EXAMPLE 3 Creating a log file with piped input

The following example shows the use of the `smc`(1M) `- –trust` option that is required when creating a log file. It is applicable to the `delete` and `modify` subcommands also.

```
cat /tmp/users.txt | smmultiuser add --trust --  -L /tmp/mylog.txt
```

Environment Variables See `environ`(5) for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smprofile` command. If this environment variable is not specified, the `/usr/java` location is used. See `smc`(1M).

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Invalid command syntax. A usage message displays.

`2` An error occurred while executing the command. An error message displays.

Files The following files are used by the `smprofile` command:

`/etc/passwd` Contains the file format to use for the *input_file* and *piped_input*. See `passwd(4)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga

See Also `smc(1M)`, `passwd(4)`, `attributes(5)`, `environ(5)`

Notes

1. The file format used by both the `add` and `modify` subcommands is the `/etc/passwd` format. But there is an allowance for a mutated version of this file format that contains an extra field at the end of each line to be used for the Full Name. If the extra field is appended to the end of each line, it will be used for the Full Name value, but if it is omitted, it will be assumed that no FullName modification is being done. The extra field is separated with a colon (`:`), just like all the other fields.

   Example of regulation `/etc/passwd` entry:

   ```
   rick2:x:101:10:description1:/home/rick2:/bin/sh
   ```

   Example of `/etc/passwd` variant entry:

   ```
   rick2:x:101:10:description1:/home/rick2:/bin/sh:Ricks_fullname
   ```

2. The modifies are all done based on lookups of the user name in the user tables. If a user name can not be found in this lookup, a secondary check will be made to see if the uid *and* FullName can be found in the user tables. If they are both found, assume that a user rename has occurred. If neither can be found, assume that the user account does not exist and cannot be modified.
3. If no password is supplied, assume that there is no change to the password information. If a password is being changed, it should be supplied in cleartext as piped input, although this is not required. The password can be supplied in the input file also. Once read in, the password will be changed accordingly.

Name smosservice – manage OS services

Synopsis `/usr/sadm/bin/smosservice` *subcommand* `[` *auth_args*`] - –` `[`*subcommand_args*`]`

Description The `smosservice` command manages OS services.

`smosservice` *subcommand*s are:

`add` Adds the specified OS services.

`delete` Deletes the specified OS services.

`list` Either lists all the installed OS services for the server if you do not specify a hostname, or lists the OS services for the specified diskless client if you do specify a hostname.

`patch` Manages patches on all existing diskless clients. For example, you can use this subcommand to initially establish a patch spool directory on an OS server. Then, you can apply the patch to the spool area, verifying the patch as needed. Once the patch exists in the spool area, you can apply the patch to the clone area. In addition, you can migrate the patched clone area to clients.

Options The `smosservice` authentication arguments, *auth_args*, are derived from the `smc(1M)` arg set and are the same regardless of which subcommand you use. The `smosservice` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- –` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- –domain`.

`-D` | `- –domain` *13;domain* Specifies the default domain that you want to manage. The syntax of *domain* is *type*:/*host_name*/*domain_name*, where *type* is `nis`, `nis+`, `dns`, `ldap`, or `file`; *host_name* is the name of the machine that serves the domain; and *domain_name* is the name of the domain you want to manage. (*Note:* Do *not* use `nis+` for `nisplus`.)

If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.

`-H | - —hostname` *13;host_name:port*
: Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc(1M) -B` option, or set your console preferences to load a "home toolbox" by default.

`-l | - —rolepassword` *13;role_password*
: Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-p | - —password` *13;password*
: Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-r | - —rolename` *13;role_name*
: Specifies a role name for authentication. If you do not specify this option, no role is assumed.

`-u | - —username` *13;user_name*
: Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

`- —`
: This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- —` option.

subcommand_args
: *Note:* Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

- For subcommand `add`:

  `-h` (Optional) Displays the command's usage statement.

  `-o` *os_server* (Optional) Specifies the name of the host where the OS service filesystems reside. If this option is not specified, the host will be the same as that specified in the `smc`(1M) `-D` option. This option is useful in the event that the name service server and the OS server are not the same machine.

  `-x` mediapath=*path* Specifies the full path to the Solaris CD image.

  `-x` platform=*platform* Specifies the OS service to add. The instruction architecture, machine class, OS, and version are given in the form:

  instruction_set.machine_class.Solaris_*os_version*

  for example, `sparc.sun4m.Solaris_10`

  `-x` cluster=*cluster* Specifies the Solaris cluster to install. For example, `SUNWCall`.

  `-x` locale=*locale*[*locale*, . . .] (Optional) Specifies the locales to install from the specified cluster. A comma-delimited list of locales can be specified.

- For subcommand `delete`:

  `-h` (Optional) Displays the command's usage statement.

  `-o` *os_server* (Optional) Specifies the name of the host where the OS service filesystems reside. If this option is not specified, the host will be the same as that specified in the `smc`(1M) `-D` option. This option is useful in the event that the name service server and the OS server are not the same machine.

  `-x` rmplatform=*platform* Specifies the OS service to remove. The instruction architecture, machine class, OS, and version are given in the form:

  instruction_set.machine_class.Solaris_*os_version*

  for example, `sparc.all.Solaris_10`. *Note:* Only a machine class of `all` is supported.

- For subcommand `list`:

  `-h` (Optional) Displays the command's usage statement.

  `-o` *os_server* (Optional) Specifies the name of the host where the OS service filesystems reside. If this option is not specified, the host will be the same as that

specified in the smc(1M) -D option. This option is useful in the event that the name service server and the OS server are not the same machine.

- For subcommand patch:

  -a *patch_directory/patch_ID* — Adds the specified patch, *patch_ID*, to the spool directory. *patch_directory* specifies the source path of the patch to be spooled which includes the patchid directory name. Patches are spooled to /export/diskless/Patches/. If the patch being added obsoletes an existing patch in the spool, the obsolete patch is moved to the archive area, /export/diskless/Patches/Archive (to be restored if this new patch is ever removed).

  -h — (Optional) Displays the command's usage statement.

  -m — (Optional) Synchronizes spooled patches with offline copies of each diskless client OS service on the server. Spooled patches and applied patches are compared so that newly spooled patches can be installed and patches recently removed from the spool can be backed out. This option does not apply to patches directly to diskless client OS services or diskless clients; the -u option must be used to update the services and clients with the changes. Clients are not required to be down at this time, as all patching is done off line. *Note:* The server is fully available during this operation.

  -P — Lists all currently spooled patches with an associated synopsis. The list is split up into sections detailing the patches for each OS and architecture in this format:

```
Solaris os_rel1 architecture1:
patchid Synopsis
patchid Synopsis
......
Solaris os_rel1 architecture2:
patchid Synopsis
......
```

  -r *patchid* — Removes the specified patchid from the spool if it is not a requirement for any of the other patches in the spool. All archived patches that were obsoleted by the removed patch are restored to the spool.

  -U — (Optional) Updates all diskless client OS services and diskless clients with any changes after synchronizing

patches with the `-m` option. Clients must be brought down during this operation. Once execution has completed, each client should be booted again.

Examples

EXAMPLE 1 Creating a new OS service

The following command adds an OS service for Solaris 10 for the `sun4u` machine class where the OS server is *not* using a name service:

```
example% /usr/sadm/bin/smosservice add -- \
            -x mediapath=/net/imageserver/5.8/sparc \
            -x platform=sparc.sun4u.Solaris_10 \
            -x cluster=SUNWCXall -x locale=en_US
```

The following command adds an OS service for Solaris 10 for the `sun4u` machine class where the OS server is using NIS, the NIS server is `nisserver`, the OS server is `osserver`, and the port to which you connect on `osserver` is 898:

```
example% /usr/sadm/bin/smosservice add -D nis:/nisserver/my.domain.com -- \
            -H osserver:898 -- \
            -x mediapath=/net/imageserver/5.8/sparc \
            -x platform=sparc.sun4u.Solaris_10 \
            -x cluster=SUNWCXall -x locale=en_US \
            -o osserver
```

In the preceding example, the OS service is placed in `/export` on `osserver`, while the `hosts.byaddr`, `ethers`, and `bootparams` maps are updated on the NIS server.

EXAMPLE 2 Deleting an OS service

The following command deletes the OS service for Solaris 10 for the `sun4u` machine class where the OS server is using NIS, the NIS server is `nisserver`, and the OS server is `osserver`:

```
example% /usr/sadm/bin/smosservice delete\
            -D nis:/nisserver/my.domain.com -- \
            -x rmplatform=sparc.all.Solaris_10 \
            -o osserver
```

EXAMPLE 3 Listing installed OS services

The following command lists the OS services installed on the machine, `osserver`:

```
example% /usr/sadm/bin/smosservice list \
            -D file:/osserver/osserver -- -o osserver
```

Environment Variables

See `environ(5)` for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smosservice` command. If this environment variable is not specified, the `/usr/java1.2` location is used. See `smc(1M)`.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` Invalid command syntax. A usage message displays.

`2` An error occurred while executing the command. An error message displays.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdclnt

See Also `smc(1M)`, `smdiskless(1M)`, `attributes(5)`, `environ(5)`

Name smpatch – download, apply, and remove updates

Synopsis

```
/usr/sbin/smpatch add -i update-id [auth-opts]
     [-i update-id] ... [-d update-dir] [-b BE_name]
     [-n system-name] ... [-x mlist=system-list-file] [-V]

/usr/sbin/smpatch add -x idlist=update-list-file
     [auth-opts]
     [-d update-dir] [-n system-name] ...
     [-b BE_name] [-x mlist=system-list-file] [-V]

/usr/sbin/smpatch analyze [auth-opts] [-i update-id] ...
     [-n system-name] ...
     [-x idlist=update-list-file] [-V]

/usr/sbin/smpatch download [auth-opts] [-i update-id] ...
     [-d update-dir] [-f] [-n system-name] ...
     [-t] [-x idlist=update-list-file] [-V]

/usr/sbin/smpatch get [auth-opts] [-n system-name] ...
     [parameter-name]... [-V]

/usr/sbin/smpatch messages [-a] [-V]

/usr/sbin/smpatch order -i update-id [auth-opts]
     [-i update-id] ... [-d update-dir]
     [-n system-name] ... [-V]

/usr/sbin/smpatch order -x idlist=update-list-file
     [auth-opts]
     [-d update-dir] [-n system-name] ... [-V]

/usr/sbin/smpatch remove -i update-id [auth-opts]
     [-n system-name] ... [-b BE_name] [-V]

/usr/sbin/smpatch set [auth-opts] [-n system-name] ...
     parameter-name=parameter-value... [-V]

/usr/sbin/smpatch unset [auth-opts] [-n system-name] ...
     parameter-name... [-V]

/usr/sbin/smpatch update [auth-opts] [-i update-id] ...
     [-d update-dir] [-n system-name] ... [-b BE_name]
     [-x idlist=update-list-file] [-V]
```

Description The `smpatch` command manages the update process on a single system or on multiple systems. Use this command to download, apply, and remove updates. Also, use the `smpatch` command to configure the update management environment for your system.

If you want to run the `smpatch` command in remote mode, your system must run at least the Developer Solaris Software Group of the Solaris 10 system.

The `smpatch analyze` command determines the updates that are appropriate for the systems you want to update. The `smpatch` command can download and apply updates that you specify

on the command line. Or, `smpatch` can download and apply updates based on an analysis of one or more systems. Use the `-i` option or the `-x idlist=` option to specify the particular updates.

All of the systems on which you want to apply updates must be running the same version of the Solaris Operating System, have the same hardware architecture, and have the same updates applied.

The list of updates that is generated by the analysis is based on all of the available updates from the Sun update server. No explicit information about your host system or its network configuration is transmitted to Sun. Only a request for the Sun update set is transmitted. The update set is scanned for updates that are appropriate for this host system, the results are displayed, and those updates are optionally downloaded.

`smpatch` supports the Live Upgrade feature of the Solaris operating system (see `live_upgrade(5)`). Through the `add`, `remove`, and `update` subcommands, described below, `smpatch` enables you to perform operations on a boot environment (BE). A BE is an operating system image, consisting of a particular set of operating system and application software packages.

The `smpatch` command supports the following subcommands:

`add`
: Applies one or more updates to one or more systems. You must specify at least one update to apply. By default, updates are applied to the local system.

  This subcommand attempts to apply *only* the updates you specify. If you specify an update that depends on another that has not been applied, the `add` command fails to apply the update you specified.

  This subcommand does not apply updates based on the specified update policy. To apply updates based on the update policy, use the `update` subcommand.

  Use the `-i` or `-x idlist=` option to specify the updates to apply. Note that all of the updates you specify, and those on which they depend, must exist in the download directory.

  Use the `-n` or the `-x mlist=` option to specify the systems on which to apply updates.

  Optionally use the `-d` option to specify an alternate download directory.

  If the updates on which the specified updates depend are unavailable, run the `smpatch download` subcommand to obtain the updates you need.

  When you use the `-b` *BE_name* option, the `add` subcommand applies a specified update to the BE *BE_name*, rather than to the currently running operating system. Upon successful installation of the update, `smpatch` activates *BE_name* (see `luactivate(1M)`) and informs you that you can, at that point, boot from *BE_name*.

`analyze`
: Analyzes a system to generate a list of the appropriate updates.

After analyzing the system, use the `update` subcommand or the `download` and `add` subcommands to download and apply the updates to your systems.

The list of updates is written to standard output, so you can redirect standard output to a file to create an update list.

If you supply a list of one or more updates, the list is augmented with the updates on which those updates depend. The list is also put in an order suitable for applying updates.

The `smpatch analyze` command depends on network services that are not available while the system is in single-user mode.

`download`
Downloads updates from the Sun update server to a system. You can optionally specify which updates to download. You can also specify the name of a system and download the appropriate updates to that system.

Use the `-i` or `-x idlist=` option to specify the updates to download.

Use the `-f` option to force the download of the exact update revision specified by `-i` *update-id*. If `-i` *update-id* does not specify a revision, the highest revision of the update is downloaded.

Use the `-t` option to download the `README` file for the update specified by `-i` *update-id*. If `-i` *update-id* does not specify a revision, the `README` file for the highest revision of the update is downloaded.

Use the `-n` option to analyze a remote system and to determine which updates to download. The updates, and those on which they depend, are downloaded from the Sun update server to the download directory of the system you specified.

The `smpatch download` command depends on network services that are not available while the system is in single-user mode.

`get`
Lists one or more of the `smpatch` configuration parameter values. See "Configuring Your Update Management Environment."

To see values for all parameters, run the `smpatch get` command with no arguments. The output shows an entry for all configuration parameters. Each entry appears on a line by itself. Each entry includes three fields: the parameter name, the value you have assigned it, and its default value. The fields are separated by one or more tab characters.

The following values have special meaning: `-` means that no value is set, "" means that the value is the null string, `\-` means that the value is `-`, and `\""` means that the value is "" (two double quotes).

In addition to these special values, these special characters might appear in the output: `\t` for a tab, `\n` for a newline, and `\\` for a backslash.

To see values for particular parameters, run the `smpatch get` command with one or more parameter names. The output lists one parameter value per line in the order in which the parameter names are specified on the command line.

`messages`
Displays messages from the message–of–the–day (MOTD) file. This file stores messages, some of which can be urgent, from Sun. By default, when there is a new message in this file, the message shown below is displayed in `stderr`:

```
You have new messages. To retrieve: smpatch messages [-a]
```

The preceding message appears at the top of any `smpatch analyze/download/update` reports. The `-a` option to the `messages` subcommand displays all messages. Without the `-a` option, only new (that is, as yet unread) messages are displayed.

The receiving of messages from the software update client is controlled by the property:

```
patchpro.report.motd.messages
```

When this property is enabled, the default condition, the client receives messages from Sun. To disable this property, enter:

```
# smpatch set patchpro.report.motd.messages=false
```

`order`
Sorts a list of updates into an order that can be used to apply updates.

The list of updates is written to standard output, so you can redirect standard output to a file to create an update list.

Use the `-i` or `-x idlist=` option to specify the updates to order. Note that all of the updates you specify must exist in the download directory.

`remove`
Removes a single update from a single system.

Use the `-i` option to specify which update to remove. Do not use the `-x idlist=` option. Optionally, use the `-n` option to specify the name of a system. Do not use the `-x mlist=` option. By default, the update is removed from the local system.

If the update that you want to remove is required by one or more of the updates that have already been applied to the system, the update is not removed.

When you use the `-b` *BE_name* option, the `remove` subcommand removes a specified update from the BE *BE_name*, rather than from the currently running operating system.

`set`
Sets the values of one or more configuration parameters. Nothing is written to standard output or standard error when you set parameters, even if a parameter value you set is invalid. This command does not validate the values you set.

`unset`
: Resets one or more configuration parameters to the default values. You must specify at least one configuration parameter.

`update`
: Updates a single local or remote system by applying appropriate updates. This subcommand analyzes the system, then downloads the appropriate updates from the Sun update server to your system. After the availability of the updates has been confirmed, the updates are applied based on the update policy.

  By default, standard updates and those that have `rebootafter` or `reconfigafter` properties are applied.

  If an update does not meet the policy for applying updates, the update is not applied. Instead, the ID of the update is written to a file in the download directory. After the update ID is written to the file, `smpatch` continues to apply the other updates. Later, you can use `patchadd` to manually apply any updates listed in this file. The updates listed in the file are still in the download directory.

  When you use the `-b` *BE_name* option, the `update` subcommand applies an update to the BE *BE_name*, rather than to the currently running operating system. Upon successful installation of the update, `smpatch` activates *BE_name* (see `luactivate(1M)`) and informs you that you can, at that point, boot from *BE_name*.

  Installation instructions for updates that require special handling are included in the README file for each update.

  The `smpatch update` command depends on network services that are not available while the system is in single-user mode.

Using Local Mode or Remote Mode

Starting with Solaris 9, the `smpatch` command is available in two modes: local mode and remote mode. *Local mode* can be run only on the local system and can be run by users who have the appropriate authorizations. This mode can be run while the system is in single-user mode. *Remote mode* can be used to perform tasks on remote systems and can be run by users or roles that have the appropriate authorizations.

By default, local mode is run. In local mode, the Solaris WBEM services are not used, and none of the authentication options or those options that refer to remote systems are available. The command in local mode runs faster than in remote mode.

If the Solaris WBEM services are running and you specify any of the remote or authentication options, the command in remote mode is used.

On Solaris 8 systems, the `smpatch` command only supports local mode operations.

Specifying the Source of Updates

Your system must specify the source of updates to use. By default, you obtain updates from the Sun update server. However, you can also obtain updates from an update server on your intranet or from a local collection of updates on your system.

You must specify the URL that points to the collection of updates. By default, the Sun update server is the source of updates. The URL is:

```
https://getupdates.sun.com/solaris/
```

The URL must point to an update server or to a collection of updates that is available to the local system. The value of this URL cannot be null.

Configuring Your Update Management Environment

You can use the `smpatch set` command to configure the update management environment for your system. Use these parameters:

`patchpro.patchset`
: Name of the update set to use. The default name is `current`.

`patchpro.download.directory`
: Path of the directory where downloaded updates are stored and from which updates are applied. The default location is `/var/sadm/spool`.

`patchpro.backout.directory`
: Path of the directory where update backout data is saved. When an update is removed, the data is retrieved from this directory as well. By default, backout data is saved in the package directories.

`patchpro.patch.source`
: URL that points to the collection of updates. The default URL is that of the Sun update server, `https://getupdates.sun.com/solaris/`.

`patchpro.proxy.host`
: Host name of your web proxy. By default, no web proxy is specified, and a direct connection to the Internet is assumed.

`patchpro.proxy.port`
: Port number used by your web proxy. By default, no web proxy is specified, and a direct connection to the Internet is assumed. The default port is `8080`.

`patchpro.proxy.user`
: Your user name used by your web proxy for authentication.

`patchpro.proxy.passwd`
: Password used by your web proxy for authentication.

`patchpro.install.types`
: Your policy for applying updates. The value is a list of zero or more colon-separated update properties that are permitted to be applied by an update operation (`smpatch update`).

  This policy *only* affects which updates are installed on your system. The policy does not affect which updates are downloaded.

By default, updates that have the `standard`, `rebootafter`, and `reconfigafter` properties can be applied. See "Setting an Update Policy."

Setting an Update Policy

Updates are classified as being standard or nonstandard. A *standard update* can be applied by `smpatch update`. Such an update is associated with the `standard` update property. `smpatch` applies standard updates immediately. These updates require no system restart. A *nonstandard update* has one of the following characteristics:

- An update that is associated with the `rebootafter`, `rebootimmediate`, `reconfigafter`, `reconfigimmediate`, or `singleuser` properties. Such a nonstandard update can be applied automatically during a system shutdown if permitted by the policy.
- An update that is associated with the `interactive` property. Such an update cannot be applied by using automated installation mechanisms like `smpatch update`. When you attempt to apply one of these updates using `smpatch update`, the update will be downloaded but not installed. You must examine the update's README file and perform whatever manual steps it describes. Typically, you are instructed to apply the update manually using `patchadd` or `smpatch add`.

Use `smpatch set` to specify the types of updates that `smpatch` can additionally apply during an update operation. Such updates might include those that require a reboot and those that must be applied while the system is in single-user mode.

This policy only affects which updates are installed on your system. The policy does not affect which updates are downloaded.

Specify the types of updates that can be applied by using the following command:

```
# smpatch set patchpro.install.types=update-property-list
```

update-property-list is a colon-separated list of one or more of the following update properties:

`interactive`
An update that cannot be applied by running the usual update management tools (`pprosvc`, `smpatch`, or `patchadd`). Before this update is applied, the user must perform special actions. Such actions might include checking the serial number of a disk drive, stopping a critical daemon, or reading the update's README file.

`rebootafter`
The effects of this update are not visible until after the system is rebooted.

`rebootimmediate`
When this update is applied, the system becomes unstable until the system is rebooted. An unstable system is one in which the behavior is unpredictable and data might be lost.

`reconfigafter`
The effects of this update are not visible until after a reconfiguration reboot (`boot -r`). See the `boot(1M)` man page.

`reconfigimmediate`
When this update is applied, the system becomes unstable until the system gets a reconfiguration reboot (`boot -r`). An unstable system is one in which the behavior is unpredictable and data might be lost.

`singleuser`
Do not apply this update while the system is in multiuser mode. You must apply this update on a quiet system with no network traffic and with extremely restricted I/O activity.

`standard`
This update can be applied while the system is in multiuser mode. The effects of the update are visible as soon as it is applied unless the application being updated is running while the update is applied. In this case, the effects of the update are visible after the affected application is restarted.

Options The `smpatch` command supports two kinds of options: authentication options and subcommand options.

Authentication Options The `smpatch` authentication options, *auth-opts*, apply to all of the subcommands.

If no authentication options are specified, certain defaults are assumed and the user might be prompted for additional information, such as a password for authentication purposes.

These authentication options are only available if the Solaris Management Console and the Solaris WBEM services are available on the local system. If the WBEM services are not running on the local system, `smpatch` performs update operations on the local system only. You can also "force" the use of the local-mode `smpatch` command by using the `-L` option.

The single letter options can also be specified by their equivalent option words preceded by two hyphens. For example, you can specify either `-l` or `--rolepassword`.

The following authentication options are supported:

`-H` | `--hostname` *host-name*:*port*
Specifies the host and port to which you want to connect. If you do not specify a port, the system connects to the default port, `898`. If you do not specify a host (*host-name*:*port*), the Solaris Management Console connects to the local host on port `898`. You might still have to choose a toolbox to load into the console. To override this behavior, use the `smc -B` command, or set your console preferences to load a home toolbox by default.

`-L`
Forces the `smpatch` command to use local mode, which does not rely on Solaris WBEM services. On Solaris 8 systems, this option does not do anything.

This option is mutually exclusive with the other authentication options.

`-l` | `--rolepassword` *role-password*
Specifies the password for *role-name*. If you specify *role-name* but do not specify *role-password*, you are prompted to supply *role-password*. Because passwords specified on the command line can be seen by any user on the system, this option is considered to be insecure.

`-p` | `--password` *password*
Specifies the password for *user-name*. If you do not specify a password, you are prompted to supply one. Because passwords specified on the command line can be seen by any user on the system, this option is considered to be insecure.

`-r` | `--rolename` *role-name*
Specifies a role name for authentication. If this option is not specified, no role is assumed.

`-u` | `--username` *user-name*
Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

Subcommand Options

The following options pertain to the `smpatch` subcommands:

`-b` *BE_name*
Specifies the name of BE that is to be created and updated (for the `add` and `update` subcommands), or to be removed, for the `remove` subcommand.

`-d` *update-dir*
Specifies an alternate download directory in which updates are downloaded and from which they are applied.

The default download directory is `/var/sadm/spool`.

The directory must be writable by `root` and be publicly readable.

update-dir uses one of the following forms:

- For remote mode, specify *host-name*:*/update-dir*, where */update-dir* is a fully qualified, shared directory.
- For local mode, specify */update-dir*, which is a fully qualified, shared directory.

This option is supported by the `add`, `download`, `order`, and `update` subcommands.

`-f`
Forces the download of the exact update revision specified by `-i` *update-id*. If `-i` *update-id* does not specify a revision, the highest revision of the update is downloaded.

This option is supported by the `download` subcommand.

`-h`
Displays information about the command-line options for the specified subcommand. This option is mutually exclusive with all other options.

`-i` *update-id*
Specifies the ID of an update.

You can specify more than one update ID by using the `-i` option for each update. You can also use the `-x idlist=` option to point to a list of update IDs.

When using the `remove` subcommand, you can specify exactly one update ID.

This option is supported by the `add`, `analyze`, `download`, `order`, `remove`, and `update` subcommands.

`-n` *system-name*
Specifies the name of the system on which to manage updates.

When using the `add` subcommand, you can specify more than one system by using the `-n` option for each system. When using the `analyze`, `download`, `remove`, and `update` subcommands, you can only specify a single system.

To specify more than one system for the `smpatch add` command, use the `-x mlist=` option. This option enables you to specify a list of systems instead of using the `-n` option to specify each system. The `-n` option and the `-x mlist=` option are mutually exclusive.

If you do not specify this option, the system is assumed to be the one specified by the `-H` option.

This option is supported only if the Solaris Management Console and the Solaris WBEM services are running on the local system and any system that is specified by this option.

This option is supported by the `add`, `analyze`, `download`, `get`, `order`, `remove`, `set`, `unset`, and `update` subcommands.

`-t`
Downloads the `README` file associated with the update specified by `-i` *update-id*. If `-i` *update-id* does not specify a revision, the `README` file for the highest revision of the update is downloaded.

This option is supported by the `download` subcommand.

`-V`
Displays version information for an `smpatch` subcommand.

`-x idlist=`*update-list-file*
Specifies the name of a file, *update-list-file*, that contains a list of updates to download or apply.

Each update ID in the file must be terminated by a newline character. The file name you specify must be a full path name.

You can use the `-i` option to specify a list of update IDs.

This option is supported by the `add`, `analyze`, `download`, `order`, and `update` subcommands.

`-x mlist=`*system-list-file*
Specifies the name of a file, *system-list-file*, that contains a list of systems on which to manage updates.

Each system name must be terminated by a newline character. The file name you specify must be a full path name.

You can use the `-n` option to specify a list of systems instead of using the `-x mlist=` option. The `-n` option and the `-x mlist=` option are mutually exclusive.

This option is supported only if the Solaris Management Console and the Solaris WBEM services are running on the local system and any system that is specified in *system-list-file*.

This option is supported by the `add` subcommand.

Examples

EXAMPLE 1 Analyzing Your System to Obtain the List of Appropriate Updates for the Local System

```
# smpatch analyze
```

Shows how to analyze your system to obtain the list of appropriate updates. After the analysis, you can download and apply the updates to your system.

EXAMPLE 2 Analyzing Your System to Obtain the List of Appropriate Updates for Another System

```
# smpatch analyze -n lab1
```

Shows how to analyze a different system, `lab1`, to obtain the list of appropriate updates. After the analysis, you can download and apply the updates to that system.

EXAMPLE 3 Applying Updates to Multiple Systems

```
# smpatch add -i 102893-01 -i 106895-09 -i 106527-05 \
-d fileserver:/files/updates/s10 -n lab1 -n lab2
```

Applies updates `102893-01`, `106895-09`, and `106527-05` to the systems `lab1` and `lab2`. The updates are located in the `/files/updates/s10` directory on the system named `fileserver`.

EXAMPLE 4 Applying Updates by Using an Update List File

```
# smpatch add -x idlist=/tmp/update/update_file \
-d /net/fileserver/export/updatespool/Solaris10 -n lab1 -n lab2
```

Applies the updates specified in the file `/tmp/update/update_file` to the systems `lab1` and `lab2`. The updates are located in the NFS-mounted directory named `/net/fileserver/export/updatespool/Solaris10`.

EXAMPLE 5 Applying Updates by Using an Update List File and a System List File

```
# smpatch add -x idlist=/tmp/update/update_file \
-x mlist=/tmp/update/sys_file
```

Applies the updates listed in the file `/tmp/update/update_file` to the systems listed in the file `/tmp/update/sys_file`. The updates are located in the default `/var/sadm/spool` directory on the local system.

EXAMPLE 6 Analyzing a System and Downloading Updates From the Sun Update Server

```
# smpatch download -n lab1
```

Analyzes the `lab1` system and downloads the appropriate updates from the Sun update server to the download directory.

EXAMPLE 7 Downloading Updates From the Sun Update Server

The command below downloads the `102893-01` and `106895-09` updates from the Sun update server to the `/files/updates/s10` directory.

```
# smpatch download -i 102893-01 -i 106895-09 -d /files/updates/s10
```

EXAMPLE 8 Downloading Specific Update Revisions From the Sun Update Server

The command below downloads the `102893-01` and `106895-02` updates from the Sun update server. The specific revisions are downloaded, not the highest available revision.

```
# smpatch download -f -i 102893-01 -i 106895-02
```

EXAMPLE 9 Downloading the Highest Available Update Revisions From the Sun Update Server

The command below downloads the `102893-01` and `106895-09` updates, which are the highest available revisions, from the Sun update server.

```
# smpatch download -f -i 102893 -i 106895
```

EXAMPLE 10 Downloading Update README Files From the Sun Update Server

The command below downloads the `README` files for updates `102893-01` and `106895-02`. Because update `102893` was specifi ed without a revision number, the `README` file for the highest available update revision, `102893-01`, is downloaded from the Sun update server.

```
# smpatch download -t -i 102893 -i 106895-02
```

EXAMPLE 11 Listing All Configuration Parameter Values

```
# smpatch get -p password
Loading Tool: com.sun.admin.patchmgr.cli.PatchMgrCli from mars
Login to mars as user root was successful.
Download of com.sun.admin.patchmgr.cli.PatchMgrCli from mars
was successful.
```

EXAMPLE 11 Listing All Configuration Parameter Values *(Continued)*

```
On machine mars:

patchpro.backout.directory  -      ""
patchpro.download.directory -      /var/sadm/spool
patchpro.install.types      -      rebootafter:reconfigafter:standard
patchpro.patch.source       -      https://getupdates.sun.com/solaris/
patchpro.patchset           -      current
patchpro.proxy.host         -      ""
patchpro.proxy.passwd       ****   ****
patchpro.proxy.port         -      8080
patchpro.proxy.user         -      ""
patchpro.sun.passwd         ****   ****
patchpro.sun.user           -      ""
```

Lists the configuration settings for the system.

EXAMPLE 12 Listing One or More Configuration Parameter Values

```
# smpatch get -L patchpro.patch.source patchpro.download.directory
https://getupdates.sun.com/solaris/
/var/sadm/spool
```

Uses `smpatch` in local mode to list the values of the `patchpro.patch.source` and the `patchpro.download.directory` parameters.

EXAMPLE 13 Reordering a List of Updates

```
# smpatch order -x idlist=/tmp/plist
```

Reorders the update list called `/tmp/plist` in an order that is suitable for applying the updates.

EXAMPLE 14 Removing an Update

```
# smpatch remove -i 102893-01
```

Removes update `102893-01`.

EXAMPLE 15 Specifying the Update Policy

The following command specifies the update policy.

```
# smpatch set \
patchpro.install.types=standard:singleuser:reconfigafter:rebootafter
```

Specifies the update policy for your system. The following types of updates are allowed to be applied to your system:

- Standard updates
- Updates that must be applied in single-user mode

EXAMPLE 15 Specifying the Update Policy *(Continued)*

- Updates that require that the system undergo a reconfiguration reboot after they have been applied
- Updates that require that the system undergo a reboot after they have been applied

EXAMPLE 16 Changing the Download Directory Location

```
# smpatch set patchpro.download.directory=/export/home/updates
```

EXAMPLE 17 Specifying a Local Web Proxy

```
# smpatch set patchpro.proxy.host=webaccess.corp.net.com \
patchpro.proxy.port=8080
```

Specifies the host name, `webaccess.corp.net.com`, and port, `8080`, of the local web proxy.

EXAMPLE 18 Resetting a Configuration Parameter Value

```
# smpatch unset patchpro.patch.source
```

Resets the value of the `patchpro.patch.source` parameter to its default value, which is the URL that points to the Sun update server.

EXAMPLE 19 Updating Your System

```
# smpatch update -L
```

Analyzes your local system, determines the appropriate updates, downloads those updates to the download directory, and applies those updates.

EXAMPLE 20 Adding an Update to a BE

The following command adds a specific update to the BE `altboot`.

```
# smpatch add -b altboot 111111-01
```

Following successful completion of this command, you can then boot from `altboot`.

EXAMPLE 21 Updating a BE

The following command performs an update on the BE `altboot`.

```
# smpatch update -b altboot
```

This command performs all of the usual analysis and dependency checking that occurs with any `update` command. Following successful completion of this command, you can then boot from `altboot`.

EXAMPLE 22 Obtaining `smpatch` Version Number

The following command returns the version number for an `smpatch` subcommand.

EXAMPLE 22 Obtaining `smpatch` Version Number *(Continued)*

```
# smpatch update -V
1.0.9
```

Environment Variables

See `environ`(5) for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smpatch` command. The default value of this variable is `/usr/java`. See the `smc`(1M) man page.

Exit Status

The following exit values are returned:

`0`
Successful completion.

`1`
Invalid command syntax. A usage message displays.

`2`
An error occurred while executing the command. An error message displays.

Attributes

See the `attributes(5)` man page for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWswmt
Interface Stability	Evolving

See Also

`boot`(1M), `luactivate`(1M), `patchadd`(1M), `patchrm`(1M), `smc`(1M), `attributes`(5), `environ`(5), `live_upgrade`(5)

Sun Update Manager 1.0 Administration Guide

Name smprofile – manage profiles in the prof_attr and exec_attr databases

Synopsis

```
/usr/sadm/bin/smprofile subcommand [ auth_args] - –
    [subcommand_args]
```

Description The `smprofile` command manages one or more profiles in the `prof_attr(4)` or `exec_attr(4)` databases in the local `/etc` files name service or a NIS or NIS+ name service.

subcommands `smprofile` *subcommand*s are:

`add` Adds a new profile (right) to the `prof_attr(4)` database. To add a profile, the administrator must have the `solaris.profmgr.write` authorization.

`delete` Deletes a profile from the `prof_attr(4)` database, deletes all associated entries from the `exec_attr(4)` database, and deletes the assigned profile from the `user_attr(4)` database. To delete a profile, the administrator must have the `solaris.profmgr.execattr.write` and `solaris.profmgr.write` authorization.

`list` Lists one or more profiles from the `prof_attr`(4) or `exec_attr`(4) databases. To list a profile, the administrator must have the `solaris.profmgr.read` authorization.

`modify` Modifies a profile in the `prof_attr`(4) database. To modify a profile, the administrator must have the `solaris.profmgr.write` authorization.

Options The `smprofile` authentication arguments, *auth_args*, are derived from the `smc(1M)` arg set and are the same regardless of which subcommand you use. The `smprofile` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- –` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- –domain` with the *domain* argument.

`-D | - –domain` *13;domain* Specifies the default domain that you want to manage. The syntax of *domain* is *type:/host_name/domain_name*, where *type* is `nis`, `nisplus`, `dns`, `ldap`, or `file`; *host_name* is the name of the machine that serves the domain; and *domain_name* is the name of the domain you want to manage. (*Note:* Do *not* use `nis+` for `nisplus`.)

	If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.
`-H | - –hostname` *13;host_name:port*	Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc(1M)` `-B` option, or set your console preferences to load a "home toolbox" by default.
`-l | - –rolepassword` *13;role_password*	Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-p | - –password` *13;password*	Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-r | - –rolename` *13;role_name*	Specifies a role name for authentication. If you do not specify this option, no role is assumed.
`-u | - –username` *13;user_name*	Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.
`- –`	This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- –` option.

subcommand_args *Note:* Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

To add privileges to or modify privileges in a profile entry, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.

- For subcommand `add`:

  `-a` *addauth1* `-a` *addauth2* . . . (Optional) Specifies the authorization name(s) to add to the new profile. The administrator must have the `solaris.profmgr.write` authorization and must have the corresponding "grant" authorization. A "grant" authorization is one in which the lowest component of the authorization name is replaced by the word `grant`. For example, to grant some profile the `solaris.role.write` authorization, the administrator needs that authorization and also the `solaris.role.grant` authorization. For more information on granting authorizations, see `auth_attr(4)`.

  `-d` *description* Specifies the description of the new profile.

  `-h` (Optional) Displays the command's usage statement.

  `-m` *html_help* Specifies the HTML help file name for the new profile. The help file name must be put in the `/usr/lib/help/profiles/locale/C` directory.

  `-n` *name* Specifies the name of the new profile.

  `-p` *addprof1* `-p` *addprof2* . . . (Optional) Specifies the supplementary profile name(s) to add to the new profile.

  `-I` *inherited_privs* Specifies the inherited privilege name(s) to add to the new `prof_attr(4)` entry.

  To add privileges to or modify privileges in a profile entry, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.

- For subcommand `delete`:

  `-h` (Optional) Displays the command's usage statement.

  `-n` *name* Specifies the name of the profile you want to delete.

- For subcommand `list`:

  `-h` (Optional) Displays the command's usage statement.

  `-l` (Optional) Displays the detailed output for each profile in a block of *key*:*value* pairs, followed by a blank line that delimits

each profile block. Each *key*:*value* pair is displayed on a separate line. All the attributes associated with a profile from the `prof_attr` and `exec_attr` databases are displayed. If you do not specify this option, only the specified profile name(s) and associated profile description(s) are displayed.

`-n` *name1* `-n` *name2* . . . (Optional) Specifies the profile(s) that you want to display. If you do not specify a profile name, all profiles are displayed.

- For subcommand `modify`:

  `-a` *addauth1* `-a` *addauth2* . . . (Optional) Specifies the authorization name(s) to add to the profile. The administrator must currently have been granted each of the specified authorizations and must have the ability to grant each of those authorizations to other users or roles. For more information on granting authorizations, see `auth_attr(4)`.

  `-d` *description* (Optional) Specifies the new description of the profile.

  `-h` (Optional) Displays the command's usage statement.

  `-m` *html_help* (Optional) Specifies the new HTML help file name of the profile. If you change this name, you must accordingly rename the help file name entered in the `/usr/lib/help/profiles/locale/C` directory.

  `-n` *name* Specifies the name of the profile you want to modify.

  `-p` *addprof1* `-p` *addprof2* . . . (Optional) Specifies the supplementary profile name(s) to add to the profile. The administrator must have the `solaris.profmgr.assign` authorization to add any profile and the `solaris.profmgr.delegate` authorization to add any profile that has been assigned to the authenticated user.

  `-q` *delprof1* `-q` *delprof2* . . . (Optional) Specifies the supplementary profile name(s) to delete from the profile. The administrator must have the `solaris.profmgr.assign` authorization to delete any profile and the `solaris.profmgr.delegate` authorization to delete any profile that has been assigned to the authenticated user.

  `-r` *delauth1* `-r` *delauth2* . . . (Optional) Specifies the authorization name(s) to delete from the profile. The administrator must have the `solaris.profmgr.write` authorization and must have the corresponding "grant" authorization. For more information about "grant" authorizations, see the `-a` option description for the `add` subcommand above.

-I *inherited_privs*

Specifies the inherited privilege name(s) to modify in the prof_attr(4) entry.

To add privileges to or modify privileges in a profile entry, the administrator must have the solaris.admin.privilege.write authorization. See privileges(5).

Examples

EXAMPLE 1 Creating a new profile

The following creates a new User Manager profile on the local file system. The new profile description is Manage users and groups, and the authorizations assigned are solaris.admin.usermgr.write and solaris.admin.usermgr.read. The supplementary profile assigned is Operator. The help file name is RtUserMgmt.html.

```
./smprofile add -H myhost -p mypasswd -u root -- -n "User Manager" \
   -d "Manage users and groups" -a solaris.admin.usermgr.write \
   -a solaris.admin.usermgr.read -p Operator -m RtUserMgmt.html
```

EXAMPLE 2 Deleting a profile

The following deletes the User Manager profile from the local file system:

```
./smprofile delete -H myhost -p mypasswd -u root -- -n "User Manager"
```

EXAMPLE 3 Listing all profiles

The following lists all profiles and their associated profile descriptions on the local file system.

```
./smprofile list -H myhost -p mypasswd -u root --
```

EXAMPLE 4 Modifying a profile

The following modifies the User Manager profile on the local file system. The new profile description is Manage world, the new authorization assignment is solaris.admin.usermgr.* authorizations, and the new supplementary profile assignment is All. (The -a option argument must be enclosed in double quotes when the wildcard character (*) is used.)

```
./smprofile modify -H myhost -p mypasswd -u root -- -n "User Manager" \
   -d "Manage world" -a "solaris.admin.usermgr.*" -p All
```

Environment Variables

See environ(5) for a description of the JAVA_HOME environment variable, which affects the execution of the smprofile command. If this environment variable is not specified, the /usr/java location is used. See smc(1M).

Exit Status

The following exit values are returned:

0 Successful completion.

1 Invalid command syntax. A usage message displays.

2 An error occurred while executing the command. An error message displays.

Files The following files are used by the `smprofile` command:

`/etc/security/exec_attr`	Rights profiles database. See `exec_attr(4)`.
`/etc/security/prof_attr`	Profile description database. See `prof_attr(4)`.
`/etc/user_attr`	Extended user attribute database. See `user_attr(4)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga
Interface Stability	Evolving

See Also `smc(1M)`, `auth_attr(4)`, `exec_attr(4)`, `prof_attr(4)`, `user_attr(4)`, `attributes(5)`, `environ(5)`

Name smreg – register objects for the Sun Web Console

Synopsis `/usr/sbin/smreg [-h | --help] subcommand options`

Description The `smreg` utility is a Command Line Interface (CLI) -based registration mechanism to manage the information in the console's registration databases. `smreg` adds management web applications, libraries, and configuration information to registration databases. It also validates an application's deployment descriptor to make sure it has certain required tags that enable the application to run in the console.

`smreg` has four subcommands: `add`, `remove`, `check`, and `list`.

Subcommands The following subcommands are supported:

add
Registers objects of a specified type. If the object is already registered, any new registration will replace the previous registration. The application registration descriptor `app.xml` is verified to ensure that meets some initial requirements.

The format of the `add` subcommand is:

add `-a` [`-x` *context*] *path*

add `-d` *path*

add `-l` [`-L`] [`-n` *lib_name*] `-s` ALL | *app_name path*

add `-m -b` *behavior* [`-s` *service*] `-o` *name=value ... module*

add `-p` [`-c` | `-e`] *name=value ...*

add `-h`

remove
Unregisters named objects.

The format of the remove subcommand is:

remove `-a` *app_name* | *context*

remove `-n` *app_name* | `-d` *path*

remove `-l -s` ALL | *app_name lib_name*

remove `-m` [`-s` *service*] *module*

remove `-p` [`-c` | `-e`] *name ...*

remove `-h`

`check`
Check an application.

The `check` subcommand parses the application's `web.xml` file to make sure that it has the *filter* and *filter-mapping* tags. The console requires these tags to run an application.

The required tags are as follows:

```
<filter>
  <filter-name>SessionManagerFilter</filter-name>
  <filter-class>
    com.sun.management.services.session.AppSessionManagerFilter
  </filter-class>
  <init-param>
    <param-name>ignore-paths</param-name>
    <param-value>
        images/*
    </param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>SessionManagerFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Elements in the `web.xml` file must appear in a fixed order. For example, the *filter* tag must appear in the `web.xml` file before the *filter-mapping tag*. For more information, see the *Java Servlet Specifications 2.3.*

The `smreg check` subcommand parses the application's `web.xml` file, checking for the existence of these tags and for the correct tag content. That is, the embedded *filter-name* tag content must be `SessionManagerFilter`, and the *filter-class* tag content must be:

```
com.sun.management.services.session.AppSessionManagerFilter
```

If the tag content is not correct, the application will not be registered.

If the tags are not included in the application's `web.xml` file, the script prints to standard output a corrected version the `web.xml` file, with the tags embedded in the correct location. Because multiple *filter* tags are allowed in a file, `smreg` also includes the new tags as the first in a series.

The format for the `check` subcommand is:

`check` [`-d`] *path*

`check -h`

The `-d` is used strictly to maintain interface compatibility with 1.0-based application packages, and is otherwise ignored.

list

Prints list of registered objects.

The format of the list subcommand is:

`list` [`-a` | `-l` | `-m` | `-p`]

If no options are specified, then all registered objects are listed.

Options The following options are supported:

`-a` | `--application`
Object type is an application.

See *app_name* under ARGUMENTS for more information on the name by which an application is registered.

`-b` *behavior* | `--behavior` *behavior*
A flag value that controls behavior as authentication proceeds down the login module stack. See *behavior* under ARGUMENTS for more information on values allowed.

`-c` *name=value* ... | `--configuration` *name=value* ...
When used with `-p`, this option specifies that the property arguments are server configuration properties. The *name=value* pairs are written to a datastore, to be read during the next server startup. Any property name can be created, but only those recognized by the server will have any effect.

`-d` *path* | `--directory` *path*
This option is deprecated and is preserved only for compatibility with existing 1.0-based applications. It will be removed in a future release.

When used with the "add" subcommand, it has the same effect as "`-a`".

When used with the "remove" subcommand, *path* is the path to the original installation location of the application. This path will be mapped to the registered *app_name* so that the application can be unregistered using "`smreg remove -a` *app_name*". When used with the "check" subcommand, `-d` is ignored. The `-d` option is preserved strictly to maintain 1.0 interface compatibility.

`-e` *name=value* ... | `--environment` *name=value* ...
When used with `-p`, this option specifies that the property arguments are server environment properties. The *name=value* pairs are placed into the server's environment during the next server startup. Any environment name can be created, and is available for use by any application.

`-h` | `-?` | `--help`
Displays command and subcommand usage information.

`-l` | `--library`
Object type is a library JAR file.

Library JAR files that are not installed in the same location as the application, or are not registered at the same time as the application (for example, localization JAR files), can be registered separately by using this option. See *lib_name* under ARGUMENTS for more information on the name by which a library is registered.

A library can be registered with global scope so that it can be used by all applications, by specifying the `-s ALL` option. A library can be registered so that it may only be used by a specific application, by specifying the `-s` *app_name* option. If a library must be shared by more than one application but not globally, then a seperate registration is required for each instance in which the library is to be shared.

`-L` | `--link`
Specifies to register the library JAR as a symbolic link rather than as a file copy. This option is ignored on operating systems which do not support symbolic links.

`-m` | `--module`
Object type is a login module.

`-n` [*app_name* | *lib_name*] | `--name` [*app_name* | *lib_name*]
When used with the "add `-l`" subcommand, specifies the name by which an application or library is known to the registration service.

When used with the "remove" subcommand, specifies the name by which an application is known to the registry service. "`remove -n`" is deprecated and is preserved only for compatibility with existing 1.0-based applications. It will be removed in a future release.

See *app_name* under ARGUMENTS for more information on the name by which an application is registered.

See *lib_name* under ARGUMENTS for more information on the name by which a library is registered.

`-o` *name=value* ... | `--option` *name=value* ...
Options specific to the login module implementation. The options are specified as *name=value* pairs, each preceded by the `-o` option. Values containing more than one word must be enclosed in double quotes (").

`-p` | `--properties`
Object type is properties. This option is specified for use with either `-c` (for configuration properties) or `-e` (for environment properties). If neither `-c` nor `-e` are specified, then `-c` is assumed.

See the PROPERTIES section for information on specific configuration properties that are useful to a system administrator.

`-s` [ALL | *app_name*] | `--scope` [ALL | *app_name*]
When used with `-l`, specifies the sharing scope for the library being registered. A scope of `ALL` makes the library available to all applications. A scope of *app_name* makes the library available only to the application already registered as *app_name*. See *app_name* under ARGUMENTS for more information on the name by which an application is registered.

`-s` *service* | `--scope` *service*
When used with `-m`, specifies the login service scope for the module. If not specified, the default is `ConsoleLogin`, which is the web console's browser login service.

`-V` | `--version`
Display console version information.

`-x` *context* | `--context` *context*
The deployment context path for an application. This option is to be used to register applications built with the SDK version 2.1 or greater. If not provided and the application is unpacked, the context is the parent directory of the application's WEB-INF directory. This option is ignored when registering applications built with an SDK version prior to 2.1.

Arguments The subcommand arguments are:

app_name — The unique name of the registered application in the format of *pluginName_version*, where *pluginName* and *version* are tags extracted from the application's registration descriptor, `app.xml`. All subsequent references to the registered application must then be made by using this registered name.

path — When used with the "`add -a`", "`add -d`", or "`remove -d`" subcommands, the full directory path where the application has been installed.

When used with the `check` subcommand, if it is a directory then it is the full directory path where the application has been installed. If it is a file, then it is the path to a `web.xml`-compliant file.

When used with the "`add -l`" subcommand, the full path where the library JAR file is installed.

behavior — Specifies the authentication behavior. The possible values are: `required`, `requisite`, `sufficient`, or `optional`.

lib_name — The name by which a library JAR file is known to the registration service. By default, libraries will be registered using the basename of the path to the library. This default value can be overridden by using the `-n` *lib_name* option to register the object by using a globally unique name.

module — The fully-qualified class name of the module. For example: `com.sun.management.services.authentication.MyLoginModule`

Properties While the list of configuration properties is unlimitted, certain properties that may be useful to a system administrator are mentioned here:

session.timeout.value
This is the time interval of no user activity after which the user will be prompted to log in again to continue. The default session timeout is 15 minutes. Setting the value to -1 means there is no timeout. To set the session timeout to 5 minutes, run the following command:

```
# smreg add -p -c session.timeout.value=5
```

debug.trace.path
This is the directory where log files are created. The default is `/var/log/webconsole`. To set the directory to `/tmp/logs`, run the following command:

```
# smreg add -p -c debug.trace.path=/tmp/logs
```

java.home
This is the path to the Java Development Kit (JDK) that will be used to run the web server. The default is `/usr/j2se` on Solaris and `/usr/java/j2sdk1.4.2` on Linux. To set the path to `/usr/jdk142`, run the following command:

```
# smreg add -p -c java.home=/usr/jdk142
```

java.options
This contains the options for configuring the Java Virtual machine (JVM). The defaults are "-server -XX:+BackgroundCompilation". To include setting the initial Java heap size to 64MB, run the following command:

```
# smreg add -p -c java.options="-Xms64 \\
-server -XX:+BackgroundCompilation"
```

Examples

EXAMPLE 1 Registering an unpacked Application

The following command registers an application which has been installed unpacked in `/opt/myCompany/myApp`:

```
# smreg add -a /opt/myCompany/myApp
```

EXAMPLE 2 Unregistering an Application

The following command unregisters an application previous registered with an *app_name* of `com.myCompany.myApp_1.2`:

```
# smreg remove -a com.myCompany.myApp_1.2
```

EXAMPLE 3 Registering a Library JAR File with Global Scope

The following command registers `myLibrary.jar`, installed in `/opt/myCompany/MyApp`, as `com_myCompany_myLibrary.jar` for use by all applications:

```
# smreg add -l -n com_myCompany_myLibrary.jar \\
      -s ALL /opt/myCompany/MyApp/myLibrary.jar
```

EXAMPLE 4 Registering a Library JAR file with a Specific Application Scope

The following command registers Utilities.jar, installed in `/opt/SomeCompany/lib`, with the application already registered as `com.myCompany.myApp_1.2`:

```
# smreg add -l -s com.myCompany.myApp_1.2 \
/opt/SomeCompany/lib/Utilities.jar
```

EXAMPLE 5 Unregister a Library JAR File from Global Scope

The following command unregisters `com_myCompany_myLibrary.jar` from global scope so that it is not available to applications:

```
# smreg remove -l -s ALL com_myCompany_myLibrary.jar
```

EXAMPLE 6 Unregister a Library JAR File from a Specific Application Scope

The following command unregisters `Utilities.jar` so that it is not available to the registered application `com.myCompany.myApp_1.2`:

```
# smreg remove -l -s com.myCompany.myApp_1.2 Utilities.jar
```

EXAMPLE 7 Checking that Application's Deployment Descriptor Meets Sun Web Console Guidelines

Either of the following commands to check to see if the deployment descriptor located at `/opt/myCompany/myApp/WEB-INF/lib/web.xml` meets the Sun Web Console guideline:

```
# smreg check /opt/myCompany/myApp
  # smreg check /opt/myCompany/myApp/WEB-INF/lib/web.xml
```

EXAMPLE 8 Registering an Environment Property

The following command registers the environment property name `GREETING` with value "Hello World" and the name `FOO` with the value "bar" so that they appear in the server's environment and are available to any application:

```
# smreg add -p -e GREETING="Hello World" FOO=bar
```

EXAMPLE 9 Unregistering an Environment Property

The following command unregisters the environment property names `GREETING` and `FOO` so that they no longer appear in the server's environment and are not available to any application:

```
# smreg remove -p -e GREETING FOO
```

EXAMPLE 10 Registering a Login Module

The following command registers the module class `com.sun.management.services.authentication.myLoginModule` with "requisite" behavior and a `commandPath` argument:

```
# smreg add -m -b requisite \\
      -o commandPath="/usr/lib/webconsole" \\
      com.sun.management.services.authentication.myLoginModule
```

EXAMPLE 11 Unregistering a Login Module

The following command unregisters the module class `com.sun.management.services.authentication.myLoginModule`:

```
# smreg remove -m com.sun.management.services.authentication.myLoginModule
```

EXAMPLE 12 Print a List of All Registered Applications

```
# smreg list -a
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` Usage error: missing or malformed arguments.

`2` An error occurred that prevents registration - either a console configuration problem, or badly-formatted `web.xml` or `app.xml`.

`3` Unable to determine OS for this machine.

`4` Detected OS for this machine is not supported.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmcon
Interface Stability	Obsolete

See Also

`smcwebserver(1M)`, `attributes(5)`

Name smrole – manage roles and users in role accounts

Synopsis

```
/usr/sadm/bin/smrole subcommand [ auth_args] - –
     [subcommand_args]
```

Description The `smrole` command manages roles and adds or deletes users in role accounts.

subcommands `smrole` *subcommand*s are:

`add` Adds a new role entry. To add an entry, the administrator must have the `solaris.role.write` authorization.

`delete` Deletes one or more roles. To delete an entry, the administrator must have the `solaris.role.write` authorization.

`list` Lists one or more roles. If you do not specify a role name, all roles are listed. To list an entry, the administrator must have the `solaris.admin.usermgr.read` authorization.

`modify` Adds or deletes users from a role account. To modify an entry, the administrator must have the `solaris.role.write` authorization.

Options The `smrole` authentication arguments, *auth_args*, are derived from the smc(1M) arg set and are the same regardless of which subcommand you use. The `smrole` command requires the Solaris Management Console to be initialized for the command to succeed (see smc(1M)). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- –` option.

auth_args The *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u` are described below. They are all optional. These options are a subset of the full complement of supported options described in smc(1M).

If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- –domain` with the *domain* argument.

`-D` | `- –domain` *13;domain* Specifies the default domain that you want to manage. The syntax of *domain* is *type*:/*host_name*/*domain_name*, where *type* is `nis`, `nisplus`, `dns`, `ldap`, or `file`; *host_name* is the name of the machine that serves the domain; and *domain_name* is the name of the domain you want to manage. (*Note:* Do *not* use `nis+` for `nisplus`.)

	If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.
`-H | - —hostname` *13;host_name:port*	Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc(1M) -B` option, or set your console preferences to load a "home toolbox" by default.
`-l | - —rolepassword` *13;role_password*	Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-p | - —password` *13;password*	Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-r | - —rolename` *13;role_name*	Specifies a role name for authentication. If you do not specify this option, no role is assumed.
`-u | - —username` *13;user_name*	Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.
`- —`	This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- —` option.

subcommand_args *Note:* Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

To add or change privileges, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.

- For subcommand `add`:

  `-a` *adduser1* `-a` *adduser2* . . .
  (Optional) Specifies the user name(s) to add to the new role. The administrator must have the `solaris.role.assign` authorization.

  `-c` *comment*
  (Optional) Includes a short description of the role. Consists of a string of up to 256 printable characters, excluding the colon (`:`).

  `-d` *dir*
  (Optional) Specifies the home directory of the new role, limited to 1024 characters.

  `-F` *full_name*
  (Optional) Specifies the full, descriptive name of the role. The *full_name* must be unique within a domain, and can contain alphanumeric characters and spaces. If you use spaces, you must enclose the *full_name* in double quotes.

  `-G` *group1* `-G` *group2* . . .
  (Optional) Specifies the new role's supplementary group membership in the system group database with the character string names of one or more existing groups. *Note:* You cannot assign a primary group to a role. A role's primary group is always `sysadmin` (group 14).

  `-h`
  (Optional) Displays the command's usage statement.

  `-n` *rolename*
  Specifies the name of the role you want to create.

  `-p` *addprof1* `-p` *addprof2* . . .
  (Optional) Specifies the profile(s) to add to the role. To assign a profile to a role, the administrator must have the `solaris.profmgr.assign` or `solaris.profmgr.delegate` authorization.

  `-P` *password*
  (Optional) Specifies the role's password. The *password* can contain up to eight characters. If you do not specify a password, the system prompts you for one. To set the password, the administrator must have the `solaris.admin.usermgr.pswd` authorization. *Note:* When you specify a password using the `-P` option, you type the password in plain text. Specifying a password using this method introduces a security gap while the command is running. However, if you do not specify a password (and the system prompts you for one), the echo is turned off when you type in the password.

`-s` *shell*	(Optional) Specifies the full pathname of the program used as the role's shell on login. Valid entries are `/bin/pfcsh` (C shell), `/bin/pfksh` (Korn shell), and `/bin/pfsh` (Bourne shell), the default.
`-u` *uid*	(Optional) Specifies the ID of the role you want to add. If you do not specify this option, the system assigns the next available unique ID greater than 100.
`-x` *autohome*=Y|N	(Optional) Sets the role's home directory. The home directory path in the password entry is set to `/home/`*login name*.
`-x` perm=*home_perm*	(Optional) Sets the permissions on the role's home directory. *perm* is interpreted as an octal number, and the default is `0775`.
`-x` serv=*homedir_server*	(Optional) If `-D` is `nis`, `nisplus`, or `ldap`, use this option to specify the name of the server where the user's home directory resides. Users created in a local scope must have their home directory server created on their local machines.
`-M` *limit_privs*	Specifies the privilege name(s) to add to the new `user_attr(4)` entry. The default is `all` for *limit* privilege. To add or change privileges, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.
`-D` *default_privs*	Specifies the default privilege name(s) to add to the new `user_attr(4)` entry.

The options to the `add` subcommand listed below are available only if a system is configured with Solaris Trusted Extensions. See "Using Options that Require Solaris Trusted Extensions," below.

`-x` clear=*clearanceval*	(Optional) Specifies the role's clearance. *clearanceval* can be a string value or a hex value. If this option is not specified, the default, `admin_high`, is in effect. To set the clearance, the administrator must have the `solaris.admin.usermgr.labels` authorization.
`-x` label=*labelval*	(Optional) Specifies the role's minimum label. *labelval* can be a string label or a

hex label. If this option is not specified, the default, `admin_low`, is in effect. To set the minimum label, the administrator must have the `solaris.admin.usermgr.labels` authorization.

`-x` labelview=HIDE|SHOW — (Optional) Specifies the second part of the `labelview` *key-value* pair. If `SHOW` is specified, `labelview=*showsl` will be recorded. If `HIDE` is specified, `labelview=*hidesl` will be recorded. The asterisk portion can be replaced by "`internal,`", "`external,`", or ""(null). If this option is not specified, the default, `SHOW`, is in effect.

`-x` view=INTERNAL|EXTERNAL|DEFAULT — (Optional) Specifies the label view type for the `labelview` in `user_attr`. If `INTERNAL` is specified, `labelview=internal` will be recorded; if `EXTERNAL` is specified, `labelview=external` will be recorded; if `DEFAULT` is specified, nothing will be recorded in `user_attr`. If this option is not specified, the default, `INTERNAL`, is in effect.

- For subcommand `delete`:

  `-h` — (Optional) Displays the command's usage statement.

  `-n` *rolename1* `-n` *rolename2* . . . — Specifies the name of the role(s) you want to delete.

- For subcommand `list`:

  `-h` — (Optional) Displays the command's usage statement.

  `-l` — (Optional) Displays the output for each user in a block of *key*:*value* pairs (for example, `user name:root`), followed by a blank line that delimits each user block. Each *key*:*value* pair is displayed on a separate line. The keys are: `autohome setup`, `comment`, `home directory`, `login shell`, `primary group`, `secondary groups`, `server`, `user ID (UID)`, and `user name`.

  `-n` *role1* `-n` *role2* . . . — (Optional) Specifies the role(s) that you want to list. If you do not specify a role name, all roles are listed.

- For subcommand `modify`:

`-a` *adduser1* `-a` *adduser2* . . .	(Optional) Specifies the user name(s) to add to the new role. The administrator must have the `solaris.role.assign` authorization, or must have the `solaris.role.delegate` authorization and be a member of the role being modified.
`-c` *comment*	(Optional) Includes a short description of the role. Consists of a string of up to 256 printable characters, excluding the colon (`:`).
`-d` *dir*	(Optional) Specifies the home directory of the new role, limited to 1024 characters.
`-F` *full_name*	(Optional) Specifies the full, descriptive name of the role. The *full_name* must be unique within a domain, and can contain alphanumeric characters and spaces. If you use spaces, you must enclose the *full_name* in double quotes.
`-G` *group1* `-G` *group2* . . .	(Optional) Specifies the new role's secondary group membership in the system group database with the character string names of one or more existing groups. *Note:* You cannot assign a primary group to a role. A role's primary group is always `sysadmin` (group 14).
`-h`	(Optional) Displays the command's usage statement.
`-n` *rolename*	Specifies the name of the role you want to modify.
`-N` *new_rolename*	(Optional) Specifies the new name of the role.
`-p` *addprof1* `-p` *addprof2* . . .	(Optional) Specifies the profile(s) to add to the role. To assign a profile to a role, the administrator must have the `solaris.profmgr.assign` or `solaris.profmgr.delegate` authorization.
`-P` *password*	(Optional) Specifies the role's password. The *password* can contain up to eight characters. To set the password, the administrator must have the `solaris.admin.usermgr.pswd` authorization. *Note:* When you specify a password, you type the password in plain text. Specifying a password using this method introduces a security gap while the command is running.
`-q` *delprof1* `-q` *delprof2* . . .	(Optional) Specifies the profile(s) to delete from the role.
`-r` *deluser1* `-r` *deluser2* . . .	(Optional) Specifies the user name(s) to delete from the role.

`-s` *shell*	(Optional) Specifies the full pathname of the program used as the role's shell on login. Valid entries are `/bin/pfcsh` (C shell), `/bin/pfksh` (Korn shell), and `/bin/pfsh` (Bourne shell), the default.
`-x` *autohome*=Y|N	(Optional) Sets the role's home directory. The home directory path in the password entry is set to `/home/`*login_name*.
`-x` *perm=home_perm*	(Optional) Sets the permissions on the role's home directory. *perm* is interpreted as an octal number, and the default is 0775.
`-M` *limit_privs*	Specifies the privilege name(s) to modify in a `user_attr(4)` entry. To add or change privileges, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.
`-D` *default_privs*	Specifies the default privilege name(s) to modify in a `user_attr(4)` entry.

The options to the `modify` subcommand listed below are available only if a system is configured with Solaris Trusted Extensions. See "Using Options that Require Solaris Trusted Extensions," below.

`-x` clear=*clearanceval*	(Optional) Specifies the role's clearance. *clearanceval* can be a string value or a hex value. If this option is not specified, the default, `admin_high`, is in effect. To set the clearance, the administrator must have the `solaris.admin.usermgr.labels` authorization.
`-x` label=*labelval*	(Optional) Specifies the role's minimum label. *labelval* can be a string label or a hex label. If this option is not specified, the default, `admin_low`, is in effect. To set the minimum label, the administrator must have the `solaris.admin.usermgr.labels` authorization.
`-x` labelview=HIDE|SHOW	(Optional) Specifies the second part of the `labelview` *key-value* pair. If `SHOW` is

specified, `labelview=*showsl` will be recorded. If `HIDE` is specified, `labelview=*hidesl` will be recorded. The asterisk portion can be replaced by "`internal,`", "`external,`", or ""(null). If this option is not specified, the default, `SHOW`, is in effect.

`-x view=INTERNAL|EXTERNAL|DEFAULT` — (Optional) Specifies the label view type for the `labelview` in `user_attr`. If `INTERNAL` is specified, `labelview=internal` will be recorded; if `EXTERNAL` is specified, `labelview=external` will be recorded; if `DEFAULT` is specified, nothing will be recorded in `user_attr`. If this option is not specified, the default, `INTERNAL`, is in effect.

Using Options that Require Solaris Trusted Extensions

To use an option that requires the Solaris Trusted Extensions feature, you must use the `-B` *toolbox* option to specify a toolbox that contains support for Trusted Extensions. For example:

```
# smrole add -H myhost -p mypasswd -u root -- -n role1 \
-F "Engineering Admin" -P abc123 -x clear=clearanceval \
-B http://<server>/toolboxes/tsol_files.tbx
```

In the command above, <*server*> is the name of the machine running the Solaris Management Console. See `smc(1M)` for a description of the `-B` option.

Examples

EXAMPLE 1 Creating a Role Account

The following creates the `role1` account with a full name of `Engineering Admin` and a password of `abc123` on the local file system, and assigns `user1` and `user2` to the role. This role has `Name Service Security` and `Audit Review` rights. The system assigns the next available unique UID greater than 100.

```
./smrole add -H myhost -p mypasswd -u root -- -n role1 \
          -F "Engineering Admin" -P abc123 -a user1 -a user2 \
          -p "Name Service Security" -p "Audit Review"
```

EXAMPLE 2 Deleting Role Accounts

The following deletes the `role1` and `role2` accounts from the local file system.

```
./smrole delete -H myhost -p mypasswd -u root -- -n role1 -n role2
```

EXAMPLE 3 Listing Role Accounts

The following lists all role accounts on the local file system in summary form.

```
./smrole list -H myhost -p mypasswd -u root --
```

EXAMPLE 4 Modifying a Role Account

The following modifies the `role1` account so the role defaults to the Korn shell, includes the `user3` account, and does not include the `user2` account.

```
./smrole modify -H myhost -p mypasswd -u root -- -n role1 \
          -s /bin/pfksh  -a user3 -r user2
```

Environment Variables

See `environ(5)` for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smrole` command. If this environment variable is not specified, the `/usr/java` location is used. See `smc(1M)`.

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` Invalid command syntax. A usage message displays.

`2` An error occurred while executing the command. An error message displays.

Files

The following files are used by the `smrole` command:

`/etc/aliases`	Mail aliases. See `aliases(4)`.
`/etc/auto_home`	Automatic mount points. See `automount(1M)`.
`/etc/group`	Group file. See `group(4)`.
`/etc/passwd`	Password file. See `passwd(4)`.
`/etc/security/policy.conf`	Configuration file for security policy. See `policy.conf(4)`.
`/etc/shadow`	Shadow password file. See `shadow(4)`.
`/etc/user_attr`	Extended user attribute database. See `user_attr(4)`.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga
Interface Stability	Evolving

See Also

`automount(1M)`, `smc(1M)`, `aliases(4)`, `group(4)`, `passwd(4)`, `policy.conf(4)`, `shadow(4)`, `user_attr(4)`, `attributes(5)`, `environ(5)`

Name smrsh – restricted shell for sendmail

Synopsis `smrsh -c` *command*

Description The `smrsh` program is intended as a replacement for the `sh` command in the `prog` mailer in `sendmail(1M)` configuration files. The `smrsh` program sharply limits commands that can be run using the `|program` syntax of `sendmail`. This improves overall system security. `smrsh` limits the set of programs that a programmer can execute, even if `sendmail` runs a program without going through an `alias` or `forward` file.

Briefly, `smrsh` limits programs to be in the directory `/var/adm/sm.bin`, allowing system administrators to choose the set of acceptable commands. It also rejects any commands with the characters: `,`, `<`, `>`, `|`, `;`, `&`, `$`, `\r` (RETURN), or `\n` (NEWLINE) on the command line to prevent end run attacks.

Initial pathnames on programs are stripped, so forwarding to `/usr/ucb/vacation`, `/usr/bin/vacation`, `/home/server/mydir/bin/vacation`, and `vacation` all actually forward to`/var/adm/sm.bin/vacation`.

System administrators should be conservative about populating `/var/adm/sm.bin`. Reasonable additions are utilities such as `vacation(1)` and `procmail`. Never include any shell or shell-like program (for example, `perl`) in the `sm.bin` directory. This does not restrict the use of `shell` or `perl` scrips in the `sm.bin` directory (using the `#!` syntax); it simply disallows the execution of arbitrary programs.

Options The following options are supported:

`-c` *command* Where *command* is a valid command, executes *command*.

Files `/var/adm/sm.bin` directory for restricted programs

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr, SUNWcsu

See Also `sendmail(1M)`, , `attributes(5)`

Name smserialport – manage serial port

Synopsis

```
/usr/sadm/bin/smserialport subcommand [auth_args] --
    [subcommand_args]
```

Description The `smserialport` command manages serial ports.

Sub-commands The following `smserialport` sub-commands (*subcommand*) are supported:

configure — Configures a serial port's basic settings for a device such as a terminal, modem or no connection.

delete — Deletes a given port. You can disable a port and prevent new services from being spawned for incoming connections, without interfering with existing services.

list — Lists all serial ports.

modify — Modifies a serial port's parameters.

Options There are two kinds of options: authentication arguments (*args*) and sub-command arguments (*subcommand_args*).

Authentication Arguments The `smserialport` authentication arguments, *args*, are derived from the `smc(1M)` argument set and are the same regardless of which sub-command you use.

Valid *args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes.

The single letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `--domain`.

Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

The following authentification arguments (*args*) are supported:

`-D` | `--domain` *domain* — Specifies the default domain that you want to manage. `smserialport` accepts only the *file* value for this option. *file* is also the default value.

The *file* default domain means that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the *domain* for all other tools.

`-H` | `--hostname` *host_name*:*port* — Specifies the host and port to which you want to connect. If you do not specify a port, the system connects to the default port, `898`. If you do not specify

a host (*host_name*:*port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc -B` option, or set your console preferences to load a `home toolbox` by default.

`-l` | `--rolepassword` *role_password*
: Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-p` | `--password` *password*
: Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Because passwords specified on the command line can be seen by any user on the system, this option is considered insecure.

`-r` | `--rolename` *role_name*
: Specifies a role name for authentication. If you do not specify this option, no role is assumed.

`-u` | `--username` *user_name*
: Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

`--`
: This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `--` option.

Sub-command Arguments

The sub-command specific options, *subcommand_args*, must come after the *args* and must be separated from them by the `--` option. Enclose descriptions and *arg* options that contain white space in double quotes.

configure
: The `configure` sub-command requires the following sub-command argument:

  `-n` *port_name*
  : Specifies the name of the serial port to reconfigure.

  The following sub-command arguments are optional for the `configure` sub-command:

  `-b` *baudrate*
  : Specifies the port baud rate. The supported baud rate are `38400`, `19200`, `9600`, `4800`, `2400`, `1200`, `300` and `auto`. The default is `9600`.

	`-c` *comment*	Specifies a short comment description of the service. The default is a description of the requested device type.
	`-h`	Displays the command's usage statement.
	`-l` *login_prompt*	Specifies the login prompt. The default is *tty*`'`*port_name*`'` `login:`.
	`-t` *terminal_type*	Specifies the terminal type. The default is `tvi925`.
	`-x` device=*device_name*	Specifies the device to be configured. Valid device_names are: `terminal`, `modemdialin`, `modemdialout`, `modemdialinout` or `initializeonly` for no connection. The default is `terminal`.
	`-x` service=y | n	Specifies the status of service, that is `y` for enabled or `n` for disabled. The default is `y`.

delete — The `delete` sub-command requires the following sub-command arguments:

`-n` *port_name* Specifies the name of the serial port to be disabled.

The following sub-command arguments are optional for the `delete` sub-command:

`-h` Displays the command's usage statement.

list — The `list` sub-command does not require any sub-command arguments.

The following sub-command arguments are optional for the `list` sub-command:

`-h` Displays the command's usage statement.

`-v` Displays the data in verbose format.

modify — The `modify` sub-command requires the following sub-command arguments:

`-n` *port_name* Specifies the name of the serial port to modify.

The following sub-command arguments are optional for the `modify` sub-command:

`-b` *baudrate*	Specifies the port baud rate. The supported baud rate are `38400`, `19200`, `9600`, `4800`, `2400`, `1200`, `300` and `auto`.
`-c` *comment*	A short comment description of the service.
`-h`	Displays the command usage statement.

-l *login_prompt*	Specifies the login prompt.
-t *terminal_type*	Specifies the terminal type.
-x bidirectional=y | n	Specifies the bi-directional port flag, y for set or n for not set. When this flag is set, the line can be used in both directions.
-x connect_on_carrier=y | n	Specifies if to connect on carrier, that is y or n.
-x initialize_only=y | n	Specifies if the service invocation. If y the service is invoked only once. This can be used to configure a particular device without actually monitoring it, as with software carrier.
-x service_program=*command*	Specifies the full pathname of the service command to invoke when a connection request is received.
-x service_status=y | n	Specifies the status of service, that is y for enabled or n for disabled.
-x software_carrier=y | n	Specifies the carrier detection. y for software or n for hardware.
-x timeout=*timeout*	Specifies the time to close a port if the open on the port succeeds, and no input data is received in timeout seconds. The supported *timeout* are never, 30, 60 and 90.

Examples

EXAMPLE 1 Listing Serial Ports

The following example lists the serial ports:

```
example% ./smserialport list -H myhost -u root -p mypassword --


Port Service  Baud-Rate Terminal-Type Prompt       Comment
a    enabled  9600      xterm         as           welcome
b    enabled  9600      tvi925        ttyb login:
```

EXAMPLE 2 Modifying Serial Ports

The following example contains two commands. The first command modifies serial port b for a baud rate of 4800, an xterm as terminal type, a b: for login prompt and a comment. The second command lists the ports.

```
example% ./smserialport modify -H myhost -u root -p mypassword -- \
     -n b -b 4800 -t xterm -l b: -c "modified port b"
```

EXAMPLE 2 Modifying Serial Ports *(Continued)*

```
example% ./smserialport list -H myhost -u root -p mypassword --


Port Service  Baud-Rate Terminal-Type Prompt      Comment
a    enabled  9600      xterm         as          welcome
b    enabled  4800      xterm         b:          modified port b
```

EXAMPLE 3 Deleting a Serial Port

The following example contains two commands. The first command deletes serial port b. The second command lists the ports.

```
example% ./smserialport delete -H myhost -u root \
     -p mypassword -- -n b

example% ./smserialport list -H myhost -u root -p mypassword --


Port Service  Baud-Rate Terminal-Type Prompt      Comment
a    enabled  9600      xterm         as          welcome
b    disabled 9600      tvi925        ttyb login:
```

EXAMPLE 4 Configuring a Serial Port

The following example contains two commands. The first command configures serial port b for a bi-directional modem. The second command lists the ports.

```
example% ./smserialport configure -H myhost -u root  \
     -p mypassword -- -n b -x device=modemdialinout

example% ./smserialport list -H myhost -u root -p mypassword --
Port Service  Baud-Rate Terminal-Type Prompt      Comment
a    enabled  9600      xterm         as          welcome
b    enabled  9600      tvi925        ttyb login: Modem - Dial In and Out
```

Environment Variables

See environ(5) for descriptions of the following environment variables that affect the execution of smserialport: JAVA_HOME. If this environment variable is not specified, the /usr/java location is used. See smc(1M).

Exit Status

The following exit values are returned:

0 Successful completion.

1 Invalid command syntax. A usage message displays.

2 An error occurred while executing the command. An error message displays.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga

See Also tip(1), pmadm(1M), sacadm(1M), smc(1M), ttyadm(1M), ttymon(1M), attributes(5), environ(5)

Name smtnrhdb – manage entries in the tnrhdb database

Synopsis `/usr/sadm/bin/smtnrhdb` *subcommand* `[`*auth_args*`] -- `*subcommand_args*`]`

Description The `smtnrhdb` command adds, modifies, deletes, and lists entries in the `tnrhdb` database.

The `tnrhdb` database specifies which remote-host template to use for each host, including the local host, in the distributed system. If a host's IP address cannot be matched to some entry in the `tnrhdb` database, communication with the host is not permitted.

The `smtnrhdb` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first `smc` connection can time out, so you might need to retry the command.

Valid Host Addresses and Wildcards

The trusted network software uses a network "longest prefix of matching bits" mechanism when looking for a host. The software looks first for the IP address of the host. If the software does not find this address, then the software falls back to searching for an IP address with the longest prefix of a matching bit pattern, and so on.

Note – The actual numeric value of the subnet address or other subnetting information on the system (for example, from the `netmasks(4)` file) are not considered by this mechanism.

Using the "longest prefix of matching bits" mechanism, an IPv4 address of 0.0.0.0 is a wildcard address with a prefix length of 0 and hence matches any IPv4 address. For more information about prefi x lengths in IPv4 and IPv6 addresses, see *System Administration Guide: IP Services*.

The `smtnrhdb` command accepts a hostname, IP address, and wildcard address with as optional prefix as valid addresses. See *subcommand_args*, below, for the format of valid addresses.

Sub-commands `smtnrhdb` *subcommand*s are:

`add`
Adds a new entry to the `tnrhdb` database. To add an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.

`delete`
Deletes an entry from the `tnrhdb` database. To delete an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.

`list`
Lists all entries in the `tnrhdb` database. To list an entry, the administrator must have the `solaris.network.host.read` and `solaris.network.security.read` authorizations.

`modify`
Modifies an entry in the `tnrhdb` database. To modify an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.

Options The `smtnrhdb` authentication arguments, *auth_args*, are derived from the `smc` arg set. These arguments are the same regardless of which subcommand you use.

The subcommand-specific options, *subcommand_args*, must be preceded by the `--` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user might be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `--domain`.

`-D` | `--domain` *domain*
Specifies the default domain that you want to manage. The syntax of *domain=type:/host_name/domain_name*, where *type* is `dns`, `ldap`, or `file`; *host_name* is the name of the server; and *domain_name* is the name of the domain you want to manage.

If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.

`-H` | `--hostname` *host_name:port*
Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`.

`-l` | `--rolepassword` *role_password*
Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-p` | `--password` *password*
Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-r` | `--rolename` *role_name*
Specifies a role name for authentication. If you do not specify this option, no role is assumed.

`-u` | `--username` *user_name*
Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

`--`
This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `--` option.

subcommand_args *Note:* Descriptions and other arg options that contain white spaces must be enclosed in double quotes.

`-h`
Displays the command's usage statement.

`-H` *hostname*
Specifies the name of the host. For the list subcommand, the hostname argument is not specified. This is not required if the ipaddress subcommand argument is specified.

`-i` *ipaddress*
Specifies the IP address of the host. This is not required if the hostname subcommand argument is specified. This option is not valid with the `-w` option.

`-n` *templatename*
Specifies the name of an existing template.

`-p` *prefixlen*
Specifies the prefix length (in bits) of a wildcard representation of the IP address. The prefix is the left-most portion of the IP address. This option is valid only with the `-w` option. For example, when the value of `-w` *ipaddress-wildcard* is 192.168.0.0, a *prefixlen* value of 24 indicates that the wildcard matches all addresses on the 192.168.0 network. With a *prefixlen* of 32, the wildcard 192.168.0.0 matches all addresses on the 192.168.0.0 network.

`-w` *ipaddress-wildcard*
Specifies the IP address of the subnet using a wildcard.

- One of the following sets of arguments must be specified for subcommand `add`:

  `-H` *hostname* `-n` *templatename* `|`
  `-i` *ipaddress* `-n` *templatename* `|`
  `-w` *ipaddress-wildcard* `-n` *templatename* `[ -p` *prefixlen* `] |`
  `-h`

- One of the following sets of arguments must be specified for subcommand `modify`:

  `-H` *hostname* `-n` *templatename* `|`
  `-i` *ipaddress* `-n` *templatename* `|`
  `-w` *ipaddress-wildcard* `-n` *templatename* `[ -p` *prefixlen* `] |`
  `-h`

- One of the following sets of arguments must be specified for subcommand `delete`:

  `-H` *hostname* `|`
  `-i` *ipaddress* `|`
  `-w` *ipaddress-wildcard* `[ -p` *prefixlen* `] |`
  `-h`

- The subcommand `list` takes the following argument:

  `-h`

Examples

EXAMPLE 1 Specifying the Template Name for a Wildcard IP Address

The admin role specifies the template name, `cipso_lan`, for a series of hosts that use the IP address wildcard `192.168.113.0` on the local file system. Since no authorization arguments were specified, the administrator connects to port 898 of the local host on the local server with the file domain type, which are the defaults. The administrator is prompted for the admin password.

```
$ usr/sadm/bin/smtnrhdb add -- -w 192.168.113.0 -n cipso_lan
```

EXAMPLE 2 Deleting an Entry in the `tnrhdb` Database

The admin role connects to port 898 (which happens to be the default) of the LDAP server and deletes a host entry from the database by specifying its IP address, `192.168.113.8`. Since the domain was not specified, the file domain type and local server are used by default. The administrator is prompted for the admin password.

```
# /usr/sadm/bin/smtnrhdb delete \
-D ldap:/example.domain -i 192.168.113.8
```

EXAMPLE 3 Adding a Subnet to the `tnrhdb` Database

The following command adds all the addresses on the 192.168.55.0 subnet, from 192.168.55.1 to 192.168.55.255, to the `tnrhdb` database:

```
# /usr/sadm/bin/smtnrhdb add \
-D file:/machine1.ExampleCo.COM/machine1.ExampleCo.COM \
 -- -w 192.168.55.0 -n cipso
Authenticating as user: root
Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::
Loading Tool: com.exampleco.admin.hostmgr.cli.smtnrhdb.HostMgrTnrhdbCli
from machine1.ExampleCo.COM
Login to machine1.ExampleCo.COM as user root was successful.
Download of com.exampleco.admin.hostmgr.cli.smtnrhdb.HostMgrTnrhdbCli
from machine1.ExampleCo.COM
was successful.
```

EXAMPLE 4 Adding Subnet 192.168.0 to the `tnrhdb` Database

The following command adds all the addresses on the 192.168.0 subnet, from 192.168.0.1 to 192.168.0.255 to the `tnrhdb` database. The prefix, 24, indicates that the first 24 bits (192.168.0) are fixed. Only the final zero is a wildcard.

```
# /usr/sadm/bin/smtnrhdb add \
-D file:/machine1.ExampleCo.COM/machine1.ExampleCo.COM \
 -- -w 192.168.0.0 -p 24 -n cipso

Login to machine1.ExampleCo.COM as user root was successful.
Download of com.exampleco.admin.hostmgr.cli.smtnrhdb.HostMgrTnrhdbCli
from machine1.ExampleCo.COM was successful.
```

Exit Status The following exit values are returned:

`0`
Successful completion.

`1`
Invalid command syntax. A usage message displays.

`2`
An error occurred while executing the command. An error message displays.

Files The following files are used by the `smtnrhdb` command:

`/etc/security/tsol/tnrhdb`
Trusted network remote-host database.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmgts
Interface Stability	Committed

See Also `smc(1M)`, `netmasks(4)`, `attributes(5)`

System Administration Guide: Security Services

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

Name smtnrhtp – manage entries in the trusted network template database

Synopsis `/usr/sadm/bin/smtnrhtp` *subcommand* [*auth_args*] -- [*subcommand_args*]

Description The `smtnrhtp` command adds, modifies, deletes, and lists entries in the `tnrhtp` database.

`smtnrhtp` *subcommand*s are:

`add` Adds a new entry to the `tnrhtp` database. To add an entry, the administrator must have the `solaris.network.security.read` and `solaris.network.security.write` authorizations.

`modify` Modifies an entry in the `tnrhtp` database. To modify an entry, the administrator must have the `solaris.network.security.read` and `solaris.network.security.write` authorizations.

`delete` Deletes an entry from `tnrhtp` database. To delete an entry, the administrator must have the `solaris.network.security.read` and `solaris.network.security.write` authorizations.

`list` Lists entries in the `tnrhtp` database. To list an entry, the administrator must have the `solaris.network.security.read` authorizations.

Options The `smtnrhtp` authentication arguments, *auth_args*, are derived from the `smc` argument set and are the same regardless of which subcommand you use. The `smtnrhtp` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first smc connection can time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must be *preceded* by the `--` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user might be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `--domain`.

`-D` | `--domain` *domain*
Specifies the default domain that you want to manage. The syntax of *domain=type:/host_name/domain_name*, where *type* is `dns`, `ldap`, or `file`; *host_name* is the name of the server; and *domain_name* is the name of the domain you want to manage.

If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.

`-H` | `--hostname` *host_name:port*
Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`.

`-l` | `--rolepassword` *role_password*
Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-p` | `--password` *password*
Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-r` | `--rolename` *role_name*
Specifies a role name for authentication. If you do not specify this option, no role is assumed.

`-u` | `--username` *user_name*
Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

`--`
This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `--` option.

subcommand_args Descriptions and other argument options that contain white spaces must be enclosed in double quotes.

`-h`	Displays the command's usage statement.
`-n` *templatename*	Specifies the name of the template.
`-t` *hosttype*	Specifies the host type of the new host. Valid values are `unlabeled` and `cipso`. The `cipso` host type is for hosts that use CIPSO (Common IP Security Options - Tag Type 1 only) to label packets.
`-x doi=`*doi-value*	Specifies the DOI value (the domain of interpretation). In the case of the `unlabeled` host type, this is the domain of interpretation for the `def_label`. The domain of interpretation defines the set of rules for translating between the external or internal representation of the security attributes and their network representation. When systems that are configured with Trusted Extensions software have the same `doi`, they share that set of rules. In the case of the `unlabeled` host type, these systems also share the same

interpretation for the default attributes that are assigned to the unlabeled templates that have that same `doi`.

`-x max=`*maximum-label* — Specifies the maximum label. Together with `min`, this value specifies the label accreditation range for the remote hosts that use this template. Values can be a hex value or string (such as `admin_high`).

`-x min=`*minimum-label* — Specifies the minimum label. Together with `max`, this value specifies the label accreditation range for the remote hosts that use this template. For gateway systems, `min` and `max` define the default range for forwarding labeled packets. The label range for routes is typically set by using a `route`(1M) subcommand with the `-secattr` option. When the label range for routes is not specified, the `min` to `max` range in the security template is used. Values can be a hex value or string (such as `admin_low`).

`-x label=`*default-label* — Specifies the default label to be applied to incoming data from remote hosts that do not support these attributes. This option does not apply if *hosttype* is `cipso`. Values can be a hex value or string (such as `admin_low`).

`-x slset=`*l1*,*l2*,*l3*,*l4* — Specifies a set of sensitivity labels. For gateway systems, the labels in `slset` are used for forwarding labeled packets. `slset` is optional. You can specify up to four label values, separated by commas. Values can be a hex value or string (such as `admin_low`).

- One of the following sets of arguments must be specified for subcommand `add`:

  `-n` *template name* (

  `-t cipso` [`-x doi=`*doi-value* `-x min=`*minimum-label* `-x max=`*maximum-label* `-x slset=`*l1*,*l2*,*l3*,*l4*] |

  `-t` unlabeled [`-x doi=`*doi-value* `-x min=`*minimum-label* `-x max=`*maximum-label* `-x label=`*default-label* `-x slset=`*l1*,*l2*,*l3*,*l4*] |

  `-h`

)

- One of the following sets of arguments must be specified for subcommand `modify`:

  `-n` *template name* (

  `-t cipso` [`-x doi=`*doi-value* `-x min=`*minimum-label* `-x max=`*maximum-label* `-x slset=`*l1*,*l2*,*l3*,*l4*] |

  `-t` unlabeled [`-x doi=`*doi-value* `-x min=`*minimum-label* `-x max=`*maximum-label* `-x label=`*default-label*`-x slset=`*l1*,*l2*,*l3*,*l4*] |

  `-h`

)

If the host type is changed, all options for the new host type must be specified.

- One of the following sets of arguments must be specified for subcommand `delete`:

  `-n` *templatename* |
  `-h`

- The following argument can be specified for subcommand `list`:

  `-n` *templatename* |
  `-h`

Examples

EXAMPLE 1 Adding a New Entry to the Network Template Database

The admin role connects to port `898` of the LDAP server and creates the `unlabeled_ntk` entry in the `tnrhtp` database. The new template is assigned a host type of `unlabeled`, a domain of interpretation of `1`, minimum label of `public`, maximum label of `restricted`, and a default label of `needtoknow`. The administrator is prompted for the admin password.

```
$ /usr/sadm/bin/smtnrhtp \
add -D ldap:directoryname -H servername:898 -- \
-n unlabeled_ntk -t unlabeled -x DOI=1 \
-x min=public -x max=restricted -x label="need to know"
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` Invalid command syntax. A usage message displays.

`2` An error occurred while executing the command. An error message displays.

Files

The following files are used by the `smtnrhtp` command:

`/etc/security/tsol/tnrhtp` Trusted network remote-host templates.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmgts
Interface Stability	Committed

See Also

`smc(1M)`, `attributes(5)`

Notes

The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

Warnings Changing a template while the network is up can change the security view of an undetermined number of hosts.

Allowing unlabeled hosts onto a Solaris Trusted Extensions network is a security risk. To avoid compromising the rest of your network, such hosts must be *trusted* in the sense that the administrator is certain that these unlabeled hosts will not be used to compromise the distributed system. These hosts should also be physically protected to restrict access to authorized individuals. If you cannot guarantee that an unlabeled host is physically secure from tampering, it and similar hosts should be isolated on a separate branch of the network.

If the security template is modified while the network is up, the changes do not take effect immediately unless `tnctl(1M)` is used to update the template entries. Otherwise, the changes take effect when next polled by the trusted network daemon, `tnd(1M)`. Administrators are allowed to add new templates and modify attributes of existing templates while the network is up.

Name smtnzonecfg – manage entries in the zone configuration database for Trusted Extensions networking

Synopsis `/usr/sadm/bin/smtnzonecfg` *subcommand* [*auth_args*] `--` [*subcommand_args*]

Description The `smtnzonecfg` command adds, modifies, deletes, and lists entries in the `tnzonecfg` database.

`smtnzonecfg` *subcommand*s are:

`add` Adds a new entry to the `tnzonecfg` database. To add an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.

`modify` Modifies an entry in the `tnzonecfg` database. To modify an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.

`delete` Deletes an entry from the `tnzonecfg` database. To delete an entry, the administrator must have the `solaris.network.host.write` and `solaris.network.security.write` authorizations.

`list` Lists entries in the `tnzonecfg` database. To list an entry, the administrator must have the `solaris.network.host.read` and `solaris.network.security.read` authorizations.

Options The `smtnzonecfg` authentication arguments, *auth_args*, are derived from the `smc` argument set and are the same regardless of which subcommand you use. The `smtnzonecfg` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc`(1M)). After rebooting the Solaris Management Console server, the first smc connection can time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must be *preceded* by the `--` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u`; they are all optional. If no *auth_args* are specified, certain defaults will be assumed and the user can be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `--domain`.

`-D` | `--domain` *domain*
Specifies the default domain that you want to manage. The syntax of *domain=type:/host_name/domain_name*, where *type* is `dns`, `ldap`, or `file`; *host_name* is the name of the server; and *domain_name* is the name of the domain you want to manage.

If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis. This option specifies the domain for all other tools.

`-H` | `--hostname` *host_name:port*
Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`.

`-l` | `--rolepassword` *role_password*
Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-p` | `--password` *password*
Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.

`-r` | `--rolename` *role_name*
Specifies a role name for authentication. If you do not specify this option, no role is assumed.

`-u` | `--username` *user_name*
Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.

`--`
This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `--` option.

subcommand_args Descriptions and other argument options that contain white spaces must be enclosed in double quotes.

`-h`
Displays the command's usage statement.

`-n` *zonename*
Specifies the zone name for the entry. This name is used when the zone is configured. See `zonecfg(1M)`, under the `-z zonename` option, for the constraints on zone names. The specified zone name must be one of the configured zones on the system. The following command returns a list of configured zones:

```
/usr/sbin/zoneadm list -c
```

`-l` *label*
Specifies the label for the zone. This field is used to label the zone when the zone is booted. Each zone must have a unique label.

`-x policymatch=0|1`
Specifies the policy match level for non-transport traffic. Only values of `0` (match the label) or `1` (be within the label range of the zone) are accepted.

ICMP packets that are received on the global zone IP address are accepted based on the label range of the global zone's security template if the global zone's *policymatch* field is set to 1. When this field is set to 0 for a zone, the zone will not respond to an ICMP echo request from a host with a different label.

This subcommand argument is optional. If not specified, it will have a default value of 0.

`-x mlpzone=“”|`*port/protocol*
Specifies the multilevel port configuration entry for zone-specific IP addresses. Multiple *port/protocol* combinations are separated by a semi-colon. The empty string can be specified to remove all existing MLP zone values. This subcommand argument is optional.

An MLP is used to provide multilevel service in the global zone as well as in non-global zones. As an example of how a non-global zone can use an MLP, consider setting up two labeled zones, `internal` and `public`. The `internal` zone can access company networks; the `public` zone can access public internet but not the company's internal networks. For safe browsing, when a user in the `internal` zone wants to browse the Internet, the `internal` zone browser forwards the URL to the `public` zone, and the web content is then displayed in a `public` zone web browser. That way, if the download in `public` zone compromises the web browser, it cannot affect the company's internal network. To set this up, TCP port `8080` in the `public` zone is an MLP (`8080/tcp`), and the security template for the `public` zone has a label range from `PUBLIC` to `INTERNAL`.

`-x mlpshared=“”|`*port/protocol*
Specifies the multilevel port configuration entry for shared IP addresses. Multiple *port/protocol* combinations are separated by a semi-colon. The empty string can be specified to remove all existing MLP shared values. This subcommand argument is optional.

A shared IP address can reduce the total number of IP addresses that are needed on the system, especially when configuring a large number of zones. Unlike the case of the zone-specific IP address, when MLPs are declared on shared IP addresses, only the global zone can receive the incoming network traffic that is destined for the MLP.

- One of the following sets of arguments must be specified for subcommand `add`:

```
-n zonename -l label [-x policymatch=policy-match-level \
-x mlpzone=port/protocol;.... | \
-x mlpshared=port/protocol;.... ]
-h
```

- One of the following sets of arguments must be specified for subcommand `modify`:

```
-n zonename [-l label] [-x policymatch=policy-match-level \
-x mlpzone=port/protocol;.... |\
-x mlpshared=port/protocol;.... ]
-h
```

- One of the following arguments must be specified for subcommand `delete`:

```
-n zonename |
-h
```

- The following argument can be specified for subcommand `list`:

```
-n zonename |
-h
```

Examples

EXAMPLE 1 Adding a New Entry to the Zone Configuration Database

The admin role creates a new zone entry, `public`, with a label of `public`, a policy match level of 1, and a shared MLP port and protocol of 666 and TCP. The administrator is prompted for the admin password.

```
$ /usr/sadm/bin/smtnzonecfg add -- -n public -l public \
-x policymatch=1 -x mlpshared=666/tcp
```

EXAMPLE 2 Modifying an Entry in the Zone Configuration Database

The admin role changes the `public` entry in the `tnzonecfg` database to `needtoknow`. The administrator is prompted for the admin password.

```
$ /usr/sadm/bin/smtnzonecfg modify -- -n public -l needtoknow
```

EXAMPLE 3 Listing the Zone Configuration Database

The admin role lists the entries in the `tnzonecfg` database. The administrator is prompted for the admin password.

```
$ /usr/sadm/bin/smtnzonecfg list --
```

Exit Status

The following exit values are returned:

`0` Successful completion.

1 Invalid command syntax. A usage message displays.

2 An error occurred while executing the command. An error message displays.

Files

The following files are used by the `smtnzonecfg` command:

`/etc/security/tsol/tnzonecfg` Trusted zone configuration database.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmgts
Interface Stability	Committed

See Also smc(1M), attributes(5)

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

Name smuser – manage user entries

Synopsis

```
/usr/sadm/bin/smuser subcommand [ auth_args] - —
     [subcommand_args]
```

Description The `smuser` command manages one or more user entries in the local `/etc` filesystem or a NIS or NIS+ target name service.

subcommands `smuser` *subcommand*s are:

`add` Adds a new user entry to the appropriate files. You can use a template and input file instead of supplying the additional command line options. If you use a template and command line options, the command line options take precedence and override any conflicting template values. To add an entry, the administrator must have the `solaris.admin.usermgr.write` authorization.

`delete` Deletes one or more user entries from the appropriate files. To delete an entry, the administrator must have the `solaris.admin.usermgr.write` authorization. *Note:* You cannot delete the system accounts with IDs less than 100, or 60001, 60002, or 65534.

`list` Lists one more user entries from the appropriate files. To list entries, the administrator must have the `solaris.admin.usermgr.read` authorization.

`modify` Modifies a user entry in the appropriate files. To modify an entry, the administrator must have the `solaris.admin.usermgr.write` authorization.

Options The `smuser` authentication arguments, *auth_args*, are derived from the `smc(1M)` arg set and are the same regardless of which subcommand you use. The `smuser` command requires the Solaris Management Console to be initialized for the command to succeed (see `smc(1M)`). After rebooting the Solaris Management Console server, the first Solaris Management Console connection might time out, so you might need to retry the command.

The subcommand-specific options, *subcommand_args*, must come after the *auth_args* and must be separated from them by the `- —` option.

auth_args The valid *auth_args* are `-D`, `-H`, `-l`, `-p`, `-r`, and `-u` are described below. They are all optional. These options are a subset of the full complement of supported options described in `smc(1M)`.

If no *auth_args* are specified, certain defaults will be assumed and the user may be prompted for additional information, such as a password for authentication purposes. These letter options can also be specified by their equivalent option words preceded by a double dash. For example, you can use either `-D` or `- —domain` with the *domain* argument.

`-D | - —domain` *13;domain* Specifies the default domain that you want to manage. The syntax of *domain* is *type*:/*host_name*/*domain_name*, where *type* is `nis`, `nisplus`, `dns`, `ldap`, or `file`; *host_name* is

	the name of the machine that serves the domain; and *domain_name* is the name of the domain you want to manage. (*Note:* Do *not* use `nis+` for `nisplus`.) If you do not specify this option, the Solaris Management Console assumes the `file` default domain on whatever server you choose to manage, meaning that changes are local to the server. Toolboxes can change the domain on a tool-by-tool basis; this option specifies the domain for all other tools.
`-H | - —hostname` *13;host_name:port*	Specifies the *host_name* and *port* to which you want to connect. If you do not specify a *port*, the system connects to the default port, `898`. If you do not specify *host_name:port*, the Solaris Management Console connects to the local host on port `898`. You may still have to choose a toolbox to load into the console. To override this behavior, use the `smc(1M) -B` option, or set your console preferences to load a "home toolbox" by default.
`-l | - —rolepassword` *13;role_password*	Specifies the password for the *role_name*. If you specify a *role_name* but do not specify a *role_password*, the system prompts you to supply a *role_password*. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-p | - —password` *13;password*	Specifies the password for the *user_name*. If you do not specify a password, the system prompts you for one. Passwords specified on the command line can be seen by any user on the system, hence this option is considered insecure.
`-r | - —rolename` *13;role_name*	Specifies a role name for authentication. If you do not specify this option, no role is assumed.
`-u | - —username` *13;user_name*	Specifies the user name for authentication. If you do not specify this option, the user identity running the console process is assumed.
`- —`	This option is required and must always follow the preceding options. If you do not enter the preceding options, you must still enter the `- —`

option.

subcommand_args *Note:* Descriptions and other arg options that contain whitespace must be enclosed in double quotes.

To add or change privileges, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.

- For subcommand `add`:

`-c` *comment*	(Optional) Includes a short description of the login, which is typically the user's name. Consists of a string of up to 256 printable characters, excluding the colon (:).
`-d` *dir*	(Optional) Specifies the home directory of the new user, limited to 1024 characters.
`-e` *ddmmyyyy*	(Optional) Specifies the expiration date for a login. After this date, no user can access this login. This option is useful for creating temporary logins. Specify a null value (“ ”) to indicate that the login is always valid. The administrator must have the `solaris.admin.usermgr.pswd` authorization.
`-f` *inactive*	(Optional) Specifies the maximum number of days allowed between uses of a login ID before that ID is declared invalid. Normal values are positive integers. Enter zero to indicate that the login account is always active.
`-F` *full_name*	(Optional) Specifies the full, descriptive name of the user. The *full_name* must be unique within a domain and can contain alphanumeric characters and spaces. If you use spaces, you must enclose the *full_name* in double quotes.
`-g` *group*	(Optional) Specifies the new user's primary group membership in the system group database with an existing group's integer ID.
`-G` *group1* `-G` *group2* . . .	(Optional) Specifies the new user's supplementary group membership in the system group database with the character string names of one or more existing groups. Duplicates of groups specified with the `-g` and `-G` options are ignored.
`-h`	(Optional) Displays the command's usage statement.
`-n` *login*	Specifies the new user's login name. The login name must be unique within a domain, contain 2–32 alphanumeric characters, begin with a letter, and contain at least one lowercase letter.

`-P` *password*	(Optional) Specifies up to an eight-character password assigned to the user account. *Note:* When you specify a password, you type the password in plain text. Specifying a password using this method introduces a security gap while the command is running. To set the password, the administrator must have the `solaris.admin.usermgr.pswd` authorization.
`-s` *shell*	(Optional) Specifies the full pathname (limited to 1024 characters) of the program used as the user's shell on login. Valid entries are a user-defined shell, `/bin/csh` (C shell), `bin/ksh` (Korn shell), and the default, `/bin/sh` (Bourne shell).
`-t` *template*	(Optional) Specifies a template, created using the User Manager tool, that contains a set of pre-defined user attributes. You may have entered a name service server in the template. However, when a user is actually added with this template, if a name service is unavailable, the user's local server will be used for both the Home Directory Server and Mail Server.
`-u` *uid*	(Optional) Specifies the user ID of the user you want to add. If you do not specify this option, the system assigns the next available unique user ID greater than 100.
`-x` *autohome*=Y|N	(Optional) Sets the home directory to automount if set to `Y`. The user's home directory path in the password entry is set to `/home/`*login name*.
`-x` *mail=mail_server*	(Optional) Specifies the host name of the user's mail server, and creates a mail file on the server. Users created in a local scope must have a mail server created on their local machines.
`-x` *perm=home_perm*	(Optional) Sets the permissions on the user's home directory. *perm* is interpreted as an octal number, and the default is `0775`.
`-x` *pwmax=days*	(Optional) Specifies the maximum number of days that the user's password is valid. The administrator must have the `solaris.admin.usermgr.pswd` authorization.
`-x` *pwmin=days*	(Optional) Specifies the minimum number of days between user password changes. The administrator must have the `solaris.admin.usermgr.pswd` authorization.
`-x` *pwwarn=days*	(Optional) Specifies the number of days relative to *pwmax* that the user is warned about password expiration prior to the password expiring. The administrator must have the `solaris.admin.usermgr.pswd` authorization.

`-x` *serv=homedir_server*	(Optional) Specifies the name of the server where the user's home directory resides. Users created in a local scope must have their home directory server created on their local machines.
`-M` *limit_privs*	Specifies the privilege name(s) to add to the new `user_attr(4)` entry. The default is `all` for limit privilege. To add or change privileges, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.
`-D` *default_privs*	Specifies the default privilege name(s) to add to the new `user_attr(4)` entry.

The following options to the `add` subcommand are available only if a system is configured with Solaris Trusted Extensions. See "Using Options that Require Solaris Trusted Extensions," below.

`-x` clear=*clearanceval*	(Optional) Specifies the role's clearance. *clearanceval* can be a string value or a hex value. If this option is not specified, the default is the user's system default clearance. To set the clearance, the administrator must have the `solaris.admin.usermgr.labels` authorization.
`-x` idlecmd=LOGOUT|LOCK	Specifies the command to execute if the system has been idled. If `LOGOUT` is specified, `idlecmd=logout` will be recorded in `user_attr`. If `LOCK` is specified, `idlecmd=lock` will be recorded in `user_attr`. If this option is not specified, the default is the `IDLECMD` in the `/etc/security/policy.conf` file.
`-x` idletime=*minutes*	(Optional) Specifies the number of minutes before the specified idle command gets executed. Any integer value in the range from 1 to 120 is valid. This value is recorded in `user_attr` as `idletime=`*val*. If this option is not specified, the default is the `IDLETIME` in the `/etc/security/policy.conf` file.

-x label=*labelval*	(Optional) Specifies the user's minimum label. *labelval* can be a string label or a hex label. If this option is not specified, the default is the user's system default minimum label. To set the minimum label, the administrator must have the `solaris.admin.usermgr.labels` authorization.
-x labelview=HIDE|SHOW	(Optional) Specifies the second part of the `labelview` *key-value* pair. If `SHOW` is specified, `labelview=*showsl` will be recorded. If `HIDE` is specified, `labelview=*hidesl` will be recorded. The asterisk portion can be replaced by "`internal,`", "`external,`", or ""(null). If this option is not specified, the default is the `LABELVIEW` in the `/etc/security/policy.conf` file.
-x lock=Y|N	(Optional) Specifies if an account is locked after a specified number of failed logins. This value is recorded in `user_attr` as `lock_after_retries`. If this option is not specified, the default is the `LOCK_AFTER_RETRIES` in the `/etc/security/policy.conf` file.
-x view=INTERNAL|EXTERNAL|DEFAULT	(Optional) Specifies the label view type for the `labelview` in `user_attr`. If `INTERNAL` is specified, `labelview=internal` will be recorded; if `EXTERNAL` is specified, `labelview=external` will be recorded; if `DEFAULT` is specified, nothing will be recorded in `user_attr`. If this option is not specified, the default action, that nothing gets recorded in `user_attr`, is in effect.

- For subcommand `delete`:

-h	(Optional) Displays the command's usage statement.
-n *login1*	Specifies the login name of the user you want to delete.

`-n` *login2* . . . (Optional) Specifies the additional login name(s) of the user(s) you want to delete.

- For subcommand `list`:

  `-h` (Optional) Displays the command's usage statement.

  `-l` Displays the output for each user in a block of *key:value* pairs (for example, `user name:root`) followed by a blank line to delimit each user block. Each *key:value* pair is displayed on a separate line. The keys are: `autohome setup`, `comment`, `days to warn`, `full name`,`home directory`, `home directory permissions`, `login shell`, `mail server`, `max days change`, `max days inactive`, `min days change`, `password expires`, `password type`, `primary group`, `rights`, `roles`, `secondary groups`, `server`, `user ID (UID)`, and `user name`.

  `-n` *login1* Specifies the login name of the user you want to list.

  `-n` *login2* . . . (Optional) Specifies the additional login name(s) of the user(s) you want to list.

- For subcommand `modify`:

  `-a` *addrole1* `-a` *addrole2* . . . (Optional) Specifies the role(s) to add to the user account. To assign a role to a user, the administrator must have the `solaris.role.assign` authorization or must have the `solaris.role.delegate` authorization and be a member of each of the roles specified.

  `-c` *comment* (Optional) Describes the changes you made to the user account. Consists of a string of up to 256 printable characters, excluding the colon (:).

  `-d` *description* (Optional) Specifies the user's home directory, limited to 1024 characters.

  `-e` *ddmmyyyy* (Optional) Specifies the expiration date for a login in a format appropriate to the locale. After this date, no user can access this login. This option is useful for creating temporary logins. Specify a null value (" ") to indicate that the login is always valid.

  `-f` *inactive* (Optional) Specifies the maximum number of days allowed between uses of a login ID before the ID is declared invalid. Normal values are positive integers. Specify zero to indicate that the login account is always active.

  `-F` *full_name* (Optional) Specifies the full, descriptive name of the user. The *full_name* must be unique within a domain and can

	contain alphanumeric characters and spaces. If you use spaces, you must enclose the *full_name* in double quotes.
`-g` *group*	(Optional) Specifies the new user's primary group membership in the system group database with an existing group's integer ID.
`-G` *group1* `-G` *group2* . . .	(Optional) Specifies the new user's supplementary group membership in the system group database with the character string names of one or more existing groups. Duplicates of groups specified with the `-g` and `-G` options are ignored.
`-h`	(Optional) Displays the command's usage statement.
`-n` *name*	Specifies the user's current login name.
`-N` *new_name*	(Optional) Specifies the user's new login name. The login name must be unique within a domain, contain 2–32 alphanumeric characters, begin with a letter, and contain at least one lowercase letter.
`-p` *addprof1* `-p` *addprof2* . . .	(Optional) Specifies the profile(s) to add to the user account. To assign a profile to a user, the administrator must have the `solaris.profmgr.assign` or `solaris.profmgr.delegate` authorization.
`-P` *password*	(Optional) Specifies up to an eight-character password assigned to the user account. When you specify a password, you type the password in plain text. Specifying a password using this method introduces a security gap while the command is running.
`-q` *delprof1* `-q` *delprof2* . . .	(Optional) Specifies the profile(s) to delete from the user account.
`-r` *delrole1* `-r` *delrole2* . . .	(Optional) Specifies the role(s) to delete from the user account.
`-s` *shell*	(Optional) Specifies the full pathname (limited to 1024 characters) of the program used as the user's shell on login. Valid entries are a user-defined shell, `/bin/csh` (C shell), `bin/ksh` (Korn shell), and the default, `/bin/sh` (Bourne shell).l)
`-x` *autohome*=Y|N	(Optional) Sets up the home directory to automount if set to `Y`. The user's home directory path in the password entry is set to `/home/`*login name*.

`-x` *pwmax=days*	(Optional) Specifies the maximum number of days that the user's password is valid.
`-x` *pwmin=days*	(Optional) Specifies the minimum number of days between password changes.
`-x` *pwwarn=days*	(Optional) Specifies the number of days relative to *pwmax* that the user is warned about password expiration before the password expires.
`-M` *limit_privs*	Specifies the privilege name(s) to modify in the `user_attr(4)` entry. The default is `all` for limit privilege. To add or change privileges, the administrator must have the `solaris.admin.privilege.write` authorization. See `privileges(5)`.
`-D` *default_privs*	Specifies the default privilege name(s) to modify in the `user_attr(4)` entry.

The following options to the `modify` subcommand are available only if a system is configured with Solaris Trusted Extensions. See "Using Options that Require Solaris Trusted Extensions," below.

`-x` clear=*clearanceval*	(Optional) Specifies the role's clearance. *clearanceval* can be a string value or a hex value. If this option is not specified, the default is the user's system default clearance. To set the clearance, the administrator must have the `solaris.admin.usermgr.labels` authorization.
`-x` idlecmd=LOGOUT|LOCK	Specifies the command to execute if the system has been idled. If `LOGOUT` is specified, `idlecmd=logout` will be recorded in `user_attr`. If `LOCK` is specified, `idlecmd=lock` will be recorded in `user_attr`. If this option is not specified, the default is the `IDLECMD` in the `/etc/security/policy.conf` file.
`-x` idletime=*minutes*	(Optional) Specifies the number of minutes before the specified idle command gets executed. Any integer value in the range from 1 to 120 is valid.

	This value is recorded in `user_attr` as `idletime=`*val*. If this option is not specified, the default is the `IDLETIME` in the `/etc/security/policy.conf` file.
-x label=*labelval*	(Optional) Specifies the user's minimum label. *labelval* can be a string label or a hex label. If this option is not specified, the default is the user's system default minimum label. To set the minimum label, the administrator must have the `solaris.admin.usermgr.labels` authorization.
-x labelview=HIDE|SHOW	(Optional) Specifies the second part of the `labelview` *key-value* pair. If `SHOW` is specified, `labelview=*showsl` will be recorded. If `HIDE` is specified, `labelview=*hidesl` will be recorded. The asterisk portion can be replaced by "`internal,`", "`external,`", or ""(null). If this option is not specified, the default is the `LABELVIEW` in the `/etc/security/policy.conf` file.
-x lock=Y|N	(Optional) Specifies if an account is locked after a specified number of failed logins. This value is recorded in `user_attr` as `lock_after_retries`. If this option is not specified, the default is the `LOCK_AFTER_RETRIES` in the `/etc/security/policy.conf` file.
-x view=INTERNAL|EXTERNAL|DEFAULT	(Optional) Specifies the label view type for the `labelview` in `user_attr`. If `INTERNAL` is specified, `labelview=internal` will be recorded; if `EXTERNAL` is specified, `labelview=external` will be recorded; if `DEFAULT` is specified, nothing will be recorded in `user_attr`. If this option is not specified, the default action, that nothing gets recorded in `user_attr`, is in effect.

Using Options that Require Solaris Trusted Extensions

To use an option that requires the Solaris Trusted Extensions feature, you must use the `-B` *toolbox* option to specify a toolbox that contains support for Trusted Extensions. For example:

```
# smuser add -H myhost -p mypasswd -x idlecmd=LOGOUT \
-B http://<server>/toolboxes/tsol_files.tbx
```

In the command above, <*server*> is the name of the machine running the Solaris Management Console. See smc(1M) for a description of the `-B` option.

Examples

EXAMPLE 1 Creating a New User Account

The following creates a new user account on the local file system. The account name is `user1`, and the full name is `Joe Smith`. The comment field verifies that the account is for Joe Smith. The system will assign the next available user ID greater than 100 to this account. There is no password set for this account, so when Joe Smith logs in for the first time, he will be prompted to enter a password.

```
./smuser add -H myhost -p mypasswd -u root -- -F "Joe Smith" \
             -n user1 -c "Joe's account"
```

EXAMPLE 2 Deleting a User Account

The following deletes the `user1` account from the local file system:

```
./smuser delete -H myhost -p mypasswd -u root -- -n user1
```

EXAMPLE 3 Listing All User Accounts

The following lists all user accounts on the local file system in summary form:

```
./smuser list -H myhost -p mypasswd -u root --
```

EXAMPLE 4 Modifying a User Account

The following modifies the `user1` account to default to a Korn shell, and assigns the account to the `qa_group` secondary group.

```
./smuser modify -H myhost -p mypasswd -u root -- -n user1 \
             -s /bin/ksh -G qa_group
```

Environment Variables

See environ(5) for a description of the `JAVA_HOME` environment variable, which affects the execution of the `smuser` command. If this environment variable is not specified, the `/usr/java` location is used. See smc(1M).

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` Invalid command syntax. A usage message displays.

`2` An error occurred while executing the command. An error message displays.

Files The following files are used by the `smuser` command:

`/etc/aliases`	Mail aliases. See `aliases(4)`.
`/etc/auto_home`	Automatic mount points. See `automount(1M)`.
`/etc/group`	Group file. See `group(4)`.
`/etc/passwd`	Password file. See `passwd(4)`.
`/etc/security/policy.conf`	Configuration file for security policy. See `policy.conf(4)`.
`/etc/shadow`	Shadow password file. See `shadow(4)`.
`/etc/user_attr`	Extended user attribute database. See `user_attr(4)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmga
Interface Stability	Evolving

See Also `automount(1M)`, `smc(1M)`, `aliases(4)`, `group(4)`, `passwd(4)`, `policy.conf(4)`, `shadow(4)`, `user_attr(4)`, `attributes(5)`, `environ(5)`

Name snmpbulkget – communicate with a network entity using SNMP GETBULK requests

Synopsis

```
snmpbulkget [application options] [common options] oid
     [oid]...
```

Description The `snmpbulkget` utility is an SNMP application that uses the SNMP `GETBULK` operation to send information to a network manager. You can specify one or more object identifiers (OIDs) on the command line. Each variable name must be entered in the format specified in `snmp_variables(4)`.

If the network entity has an error processing the request packet, an error packet is returned and a message displayed, indicating the way in which the request was malformed.

Options The following options are supported:

`-Cn`*num* Set the `non-repeaters` field in the `GETBULK` PDU. This specifies the number of supplied variables that should not be iterated over. The default is 0.

`-Cr`*num* Set the `max-repetitions` field in the `GETBULK` PDU. This specifies the maximum number of iterations over the repeating variables. The default is 10.

In addition to this option, `snmpbulkget` takes the common options described in the `snmpcmd(1M)` manual page.

Examples **EXAMPLE 1** Retrieving Multiple Objects

The following `snmpbulkget` command retrieves the variable `system.sysDescr.0` (which is the lexicographically next object to `system`) and the first five objects in the `ifTable`:

```
# snmpbulkget -v2c -Cn1 -Cr5 -Os -c public zeus system ifTable
```

This command produces output such as the following:

```
sysDescr.0 = STRING: "SunOS zeus.net.cmu.edu 4.1.3_U1 1 sun4m"
ifIndex.1 = INTEGER: 1
ifIndex.2 = INTEGER: 2
ifDescr.1 = STRING: "lo0"
[...]
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors.

2 An error occurred while executing the command. An error message is displayed.

See Also snmpcmd(1M), snmp_variables(4), attributes(5)

RFC 1905

Notes As the name implies, `snmpbulkget` uses the SNMP `GETBULK` message, which is not available in SNMPv1.

Name snmpbulkwalk – communicate with a network entity using SNMP BULK requests

Synopsis

```
/usr/sfw/bin/snmpbulkwalk [application_options]
     [common_options] [oid]
```

Description The `snmpbulkwalk` utility is an SNMP application that uses SNMP `GETBULK` requests to query a network entity efficiently for a tree of information.

You can specify an object identifier (OID) on the command line. This OID identifies the portion of the object identifier space that will be searched using `GETBULK` requests. All variables in the subtree below the given OID are queried and their values returned. Each variable name is given in the format specified in `snmp_variables(4)`. If no OID argument is present, `snmpbulkwalk` searches MIB-2.

If a network entity has an error processing the request packet, an error packet is returned and a message is displayed. The message helps to pinpoint the way in which the request was malformed.

If the tree search causes attempts to search beyond the end of the MIB, the message "End of MIB" is displayed.

Options The following options are supported:

`-Cc` Do not check whether the returned OIDs are increasing. Some agents (agents for Laser-Jet printers are an example) return OIDs out of order, but can complete the walk anyway. Other agents return OIDs that are out of order and can cause `snmpbulkwalk` to loop indefinitely. By default, `snmpbulkwalk` tries to detect this behavior and warns you when it hits an agent acting illegally. Use `-Cc` to turn off this behavior.

`-Ci` Include the given OID in the search range. Normally, `snmpbulkwalk` uses `GETBULK` requests starting with the OID you specify and returns all results in the MIB tree beyond that OID. Use this option to include the OID specified on the command line in the printed results if it is a valid OID in the tree itself.

`-Cn`*num* Set the non-repeaters field in the GETBULK PDUs. This specifies the number of supplied variables that should not be iterated over. The default is 0.

`-Cp` Upon completion of the walk, display the number of variables found.

`-Cr`*num* Set the `max-repetitions` field in the `GETBULK` PDUs. This specifies the maximum number of iterations over the repeating variables. The default is 10.

In addition to these options, `snmpbulkwalk` takes the common options described in the `snmpcmd(1M)` manual page.

Examples

EXAMPLE 1 Retrieving Variables Under `system`

The following command retrieves all of the variables under `system`:

```
# snmpbulkwalk -v2c -Os -c public zeus system
```

The return from `snmpbulkwalk` is as follows:

```
sysDescr.0 = STRING: "SunOS zeus.net.cmu.edu 4.1.3_U1 1 sun4m"
sysObjectID.0 = OID: enterprises.hp.nm.hpsystem.10.1.1
sysUpTime.0 = Timeticks: (155274552) 17 days, 23:19:05
sysContact.0 = STRING: ""
sysName.0 = STRING: "zeus.net.cmu.edu"
sysLocation.0 = STRING: ""
sysServices.0 = INTEGER: 72
```

In contrast to `snmpwalk(1M)`, this information will be gathered in a single transaction with the agent, rather than one transaction per variable found. `snmpbulkwalk` is thus more efficient in terms of network utilization, which might be especially important when retrieving large tables.

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message displays. Also used for timeout errors.

`2` An error occurred while executing the command. An error message displays.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also

`snmpcmd(1M)`, `snmpwalk(1M)`, `snmp_variables(4)`, `attributes(5)`

Notes

As the name implies, `snmpbulkwalk` uses the SNMP `GETBULK` message, which is not available in SNMP v1.

Name snmpcmd – commands to communicate with a network entity using SNMP requests

Synopsis `snmpcmd [options] agent [parameters]`

Description This manual page describes the common options for the following SNMP commands:

- `snmpbulkget(1M)`
- `snmpbulkwalk(1M)`
- `snmpdf(1M)`
- `snmpget(1M)`
- `snmpgetnext(1M)`
- `snmpnetstat(1M)`
- `snmpset(1M)`
- `snmptrap(1M)`
- `snmpusm(1M)`
- `snmpvacm(1M)`
- `snmpwalk(1M)`

The command line applications use the SNMP protocol to communicate with an SNMP-capable network entity, an agent. Individual applications usually (but not invariably) take additional parameters that are given after the agent specification. These parameters are documented in the manual pages for each application.

Options The following options are supported:

`-a` *authProtocol*	Set the authentication protocol (`MD5`|`SHA`) used for authenticated SNMPv3 messages. Overrides the `defAuthType` token in the `snmp.conf` file.
A *authPassword*	Set the authentication pass phrase used for authenticated SNMPv3 messages. Overrides the `defAuthPassphrase` token in the `snmp.conf` file. Insecure to specify pass word phrases on the command line, see `snmp.conf(4)`.
`-c` *community*	Set the community string for SNMPv1/v2c transactions. Overrides the `defcommunity` token in the `snmp.conf` file.
`-d`	Dump (in hexadecimal) the sent and received SNMP packets.
`-D` *token*[,...]	Turn on debugging output for the given *token*(s). Try `ALL` for extremely verbose output.
`-e` *engineID*	Set the authoritative (security) *engineID* used for SNMPv3 `REQUEST` messages. This is the `engineID` of the agent or proxy (for example, 800000020109840301). This value will be discovered if not supplied.

`-E` *engineID*
: Set the context *engineID* used for SNMPv3 REQUEST messages `scopedPdu`. This is the `engineID` of the agent (for example, 800000020109840301). This will be the authoritative `engineID` if not specified.

`-h`, `--help`
: Display a brief usage message and then exit.

`-H`
: Display a list of configuration file directives understood by the command and then exit.

`-I` brRhu
: Specifies input parsing options. See INPUT OPTIONS below.

`-l` *secLevel*
: Set the securityLevel used for SNMPv3 messages (`noAuthNoPriv`|`authNoPriv`|`authPriv`). Appropriate pass phrase(s) must be provided when using any level higher than `noAuthNoPriv`. Overrides the `defSecurityLevel` token in the `snmp.conf` file.

`-m` *miblist*
: Specifies a colon-separated list of MIB modules to load for this application. This overrides the environment variable `MIBS`.

  The special keyword `ALL` is used to specify all modules in all directories when searching for MIB files. Every file whose name does not begin with a period (`.`) will be parsed as if it were a MIB file.

  If the *miblist* has a leading plus sign (+), then the listed MIB modules are loaded in addition to MIB modules specified in the environment variable `MIBS`.

  If a `mibfile` token is specified in the `snmp.conf` file, the `-m MIB` option overrides the `mibfile` token.

`-M` *dirlist*
: Specifies a colon-separated list of directories to search for MIBs. This overrides the environment variable `MIBDIRS`.

  If *dirlist* has a leading plus sign (+), then the given directories are added to the list of MIB directories. Without the leading +, the given directory list overrides the list specified with the environment variable `MIBDIRS`. Note that the directories listed at the end of the list have precedence over directories at the beginning of the list.

  If no value is specified for the environment variable `MIBDIRS`, then the command will still search a default `mib` directory, after it searches the MIB directories specified on the `-M` option. The default directory is `/etc/sma/snmp/mibs`. To avoid having a default `mib` directory searched, set the `MIBDIRS` environment

variable to the empty string (""). Even if the default MIB directory is searched, the directories specified in the `-M` option have precedence in the search order over the default directory.

If the `-M` option is specified and either a `mibfile` or `mibdirs` token is also specified in the `snmp.conf` file, the directories in the `-M` option have precedence in the MIB search order, over the directories set with either the `mibdirs` token and the `mibfile` token.

`-n` *contextName* — Set the destination *contextName* used for SNMPv3 messages. The default *contextName* is the empty string (""). Overrides the `defContext` token in the `snmp.conf` file.

`-O` anEebqQfsSvXTuxUt — Specifies output printing options. See OUTPUT OPTIONS below.

`-P` cdeRuwW — Specifies MIB parsing options. See MIB PARSING OPTIONS below.

`-r` *retries* — Specifies the number of retries to be used in the requests. The default is 5.

`-t` *timeout* — Specifies the timeout in seconds between retries. The default is 1.

`-u` *secName* — Set the `securityName` used for authenticated SNMPv3 messages. Overrides the `defSecurityName` token in the `snmp.conf` file.

`-v` 1 | 2c | 3 — Specifies the protocol version to use: 1 (RFCs 1155-1157), 2c (RFCs 1901-1908), or 3 (RFCs 2571-2574). The default is version 1. This option overrides the `defVersion` token in the `snmp.conf` file.

`-V`, `--version` — Display version information for the application and then exit.

`-x` *privProtocol* — Set the privacy protocol (DES) used for encrypted SNMPv3 messages.

`-X` *privPassword* — Set the privacy pass phrase used for encrypted SNMPv3 messages. Overrides the `defPrivPassphrase` token in the `snmp.conf` file. Note that it is insecure to specify password phrases on the command line. See `snmp.conf(4)`.

`-Z` *boots*,*time* — Set the `engineBoots` and `engineTime` used for authenticated SNMPv3 messages. This will initialize the local notion of the agents *boots*/*time* with an authenticated value stored in the LCD. This value will be discovered if not supplied.

The string *agent* specifies the remote SNMP entity with which to communicate. The format of this parameter is defined in the AGENT SPECIFICATION section below.

Agent Specification

The *agent* specification (see SYNOPSIS) takes the form:

[*transport-specifier*:]*transport-address*

At its simplest, the *agent* specification consists of a hostname or an IPv4 address in the standard, "dotted quad" notation. In this case, communication will be attempted using UDP/IPv4 to port 161 of the given host. Otherwise, the *transport-address* part of the specification is parsed according to the following table:

<transport-specifier>	<transport-address> format
udp	*hostname*[:*port*]

Note that *transport-specifier* strings are case-insensitive so that, for example, "tcp" and "TCP" are equivalent. Here are some examples, along with interpretations:

```
myhost:161
```

Perform query using UDP/IPv4 datagrams sent to `myhost` on port 161. The `:161` is redundant here because that is the default SNMP port.

```
udp:myhost
```

Identical to the previous specification. The `udp:` is redundant here because UDP/IPv4 is the default transport.

MIB Parsing Options

The Net-SNMP MIB parser mostly adheres to the Structure of Management Information (SMI). As that specification has changed through time, and in recognition of the diversity in compliance expressed in MIB files, additional options provide more flexibility in reading MIB files.

`-Pw` Show some warning messages in resolving the MIB files. Can be also set with the configuration token `mibWarningLevel`.

`-PW` Show additional warning messages. Can be also set with the configuration token `mibWarningLevel`.

`-Pe` Show MIB errors. Can be also set with the configuration token `showMibErrors`. An example of an error that would be shown is if an imported module is not found during MIB parsing.

`-Pc` Allow ASN.1 comment to extend to the end of the MIB source line (that is, disallow the use of two dashes (`--`) to terminate comments). This overcomes some problems with manually maintained MIB files. Can be also set with the configuration token `strictCommentTerm`.

`-Pd` Toggles the default of whether or not to save the `DESCRIPTION`s of the MIB objects when parsing. Since the default is to save the `DESCRIPTION`s, specifying `-Pd` causes the `DESCRIPTION`s not to be saved during MIB parsing. For example:

```
snmptranslate -Td -OS -IR system.sysDescr.0
```

will show a description, while:

```
snmptranslate -Td -OS -IR -Pd system.sysDescr.0
```

will not show a description. Collecting the `DESCRIPTION` information into the parsed hierarchy increases the memory used by the size of each `DESCRIPTION` clause.

`-Pu` Allow underline characters in symbols. Can be also set with the configuration token `mibAllowUnderline`.

`-PR` Replace MIB objects using the last read MIB file. The parser replaces MIB objects in its hierarchy whenever it sees a subidentifier and name match.

Caution – Setting this option might result in an incorrect hierarchy. Can be also set with the configuration token `mibReplaceWithLatest`.

Output Options Output display can be controlled by passing various parameters to the `-O` flag. The following examples demonstrate this feature.

The default output displays as follows:

```
snmpget -c public -v 1 localhost system.sysUpTime.0
SNMPv2-MIB::sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
```

`-Oq` Removes the equal sign and type information:

```
system.sysUpTime.0 1:15:09:27.63
```

`-OQ` Removes the type information:

```
system.sysUpTime.0 = 1:15:09:27.63
```

`-Of` Gives you the complete OID:

```
.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0 = \
Timeticks: (14096763) 1 day, 15:09:27.63
```

`-Os` Deletes all but the last symbolic part of the OID:

```
sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
```

`-OS` A variation on `-Os` that adds the name of the MIB that defined the object:

```
SNMPv2-MIB::sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
```

Starting with release 5.0, this is the default output format.

`-Ou` Displays the OID in the UCD-style (inherited from the original CMU code). That means removing a series of "standard" prefixes, if relevant, and breaking down the

OID into the displayable pieces. For example, the OID `vacmSecruityModel.0.3.119.101.115` is broken down by default and the string hidden in the OID is shown. The result would be: `vacmSecurityModel.0."test"`. The `-Ob` option disables this feature.

```
system.sysUpTime.0 = Timeticks: (14096763) 1 day, 15:09:27.63
```

`-On` Displays the OID numerically:

```
.1.3.6.1.2.1.1.3.0 = Timeticks: (14096763) 1 day, 15:09:27.63
```

`-Oe` Removes the symbolic labels from enumerations:

```
snmpget -c public -v 1 localhost ip.ipForwarding.0
ip.ipForwarding.0 = INTEGER: forwarding(1)
snmpget -c public -v 1 -Oe localhost ip.ipForwarding.0
ip.ipForwarding.0 = INTEGER: 1
```

`-Ob` When OIDs contain a index to a table, they are broken into the displayable pieces and shown to you. For example, the OID `vacmSecurityModel.0.3.119.101.115` is nicely broken down by default and the string hidden in the OID is shown to you as `vacmSecurityModel.0."wes"`. The `-Ob` option disables this feature and displays it as `vacmSecurityModel.0.3.119.101.115` once again.

`-OE` Modifies the index strings to include a backslash (`\`) to escape the quotes, to allow them to be reused in shell commands, such as `vacmSecurityModel.0.\"wes\"`

`-OX` Modifies the output of index OIDs to look more "program-like". Square brackets are placed around each index and the `DISPLAY-HINT` information and string conversions are used to format each index. If you take an entry from the `IPV6-MIB::ipv6RouteTable`, it is indexed with an IPv6 address and two integers, and if you are used to IPv6 addresses you know that decimal OIDs are not the preferred notation. Compare:

```
snmpgetnext -OS host IPV6-MIB:ipv6RouteTable
IPV6-MIB::ipv6RouteIfIndex.63.254.1.0.255.0.0.0.0.0.0.0.0.0.0.64.1 \
= INTEGER: 2

snmpgetnext -OSX host IPV6-MIB:ipv6RouteTable
IPV6-MIB::ipv6RouteIfIndex[3ffe:100:ff00:0:0:0:0:0][64][1] = INTEGER: 2
```

`-Oa` If a string-valued object definition does not include a display hint, then the library attempts to determine whether it is an ASCII or binary string, and displays the value accordingly. This flag bypasses this check, and displays all strings as ASCII. Note that this does not affect objects that do have a display hint.

`-Ox` This works similarly to `-Oa`, but displays strings as hexadecimal values.

`-OT` If hexadecimal code is displayed, this will also display any printable characters after the hexadecimal codes.

`-Ov` Output only the variable value, not the OID:

```
snmpget -c public -v 1 -Ov localhost ip.ipForwarding.0
INTEGER: forwarding(1)
```

`-OU` Do not display the `UNITS` suffix at the end of the value.

`-Ot` Output timeticks values as raw numbers:

```
system.sysUpTime.0 = 14096763
```

Note that most of these options can be turned on or off by default by tuning the `snmp.conf` file. See `snmp.conf(4)` for details.

Input Options

The `-I` flag specifies various options that control how your input to the program is parsed. By default, unless one of the following flags is specified, all input parsing methods are used: First the OID is parsed in the normal way, then `-IR` is used, then `-Ib` is used. The use of one of the following flags forces a command to use only one method.

`-IR` Specifies random access lookup, so that if the entire OID path is not specified, it will search for a node in the MIB tree with the given name. Normally, you'd have to specify the `vacmSecurityModel` OID above as:

```
 .iso.org.dod.internet.snmpV2.snmpModules.snmpVacmMIB.vacmMIBObjects. \
vacmSecurityToGroupTable.vacmSecurityToGroupEntry.vacmSecurityModel.0.\
"wes"
```

But the use of the `-IR` flag allows you to shorten that to `vacmSecurityModel.0."wes"`. This OID needs to be quoted to prevent the shell from swallowing the double quotes: But the use of the -IR flag allows you to shorten that to just vacmSecurityModel.0."wes". This OID must be quoted to prevent the shell from swallowing the double quotes: `'vacmSecurityModel.0."wes"'`.

For more information, see the RANDOM ACCESS MIBS section, below.

`-Ib` Indicates that the expression you gave the command is a regular expression that should be used to search for the best match possible in the MIB tree. This would allow you to specify the `vacmSecurityModel` MIB node as something as generic as `vacmsecuritymodel` (since case-insensitive searches are done) or `vacm.\(**model`. Note that multiple matches are obviously possible (`.\(**` matches everything). The best result is calculated as the one that matches the closest to the beginning of the node name and the highest in the tree. A side effect of this option is that you cannot specify indexes or multiple nodes, because the period (`.`) is treated as part of the regular expression.

`-Iu` Use the traditional UCD-style input approach of assuming that OIDs are rooted at the `mib-2` point in the tree (unless they start with an explicit period (`.`)) If random access lookup is in effect (which is the default for most commands), then this will affect only

OIDs specified with a leading numeric subidentifier (and no initial period). Thus an input of `snmpcmd ... 1` would refer to `iso` (from v5.0 onwards) while `snmpcmd -Iu ... 1` would refer to `system`.

`-Ir` By default, indices into tables and values to be assigned to objects are checked against the range and type specified in the MIB. The `-Ir` flag disables this check. This flag is mostly useful when you are testing an agent. For normal operation, it is useful to get your requests checked before they are sent to the remote agent. The diagnostic that the library can provide is also much more precise.

`-Ih` By default, the library will use DISPLAY-HINT information when assigning values. This flag disables this behavior. The result is that, instead of:

```
snmpset localhost HOST-RESOURCES-MIB::hrSystemDate.0 = \
2002-12-10,2:4:6.8
```

you will have to write:

```
snmpset localhost HOST-RESOURCES-MIB::hrSystemData.0 x \
"07 D2 0C 0A 02 04 06 08"
```

Random Access MIBs

In previous releases of the UCD-SNMP package (and if using the `-Iu` option), an object identifier such as `system.sysDescr.0` is looked up in a single "well known" place, built into the SNMP library (or specified by the `P@REFIX` environment variable). The standard place is `.iso.org.dod.internet.mgmt.mib-2`. The identifier can alternatively be a complete object identifier. This is designated by a leading "dot" if using UCD-input style, and is the first thing tried otherwise. To simplify the specification of object identifiers the library supports random access to the identifiers in the MIBs. This is requested by the `-IR` option to the SNMP applications. Additionally, `-Os` prints OIDs in this manner. Using this, `system.sysDescr.0` can also be entered as `sysDescr.0`.

To search only a single MIB for the identifier (if it appears in more than one), specify it as `SNMPv2-MIB::sysDescr.0`. Use `-OS` to print output OIDs in this manner; this is the default since v5.0. This notation also ensures that the specified MIB is loaded, that is, it need not be mentioned in the `-m` option (or `MIBS` environment variable).

Environment Variables

`P@REFIX` The standard prefix for object identifiers (if using UCD-style output). Defaults to `.iso.org.dod.internet.mgmt.mib-2`.

`MIBS` The list of MIBs to load. Defaults to:

```
SNMPv2-TC:SNMPv2-MIB:IF-MIB:IP-MIB:TCP-MIB:UDP-MIB:SNMP-VACM-MIB
```

Overridden by the `-m` option.

`MIBDIRS` The list of directories to search for MIBs. Defaults to `/etc/sma/snmp/mibs`. Overridden by the `-m` option.

Files

`/etc/sma/snmp/snmpd.conf`	Agent configuration file. See `snmpd.conf(4)`.
`~/.snmp/snmp.conf`	
`~/.snmp/snmp.conf`	Application configuration files. See `snmp.conf(4)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also `snmpbulkwalk(1M)`, `snmpbulkwalk(1M)`, `snmpdf(1M)`, `snmpget(1M)`, `snmpgetnext(1M)`, `snmpnetstat(1M)`, `snmpset(1M)`, `snmptrap(1M)`, `snmpusm(1M)`, `snmpvacm(1M)`, `snmpwalk(1M)`, `snmp.conf(4)`, `snmpd.conf(4)`, `attributes(5)`

Name snmpconf – creates and modifies SNMP configuration files

Synopsis

```
snmpconf

snmpconf -g basic_setup

snmpconf [options] [file_to_create]
```

Description The `snmpconf` utility is a simple script that walks you through setting up a configuration file, step-by-step. It works by asking you a series of questions. It creates the configuration file based on your responses.

In its default mode of operation, `snmpconf` prompts you with menus showing sections of the various configuration files it knows about. When you selects a section, a submenu is shown listing the descriptions of the tokens that can be created in that section. When a description is selected, you are prompted with questions that determine the specification of the selected token.

When you quit `snmpconf`, any configuration files that have been edited are saved to the local directory. `snmpconf` supplies comments in the configuration files for each change.

A particularly useful option is the `-g` switch, which walks you through a specific set of configuration questions. For an example, invoke:

```
# snmpconf -g basic_setup
```

This command walks you through an initial setup of the `snmpd` daemon.

Options The following options are supported:

`-f` Force overwriting existing files in the current directory without prompting the user.

`-i` When finished, install the files in the location where the global system commands expect to find them.

`-p` When finished, install the files into the user's home directory's `.snmp` subdirectory. Applications will search for configuration files in this location.

`-I` *directory* When finished, install the files into the directory *directory*.

`-a` Do not issue queries. Read in the various known configuration files and write them back out again. This has the effect of "auto-commenting" the configuration files for you.

`-r` all | none Read in either all or none of the found configuration files. Normally, `snmpconf` prompts you for which files you wish to read in.

`-R` *file,...* Read in a specific list of configuration files.

`-g` *groupname* Groups of configuration entries can be created that can be used to walk a user through a series of questions to create an initial configuration file. There are no menus to navigate, just a list of questions. The command:

```
# snmpconf -g basic_setup
```

provides a good example.

`-G` List all the known groups.

`-c` *configdir* `snmpconf` uses a directory of configuration information to learn about the files and questions that it should be asking. This option tells the utility to use a different location for configuring itself.

`-q` Run slightly more quietly than the default. Because this is an interactive program, this option is not recommended. It removes information from the output that might be helpful to you.

`-d` Turn on copious debugging output.

`-D` Add more (beyond `-d`) debugging output in the form of Perl variable dumps.

Examples

EXAMPLE 1 Adding Comments to `snmpd.conf`

The following command reads in an `snmpd.conf` file and adds comments describing what each token does.

```
# snmpconf -R /etc/sma/snmp/snmpd.conf -a -f snmpd.conf
```

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message displays.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also

`snmpd(1M)`, `snmpd.conf(4)`, `snmp_config(4)`, `attributes(5)`

Name snmpd – daemon to respond to SNMP request packets

Synopsis `/usr/sfw/sbin/snmpd [`*options*`] [`*listening addresses*`]`

Description The `snmpd` daemon is an SNMP agent that binds to a port and awaits requests from SNMP management software. Upon receiving a request, it processes the request(s), collects the requested information, performs any requested operation(s), and, finally, returns information to the requester.

Options The following options are supported:

`-a` Log the source addresses of incoming requests.

`-A` Append to the log file rather than truncating it.

`-c` *file* Read *file* as a configuration file.

`-C` Do not read any configuration files except the one optionally specified by the `-c` option.

Note that this behavior also covers the persistent configuration files. This can result in dynamically-assigned values being reset following an agent restart, unless the relevant persistent configuration files are explicitly loaded using the `-c` option.

`-d` Dump (in hexadecimal) the sent and received SNMP packets.

`-D`[*token*[,...]] Turn on debugging output for the given *token*(s). Without any tokens specified, this option defaults to printing all of the tokens (which is equivalent to the keyword `ALL`). Use `ALL` for extremely verbose output. Note that you must not put a space between the `-D` flag and the listed *token*s.

`-f` Do not `fork()` from the calling shell.

`-g` *GID* Change to the numerical group ID *GID* after opening listening sockets.

`-h`, `--help` Display a brief usage message and then exit.

`-H` Display a list of configuration file directives understood by the agent and then exit.

`-I -`*initlist* This option specifies which modules you do (or do not) want to be initialized when the agent starts up. If the comma-separated *initlist* is preceded with an hyphen (`-`), it is the list of modules that you do not want to be started. Otherwise, *initlist* is the list of modules to be started.

To obtain a list of compiled modules, run the agent with the arguments `-Dmib_init -H` This command assumes you have debugging support compiled in.

`-l` [*file*]	Log all output from the agent (including stdout and stderr) to *file*. If no filename is given, log to a default file set at compile time, normally `/var/log/snmpd.log`.
`-L`	Do not open a log file. Send all messages to stderr instead.
`-P` *file*	Save the process ID of the daemon in *file*.
`-q`	Print simpler output for easier automated parsing.
`-r`	Do not require root access to run the daemon. Specifically, do not exit if files accessible only to root (such as `/dev/kmem`) cannot be opened.
`-s`	Use syslog for logging. See `syslogd(1M)`
`-S` d[0–7]	Specifies the `syslog` facility to use when logging to `syslog`. `d` means `LOG_DAEMON` and the integers 0 through 7 refer to `LOG_LOCAL0` through `LOG_LOCAL7`. `LOG_DAEMON` is the default.
`-u` *UID*	Change to the user ID *UID* (which can be given in numerical or text form) after opening listening sockets.
`-v --version`	Display version information for the agent and then exit.
`-V`	Symbolically dump SNMP transactions.
`-x` *address*	Listens for `AgentX` connections on *address* rather than on the default `/var/agentx/master`. The address can either be a Unix domain socket path or the address of a network interface. The format is the same as the format of listening addresses described below. Note that it is a possible security risk to expose the master agent listening address through TCP/UDP. See section 9 of RFC 2741 for more details.
`-X`	Run as an `AgentX` subagent rather than as an SNMP master agent.

Listening Addresses

By default, `snmpd` listens for incoming SNMP requests only on UDP port 161. However, it is possible to modify this behavior by specifying one or more listening addresses as arguments to the daemon. A listening address takes the form:

`[`<*transport-specifier*>`:]`<*transport-address*>

At its simplest, a listening address can consist of only a port number, in which case `snmpd` listens on that UDP port on all IPv4 interfaces. Otherwise, the <*transport-address*> part of the specification is parsed according to the following table:

<*transport-specifier*>	<*transport-address*> **format**
`udp`	*hostname*[:*port*] or *IPv4-address*[:*port*]
`tcp`	*hostname*[:*port*] or *IPv4-address*[:*port*]

<*transport-specifier*>	<*transport-address*> **format**
`unix`	*pathname*

Currently transports TCP/UDP over IPv4/IPv6 and unix domain sockets. Note that <*transport-specifier*> strings are case-insensitive so that, for example, `tcp` and `TCP` are equivalent. Below are some examples, with accompanying explanations.

`127.0.0.1:161`	Listen on UDP port 161, but only on the loopback interface. This prevents `snmpd` from being queried remotely. The `:161` is redundant because that is the default SNMP port.
`TCP:1161`	Listen on TCP port 1161 on all IPv4 interfaces.
`unix:/tmp/local-agent`	Listen on the Unix domain socket `/tmp/local-agent`.
`/tmp/local-agent`	Identical to the previous specification, because the Unix domain is the default transport if and only if the first character of <*transport-address*> is a slash (`/`).
`udp6:10161`	Listen on port 10161 on all IPv6 interfaces.

Note that not all the transport domains listed above will always be available. For example, hosts with no IPv6 support will not be able to use `udp6` transport addresses, and attempts to do so will result in the error "Error opening specified endpoint".

Files `snmpd` checks for the existence of and parses the following files:

`snmp.conf`	Common configuration for the agent and applications. See `snmp.conf(4)` for details.
`snmpd.local.conf`	Agent-specific configuration. See `snmp.conf(4)` for details. These files are optional and can be used to configure access control, trap generation, subagent protocols, and other features. In addition to these two configuration files, the agent will read any files with the names `snmpd.conf` and `snmpd.local.conf` in a colon-separated path specified in the `SNMPCONFPATH` environment variable, the default location upon agent startup are `/etc/sma/snmp` and `/usr/local/share/snmp`.
`/etc/sma/snmp/mibs`	The agent loads all files in this directory as MIBs. It does not, however, load any file that begins with a dot (`.`) or descend into subdirectories.

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmagt
Interface Stability	Stable

See Also svcadm(1M), svccfg(1M), snmp.conf(4), attributes(5), smf(5)

Notes In addition to basic privileges, to run successfully, the agent requires PRIV_NET_PRIVADDR. See privileges(5).

The snmpd service is managed by the service management facility, smf(5), under the service identifiers:

```
svc:/application/management/sma
svc:/application/management/seaport
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

The service uses the solaris.smf.manage.sma privilege. If /etc/sma/snmp/snmpd.conf contains DISABLE=YES, then the service does not start and displays the message:

```
snmpd disabled by config file /etc/sma/snmp/snmpd.conf
```

Name snmpdelta – monitor deltas of integer valued SNMP variables

Synopsis

```
/usr/sfw/bin/snmpdelta [common options] [-Cf] [-Ct] [-Cs]
     [-CS] [-Cm] [-CF configfile] [-Cl] [-Cp period]
     [-CP peaks] [-Ck] [-CT] [-Cv vars/pkt] agent OID
     [OID]...
```

Description The `snmpdelta` command monitors the specified integer-valued OIDs and reports changes over time.

The operand *agent* identifies a target SNMP agent, which is instrumented to monitor a given set of objects. At its simplest, the *agent* specification will consist of a hostname or an IPv4 address. With such an operand, the command attempts communication with the agent, using UDP/IPv4 to port 161 of the given target host. See `snmpcmd(1M)` for a full list of the possible formats for *agent*.

The operand *OID* is an object identifier that uniquely identifies the object type within a MIB. Multiple OIDs can be specified in a single `snmpdelta` command.

Options See `snmpcmd(1M)` for a list of *common options*. In addition to the *common options*, `snmpdelta` supports the options described below.

`-Cf` Do not fix errors and then retry the request. Without this option, if multiple OIDs have been specified for a single request and if the request for one or more of the OIDs fails, `snmpdelta` will retry the request so that data for OIDs apart from the ones that failed will still be returned. Specifying `-Cf` tells `snmpdelta` not to retry a request, even if there are multiple OIDs specified.

`-Ct` Determines time interval from the monitored entity.

`-Cs` Displays a timestamp.

`-CS` Generates a "sum count" in addition to the individual instance counts. The "sum count" is the total of all the individual deltas for each time period.

`-Cm` Displays the maximum value ever attained.

`-CF` *configfile* Tells `snmpdelta` to read its configuration from the specified file. This option allows the input to be set up in advance rather than having to be specified on the command line.

`-Cl` Tells snmpdelta to write its configuration to files whose names correspond to the MIB instances monitored. For example:

```
% snmpdelta -c public -v 1 -Cl localhost ifInOctets.1
```

...will create a file `localhost-ifInOctets.1`.

`-Cp` Specifies the number of seconds between polling periods. Polling involves sending a request to the agent. The default polling period is one second.

`-CP` *peaks* Specifies the reporting period in number of polling periods. If this option is specified, `snmpdelta` polls the agent *peaks* number of times before reporting the results. The result reported includes the average value over the reporting period. In addition, the highest polled value within the reporting period is shown.

`-Ck` When the polling period (`-Cp`) is an increment of 60 seconds and the timestamp is displayed in the output (`-Cs`), then the default display shows the timestamp in the format *hh:mm mm/dd*. This option causes the timestamp format to be *hh:mm:ss mm/dd* (adding seconds).

`-CT` Display output in tabular form.

`-Cv` *vars/pkt* Specifies the maximum number of OIDs allowed to be packaged in a single PDU. Multiple PDUs can be created in a single request. The default value of variables per packet is 60. This option is useful if a request response results in an error because the request packet is too big.

Examples

EXAMPLE 1 Obtaining Timestamped Output

The following command uses the `-Cs` option to timestamp output. This example assumes that there are at least three entries in your `ifTable`.

```
% snmpdelta -c public -v 1 -Cs localhost \
IF-MIB::ifInUcastPkts.3 IF-MIB::ifOutUcastPkts.3

[20:15:43 6/14] ifInUcastPkts.3 /1 sec: 158
[20:15:43 6/14] ifOutUcastPkts.3 /1 sec: 158
[20:15:44 6/14] ifInUcastPkts.3 /1 sec: 184
[20:15:44 6/14] ifOutUcastPkts.3 /1 sec: 184
[20:15:45 6/14] ifInUcastPkts.3 /1 sec: 184
[20:15:45 6/14] ifOutUcastPkts.3 /1 sec: 184
[20:15:46 6/14] ifInUcastPkts.3 /1 sec: 158
[20:15:46 6/14] ifOutUcastPkts.3 /1 sec: 158
[20:15:47 6/14] ifInUcastPkts.3 /1 sec: 184
[20:15:47 6/14] ifOutUcastPkts.3 /1 sec: 184
[20:15:48 6/14] ifInUcastPkts.3 /1 sec: 184
[20:15:48 6/14] ifOutUcastPkts.3 /1 sec: 184
[20:15:49 6/14] ifInUcastPkts.3 /1 sec: 158
[20:15:49 6/14] ifOutUcastPkts.3 /1 sec: 158
^C
```

EXAMPLE 2 Displaying Output in Tabular Form

The following command uses the `-CT` option to format output as a table. This example assumes that there are at least three entries in your `ifTable`.

```
% snmpdelta -c public -v 1 -Cs -CT localhost \
IF-MIB:ifInUcastPkts.3 IF-MIB:ifOutcastPkts.3 \
```

EXAMPLE 2 Displaying Output in Tabular Form *(Continued)*

```
localhost ifInUcastPkts.3 ifOutUcastPkts.3

[20:15:59 6/14] 184.00 184.00
[20:16:00 6/14] 158.00 158.00
[20:16:01 6/14] 184.00 184.00
[20:16:02 6/14] 184.00 184.00
[20:16:03 6/14] 158.00 158.00
[20:16:04 6/14] 184.00 184.00
[20:16:05 6/14] 184.00 184.00
[20:16:06 6/14] 158.00 158.00
^C
```

EXAMPLE 3 Sending Output to a File

The following example uses a number of options. This example assumes that there are at least four entries in your `ifTable`. Because the `-Cl` option is specified, the output is sent to a file and not to the screen.

```
% snmpdelta -c public -v 1 -Ct -Cs -CS -Cm -Cl -Cp 60 -CP 60 \
interlink.sw.net.cmu.edu .1.3.6.1.2.1.2.2.1.16.3 \
.1.3.6.1.2.1.2.2.1.16.4
```

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors and for cases where an SNMP client session could not be opened.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also `snmpcmd(1M)`, `snmp_variables(4)`, `attributes(5)`

Name snmpdf – get a listing of disk space usage on a remote machine by means of SNMP

Synopsis

```
/usr/sfw/bin/snmpdf [common options] [-Cu] agent
```

Description The `snmpdf` command is a networked verison of the `df(1M)` command. It checks the disk space on the remote machine by examining the `HOST-RESOURCES-MIB`'s `hrStorageTable` or the `UCD-SNMP-MIB`'s `dskTable`. By default, the `hrStorageTable` is preferred, as it typically contains more information than the `dskTable`. However, the `-Cu` argument can be passed to `snmpdf` to force the usage of `dskTable`.

The *agent* operand identifies a target SNMP agent, which is instrumented to monitor specified objects. At its simplest, the *agent* specification consists of a host name or an IPv4 address. In this situation, the command attempts communication with the agent using UDP/IPv4 to port 161 of the target host.

See the `snmpcmd(1M)` manual page for a full list of the possible formats for *agent*.

See the `snmpd.conf(4)` manual page for guidance on setting up `dskTable` using the disk directive in the `snmpd.conf` file.

Options The following options are supported:

common options — See `snmpcmd(1M)` for a list of possible values for *common options*, as well as their descriptions.

`-Cu` — Forces the command to use `dskTable` in `UCD-SNMP-MIB` instead of the default to determine the storage information. Generally, the default use of `hrStorageTable` in `HOST-RESOURCES-MIB` is preferred because it usually contains more information than `dskTable`.

Examples **EXAMPLE 1** Obtaining Disk Usage of a Remote System

The following command returns a display of the disk usage of a remote system.

```
Description size (kB) Used     Available Used%
/           7524587   2186910  5337677   29%
/proc       0         0        0          0%
/etc/mnttab 0         0        0          0%
/var/run    1223088   32       1223056    0%
/tmp        1289904   66848    1223056    5%
/cache      124330    2416     121914     1%
/vol        0         0        0          0%
Real Memory 524288    447456   76832     85%
Swap Space  1420296   195192   1225104   13%
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors.

`2` An error occurred while executing the command. An error message is displayed.

See Also `df(1M)`, `snmpcmd(1M)`, `snmp.conf(4)`, `snmpd.conf(4)`, `attributes(5)`

Name snmpdx – Sun Solstice Enterprise Master Agent

Synopsis

```
/usr/lib/snmp/snmpdx [-hy] [-a filename] [-c config-dir]
     [-d debug-level] [-i filename] [-m GROUP -m SPLIT]
     [-o filename] [-p port] [-r filename]
```

Description The Master Agent, `snmpdx`, is the main component of Solstice Enterprise Agent (SEA) technology. It runs as a daemon process and listens to User Datagram Protocol (UDP) port 161 for SNMP requests. The Master Agent also opens another port to receive SNMP trap notifications from various subagents. These traps are forwarded to various managers, as determined by the configuration file.

Upon invocation, `snmpdx` reads its various configuration files and takes appropriate actions by activating subagents, determining the subtree Object Identifier (OID) for various subagents, populating its own Management Information Bases (MIBs), and so forth. The Master Agent invokes subagents, registers subagents, sends requests to subagents, receives responses from subagents, and traps notifications from subagents.

The Master Agent is invoked by the service management facility `smf(5)` at boot time if `svc:/application/management/snmpdx` is enabled (see NOTES) and contents of the resource configuration file `/etc/snmp/conf/snmpdx.rsrc` are non-trivial.

Note – The SMA (Systems Management Agent) is the default SNMP agent in the Solaris operating system. See `netsnmp(5)`. `snmpdx` is Obsolete and may not be supported in a future release of Solaris.

Options The following options are supported:

`-a`*filename*	Specify the full path of the access control file used by the Master Agent. The default access control file is `/etc/snmp/conf/snmpdx.acl`.
`-c`*config-dir*	Specify the full path of the directory containing the Master Agent configuration files. The default directory is `/etc/snmp/conf`.
`-d`*debug-level*	Debug. Levels from 0 to 4 are supported, giving various levels of debug information. The default is 0 which means no debug information is given.
`-h`	Help. Print the command line usage.
`-i`*filename*	Specify the full path of the enterprise-name OID map. This file contains the PID used by the Master Agent for recovery after a crash. It contains tuples of the UNIX process ID, port number, resource name, and agent name. The default file is `/var/snmp/snmpdx.st`.
`-m GROUP | -m SPLIT`	Specify the mode to use for forwarding of SNMP requests.

`GROUP` Multiple variables can be included in each request from the Master Agent to the subagents. This results in, at mose, one send-request per agent.

`SPLIT` Each variable in the incoming request results in one send-request to each subagent.

The default is `GROUP`.

`-o`*filename* Specify the full path of the file containing the tuple (enterprise-name, OID). For example, (Sun Microsystems, 1.3.1.6.1.4.32). The Master Agent uses this file as a base for look-up in the trap-filtering and forwarding process. The default file is `/etc/snmp/conf/enterprises.oid`.

`-p`*port* Specify the port number. The default port number is 161.

`-r`*filename* Specify the full path of the resource file to be used by the Master Agent. This file stores information about the subagents that the Master Agent invokes and manages. The default resource file is `/etc/snmp/conf/snmpdx.rsrc`.

`-y` Set a recovery indicator to invoke the recovery module. The recovery process discovers which subagents in the previous session are still active; those subagents not active are re-spawned by the Master Agent.

Files

`/etc/snmp/conf/enterprises.oid`	Enterprise-name OID map
`/etc/snmp/conf/snmpdx.acl`	Access control file
`/etc/snmp/conf/snmpdx.rsrc`	Resource configuration file
`/var/snmp/snmpdx.st`	Master Agent status file
`/var/snmp/mib/snmpdx.mib`	Master Agent MIB file

Exit Status The following error values are returned:

`0`	Successful completion.
non-zero	An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsasnm
Interface Stability	Obsolete

See Also attributes(5), netsnmp(5), smf(5)

Notes The snmpdx service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/application/management/snmpdx
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name snmpget – communicate with a network entity using SNMP GET requests

Synopsis `/usr/sfw/bin/snmpget [`*common options*`] [-Cf]` *oid* `[`*oid*`]...`

Description The `snmpget` utility is an SNMP application that uses the SNMP `GET` request to query for information on a network entity. You can specify one or more object identifiers (OIDs) as arguments on the command line. Each variable name must be specified in the format specified in `snmp_variables(4)`.

For example, the command:

```
# snmpget -c public zeus system.sysDescr.0
```

retrieves the variable `system.sysDescr.0`:

```
system.sysDescr.0 = "SunOS zeus.net.cmu.edu 4.1.3_U1 1 sun4m"
```

If the network entity has an error processing the request packet, an error packet is returned and a message displayed. The message indicates the way in which the request was malformed. If there were other variables in the request that were correctly formed, the request will be resent without the bad variable.

Options The following option is supported:

`-Cf` If `-Cf` is not specified, some applications (including `snmpgetnext(1M)` and `snmpget`) attempt to fix errors returned by the agent that you were talking to and resend the request. The `-Cf` option suppresses this fix-and-resend feature.

Fix-and-resend is useful if you specified a nonexistent OID in your request and you are using SNMPv1, which requires "all or nothing" types of requests. In the following example note that `system.sysUpTime` is an incomplete OID, because it requires the `.0` index appended to it:

```
# snmpget -v1 -Cf -c public localhost system.sysUpTime \
system.sysContact.0
Error in packet
Reason: (noSuchName) There is no such variable name in this MIB.
This name doesn't exist: system.sysUpTime

# snmpget -v1 -c public localhost system.sysUpTime system.sysContact.0
Error in packet
Reason: (noSuchName) There is no such variable name in this MIB.
This name doesn't exist: system.sysUpTime

system.sysContact.0 = STRING: root@localhost
```

In addition to this option, `snmpwalk` takes the common options described in the `snmpcmd(1M)` manual page.

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors.

`2` An error occurred while executing the command. An error message is displayed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also `snmpcmd(1M)`, `snmp_variables(4)`, `attributes(5)`

Name snmpgetnext – communicate with a network entity using SNMP GETNEXT requests

Synopsis

```
snmpgetnext [-Cf] [common options] oid [oid]...
```

Description The `snmpgetnext` utility is an SNMP application that uses the SNMP `GETNEXT` request to query for information on a network entity. You can specify one or more object identifiers (OIDs) as arguments on the command line. Each variable name must be specified in the format specified in `snmp_variables(4)` . For each variable, the one that is lexicographically "next" in the remote entity's MIB is returned.

For example, the command:

```
# snmpgetnext -c public zeus interfaces.ifTable.ifEntry.ifType.1
```

retrieves the variable `interfaces.ifTable.ifEntry.ifType.2`:

```
interfaces.ifTable.ifEntry.ifType.2 = softwareLoopback(24)
```

If the network entity has an error processing the request packet, an error packet is returned and a message displayed. The message indicates the way in which the request was malformed.

Options The following options are supported:

`-Cf` If `-Cf` is not specified, some applications (including `snmpget(1M)` and `snmpgetnext`) attempt to fix errors returned by the agent that you were talking to and resend the request. The `-Cf` option suppresses this fix-and-resend feature.

Fix-and-resend is useful if you specified a nonexistent OID in your request and you are using SNMPv1, which requires "all or nothing" types of requests.

In addition to this option, `snmpgetnext` takes the common options described in the `snmpcmd(1M)` manual page.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors.

`2` An error occurred while executing the command. An error message is displayed.

See Also `snmpcmd(1M)`, `snmpget(1M)`, `snmp_variables(4)`, `attributes(5)`

Name snmpnetstat – show network status using SNMP

Synopsis

```
/usr/sfw/bin/snmpnetstat [common options] [-a] [-n] agent

/usr/sfw/sma_snmp/bin/snmpnetstat [common options] [-iorns] agent

/usr/sfw/sma_snmp/bin/snmpnetstat [common options] [-in]
     [-I interface] agent [interval]

/usr/sfw/sma_snmp/bin/snmpnetstat [common options] [-an]
     [-s] [-P protocol] agent
```

Description The `snmpnetstat` command symbolically displays the values of various network-related information retrieved from a remote system using the SNMP protocol. There are a number of output formats, depending on the options for the information presented. Referring to the SYNOPSIS, above:

- The first form of the command displays a list of active sockets.
- The second form presents the values of other network-related information according to the option selected.
- The third form, with an interval specified, continuously displays the information regarding packet traffic on the configured network interfaces.
- The fourth form displays statistics about the named protocol.

The operand *agent* identifies a target SNMP agent that is instrumented to monitor the given objects. At its simplest, the *agent* specification consists of a host name or an IPv4 address. In this situation, the command attempts communication with the agent using UDP/IPv4 to port 161 of the target host. See `snmpcmd(1M)` for a full list of the possible formats for *agent*.

The version 1 and version 2c community specifies the community name for the transaction with the remote system.

Options The following options are supported:

common options	See `snmpcmd(1M)` for a list of possible values for common options, as well as their descriptions.
`-a`	With the default display, show the state of all sockets. Normally, sockets used by server processes are not shown.
`-i`	Show the state of all of the network interfaces. The interface display provides a table of cumulative statistics regarding packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit (MTU) are also displayed.
`-o`	Show an abbreviated interface status, giving octets in place of packets. This is useful when observing virtual interfaces (such as Frame Relay circuits) on a router.

`-I` *interface*	Show information only about this interface; used with an interval as described below.
`-n`	Show network addresses as numbers. (Normally, `snmpnetstat` interprets addresses and attempts to display them symbolically). This option can be used with any of the display formats.
`-P` *protocol*	Show statistics about *protocol*, which is either a well-known name for a protocol or an alias for it. Some protocol names and aliases are listed in the file `/etc/protocols` or in a naming service. A null response typically means that there are no interesting numbers to report. The program will complain if *protocol* is unknown or if there is no statistics routine for it.
`-s`	Show per-protocol statistics. When used with the `-r` option, show routing statistics instead.
`-r`	Show the routing tables. When `-s` is also present, show per-protocol routing statistics instead of the routing tables.
interval	When `snmpnetstat` is invoked with an interval argument, it displays a running count of statistics related to network interfaces. *interval* is the number of seconds between reporting of statistics.

`snmpnetstat` supports the following types of display:

active sockets display (default)	The default display, for active sockets, shows the local and remote addresses, protocol, and the internal state of the protocol. Address formats are of the form *host.port* or *network.port* if a socket's address specifies a network but no specific host address. When known, the host and network addresses are displayed symbolically according to `/etc/hosts` and `/etc/networks`, respectively. If a symbolic name for an address is unknown, or if the `-n` option is specified, the address is printed numerically, according to the address family. For more information regarding the Internet "dot format," refer to `inet(3SOCKET)`. Unspecified, or wildcard, addresses and ports appear as "`\(**`".
interface display	The interface display provides a table of cumulative statistics regarding packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit (MTU) are also displayed.

routing table display	The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets. The flags field shows the state of the route (`U` if the route is up), whether the route is to a gateway (G), whether the route was created dynamically by a redirect (D), and whether the route has been modified by a redirect (M). Direct routes are created for each interface attached to the local host. The gateway field for such entries shows the address of the outgoing interface. The interface entry indicates the network interface used for the route.
interface display with an interval	When `snmpnetstat` is invoked with an interval argument, it displays a running count of statistics related to network interfaces. This display consists of a column for the primary interface and a column summarizing information for all interfaces. The primary interface can be replaced with another interface with the `-I` option. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.
active sockets display for a single protocol	When a protocol is specified with the `-P` option, the information displayed is similar to that in the default display for active sockets, except the display is limited to the given protocol.

Note that figures `snmpnetstat` reports in the `Ipkts` column (part of the interface display) might differ from figures in the `Ipkts` column in `netstat(1M)`. `snmpnetstat` displays a total of unicast, multicast, and broadcast packets. `netstat` omits broadcast packets from its total.

Examples

EXAMPLE 1 Displaying Active Sockets

The following is an example of `snmpnetstat`'s default display, which is to display active sockets.

```
% snmpnetstat -v 2c -c public -a testhost

Active Internet (tcp) Connections (including servers)
Proto Local Address Foreign Address (state)
```

EXAMPLE 1 Displaying Active Sockets *(Continued)*

```
tcp   *.echo        *.*             LISTEN
tcp   *.discard     *.*             LISTEN
tcp   *.daytime     *.*             LISTEN
tcp   *.chargen     *.*             LISTEN
tcp   *.ftp         *.*             LISTEN
tcp   *.telnet      *.*             LISTEN
tcp   *.smtp        *.*             LISTEN

Active Internet (udp) Connections
Proto Local Address
udp   *.echo
udp   *.discard
udp   *.daytime
udp   *.chargen
udp   *.time

% snmpnetstat -v 2c -c public -i testhost

Name Mtu  Network   Address  Ipkts     Ierrs  Opkts  Oerrs Queue
eri0 1500 10.6.9/24 testhost 170548881 245601 687976 0     0
lo0  8232 127      localhost 7530982   0      7530982 0    0
```

EXAMPLE 2 Displaying Statistics for a Specific Protocol

The following example shows how snmpnetstat displays statistics for a specific protocol.

```
% snmpnetstat -v 2c -c public -P tcp testhost

Active Internet (tcp) Connections
Proto Local Address Foreign Address (state)
tcp   *.echo        *.*             LISTEN
tcp   *.discard     *.*             LISTEN
tcp   *.daytime     *.*             LISTEN
tcp   *.chargen     *.*             LISTEN
tcp   *.ftp         *.*             LISTEN
tcp   *.telnet      *.*             LISTEN
tcp   *.smtp        *.*             LISTEN
```

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also iostat(1M), netstat(1M), snmpcmd(1M), vmstat(1M), attributes(5)

Name snmpset – communicate with a network entity using SNMP SET requests

Synopsis

```
snmpset [common options] oid type value [oid type value]...
```

Description The `snmpset` utility is an SNMP application that uses the SNMP `SET` request to set information on a network entity. A type and a value must accompany each object identifier. Each variable name must be entered in the format specified in `snmp_variables(4)`.

The *type* is a single character, one of:

```
i INTEGER
u UNSIGNED
c COUNTER32
s STRING
x HEX STRING
d DECIMAL STRING
n NULLOBJ
o OBJID
t TIMETICKS
a IPADDRESS
b BITS
```

If you have the proper MIB file loaded, you can, in most cases, replace the type with an equal sign (=). For an object of type `OCTET STRING` this assumes a string such as the "s"-type notation. For other types, `snmpset` interprets the data in the way you would expect.

For example, the command:

```
snmpset -c private -v 1 test-hub system.sysContact.0 s dpz@noc.rutgers.edu
ip.ipforwarding.0 = 2
```

sets the variables `sysContact.0` and `ipForwarding.0`, as follows:

```
system.sysContact.0 = STRING: "dpz@noc.rutgers.edu"
ip.ipForwarding.0 = INTEGER: not-forwarding(2)
```

If the network entity has an error processing the request packet, an error packet is returned and a message displayed, indicating the way in which the request was malformed.

Options `snmpset` takes the common options described in the `snmpcmd(1M)` manual page.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

Exit Status

`0`	Successful completion.
`1`	A usage syntax error. A usage message is displayed. Also used for timeout errors.
`2`	An error occurred while executing the command. An error message is displayed.

See Also `snmpcmd(1M)`, `snmp_variables(4)`, `attributes(5)`

Name snmptable – obtain and display an SNMP table

Synopsis

```
/usr/sfw/bin/snmptable [common options] [-Cb] [-CB] [-Cb]
     [-Ch] [-CH] [-Ci] [-Cf string] [-Cw width] [agent]
     [table-oid]
```

Description The `snmptable` command is an SNMP application that repeatedly uses the SNMP `GETNEXT` or `GETBULK` requests to query for information on a network entity. The operand *table-oid* must specify an SNMP table. Both numeric and symbolic (string) OIDs are supported.

The operand *agent* identifies a target SNMP agent that is instrumented to monitor manageable objects. At its simplest, the *agent* specification consists of a hostname or an IPv4 address. With such an address, the command attempts communication with the agent, using UDP/IPv4 to port 161 of the given target host. See `snmpcmd(1M)` for a full list of the possible formats for *agent*.

Options See `snmpcmd(1M)` for a list of *common options*. In addition to the *common options*, `snmptable` supports the options described below.

`-Cb` Display only a brief heading. Any common prefix of the table field names is not displayed.

`-CB` Use only `GETNEXT`, not `GETBULK`, requests to retrieve data.

`-Cf` *string* The string *string* is used to separate table columns. With this option, each table entry is printed in compact form, using only the specified string to separate the columns. This option can be useful if you intend to import the table into a database. Without this option, the table is displayed in vertically aligned columns.

`-Ch` Display only the column headings.

`-CH` Do not display the column headings or the table name. Only raw data is displayed.

`-Ci` Prepends the index of the entry to all printed lines.

`-Cw` *width* Specifies the width of the lines when the table is printed. If a line is longer than *width*, lines are truncated to fit within *width*.

Examples **EXAMPLE 1** Retrieving an SNMP Table

The following commands retrieve two-column SNMP tables.

```
% snmptable -v 2c -c public localhost at.atTable

SNMP table: at.atTable RFC1213-MIB::atTable
    atIfIndex atPhysAddress atNetAddress
    1 8:0:20:20:0:ab 130.225.243.33
```

EXAMPLE 1 Retrieving an SNMP Table *(Continued)*

```
% snmptable -v 2c -c public -Cf + localhost at.atTable

SNMP table: at.atTable
atIfIndex+atPhysAddress+atNetAddress
        1+8:0:20:20:0:ab+130.225.243.33
```

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors.

`2` An error occurred while executing the command. An error message is displayed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also `snmpcmd(1M)`, `snmp_variables(4)`, `attributes(5)`

Bugs The test for *table-oid* actually specifying a table is not perfect. Note also that the test requires the defining MIB file to be loaded.

Name snmptest – communicate with a network entity using SNMP requests

Synopsis `/usr/sfw/bin/snmptest [`*common options*`] agent`

Description `snmptest` is a flexible SNMP application that can monitor and manage information on a network entity.

Invoking `snmptest` invokes a command line interpreter that accepts commands. This intepreter enables you to send different types of SNMP requests to target agents.

The operand *agent* identifies a target SNMP agent that is instrumented to monitor manageable objects. At its simplest, the *agent* specification consists of a host name or an IPv4 address. In this situation, the command attempts communication with the agent, using UDP/IPv4 to port 161 of the target host. See `snmpcmd(1M)` for a full list of the possible formats for *agent*.

After you invoke `snmptest`, the command line intepreter prompts with:

```
Variable:
```

At this point you can enter one or more variable names, one per line. A blank line ends the parameter input and sends the request (variables entered) in a single packet to the remote entity. Each variable name is given in the format specified in `snmp_variables(4)`. For example, the command:

```
snmptest -c public -v 1 zeus
Variable: system.sysDescr.0
Variable:
```

...returns information about the request and reply packets, as well as the data:

```
requestid 0x5992478A errstat 0x0 errindex 0x0
system.sysDescr.0 = STRING: "Unix 4.3BSD"
```

The `errstatus` value shows the error status code for the call. The possible values for errstat are in the header file `/usr/sfw/include/net-snmp/library/snmp.h`. The `errindex` value identifies the variable that has an error. Index values are assigned to all the variables entered at the `Variable:` prompt. The first value is assigned an index of 1.

Upon startup, the program defaults to sending a `GET` request packet. The type of request can be changed by typing one of the following commands at the `Variable:` prompt:

`$G` Send a `GET` request.

`$N` Send a `GETNEXT` request.

`$S` Send a `SET` request.

`$B` Send a `GETBULK` request. Note that `GETBULK` is not available in SNMPv1.

`$I` Send an inform request.

`$T` Send an SNMPv2 trap request.

Other values that can be entered at the `Variable:` prompt are:

`$D` Toggle the dumping of each sent and received packet.

`$QP` Toggle a quicker, less verbose output form.

`$Q` Quit the program.

The following are valid request types:

`GET` request — When in "`GET` request" mode (`$G` or default), you can enter an OID at the `Variable:` prompt. You can enter multiple OIDs, one per prompt. Enter a blank line to send the `GET` request.

`GETNEXT` request — The "`GETNEXT` request" mode (`$N`) is similar to the "`GET` request" mode, described above.

`SET` request — When in the "`SET` request" mode (`$S`), more information is requested by the prompt for each variable. The prompt:

```
Type [i|s|x|d|n|o|t|a]:
```

...requests the type of the variable be entered. Depending on the type of value you want to set, type one of the following:

- `i` integer
- `u` unsigned integer
- `s` octet string in ASCII
- `x` octet string in hex bytes, separated by whitespace
- `d` octet string as decimal bytes, separated by whitespace
- `a` IP address in dotted IP notation
- `o` object identifier
- `n` null
- `t` timeticks

At this point a value will be prompted for. To enter an integer value, just type the integer (in decimal). If it is a decimal string, type in whitespace-separated decimal numbers, one per byte of the string. Again, enter a blank line at the prompt for the variable name to send the packet.

`GETBULK` request — The "`GETBULK` request" mode (`$B`) is similar to the "`SET` request" mode. Note, however, that `GETBULK` is not available in SNMPv1.

Inform request	The "Inform request" mode ($I) is similar to the "SET request" mode. However, this type of request is not available in SNMPv1. Also, the *agent* specified in the `snmptest` command should correspond to the target `snmptrapd` agent.
SNMPv2 trap request	The "SNMPv2 trap request" mode ($T) is similar to the "SET request" mode. However, this type of request is not available in SNMPv1. Also, the *agent* specified in the `snmptest` command should correspond to the target `snmptrapd` agent.

Options `snmptest` takes the common options described in the `snmpcmd(1M)`.

Examples **EXAMPLE 1** Sending a GET Request for Two OIDs

The following is an example of sending a `GET` request for two OIDs:

```
% snmptest -v 2c -c public testhost:161

Variable: system.sysDescr.0
Variable: system.sysContact.0
Variable:
Received Get Response from 128.2.56.220
requestid 0x7D9FCD63 errstat 0x0 errindex 0x0
SNMPv2-MIB::sysDescr.0 = STRING: SunOS testhost 5.9 Generic_112233-02 sun4u
SNMPv2-MIB::sysContact.0 = STRING: x1111
```

EXAMPLE 2 Sending a GETNEXT Request

The following is an example of sending a `GETNEXT` request:

```
Variable: $N
Request type is Getnext Request
Variable: SNMPv2-MIB::sysORUpTime.1
Variable:
Received Get Response from 128.2.56.220
requestid 0x7D9FCD64 errstat 0x0 errindex 0x0
SNMPv2-MIB::sysORUpTime.2 = Timeticks: (6) 0:00:00.06
Variable:
```

EXAMPLE 3 Sending a SET Request

The following is an example of sending a `SET` request:

```
Variable: $S
Request type is Set Request
Variable: system.sysLocation.0
Type [i|u|s|x|d|n|o|t|a]: s
Value: building 17
Variable:
Received Get Response from 128.2.56.220
```

EXAMPLE 3 Sending a SET Request *(Continued)*

```
requestid 0x7D9FCD65 errstat 0x0 errindex 0x0
SNMPv2-MIB::sysLocation.0 = STRING: building A
Variable:
```

EXAMPLE 4 Sending a GETBULK Request

The following is an example of sending a `GETBULK` request:

```
Variable: $B
Request type is Bulk Request
Enter a blank line to terminate the list of non-repeaters
and to begin the repeating variables
Variable:
Now input the repeating variables
Variable: system.sysContact.0
Variable: system.sysLocation.0
Variable:
What repeat count? 2
Received Get Response from 128.2.56.220
requestid 0x2EA7942A errstat 0x0 errindex 0x0
SNMPv2-MIB::sysName.0 = STRING: testhost
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (58) 0:00:00.58
SNMPv2-MIB::sysLocation.0 = STRING: bldg A
SNMPv2-MIB::sysORID.1 = OID: IF-MIB::ifMIB
Variable:
```

EXAMPLE 5 Sending an Inform Request

The following is an example of sending an Inform request:

```
snmptest -v 2c -c public snmptrapd_host:162
Variable: $I
Request type is Inform Request
(Are you sending to the right port?)
Variable: system.sysContact.0
Type [i|u|sIx|d|n|o|t|a]: s
Value: x12345
Variable:
Inform Acknowledged
Variable:
```

The `snmptrapd_host` will show:

```
snmptrapd_host [<ip address>]: Trap SNMPv2-MIB::sysContact.0 = STRING: x12345
```

EXAMPLE 6 Sending an SNMPv2 Trap Request

The following is an example of sending an SNMPv2 Trap request:

EXAMPLE 6 Sending an SNMPv2 Trap Request *(Continued)*

```
snmptest -v 2c -c public snmptrapd_host:162
Variable: $T
Request type is SNMPv2 Trap Request
(Are you sending to the right port?)
Variable: system.sysLocation.0
Type [i|u|s|x|d|n|o|t|a]: s
Value: building a
Variable:
```

The `snmptrapd_host` will show:

```
snmptrapd_host [<ip address>]: Trap SNMPv2-MIB::sys.0 = STRING:
building A
```

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also

`snmpcmd(1M)`, `snmpget(1M)`, `snmpset(1M)`, `snmp_variables(4)`, `attributes(5)`

Name snmptranslate – translate SNMP OID values into a more useful form

Synopsis

```
/usr/sfw/bin/snmptranslate [-D token...] [-h] [-m miblist]
     [-M dirlist] [-T transopts] [common options] OID
     [OID]...
```

Description `snmptranslate` is an application that translates one or more SNMP object identifier values from their symbolic (text) forms into their numerical forms or vice-versa.

OID is either a numeric or text object identifier.

Options The following options are supported:

`-D` *token*[,...] Turn on debugging output for the specified *token*(s). Use `ALL` for extremely verbose output.

`-h` Display a brief usage message and then exit.

`-m` *miblist* Specifies a colon-separated list of MIB modules to load for this application. This overrides the environment variable `MIBS`.

The special keyword `ALL` is used to specify all modules in all directories when searching for MIB files. Every file whose name does not begin with "." is parsed as if it were a MIB file.

`-M` *dirlist* Specifies a colon-separated list of directories to search for MIBs. This overrides the environment variable `MIBDIRS`.

`-T` *transopts* Provides control over the translation of the OID values. The following *transopts* are available:

- `-Td` Display full details of the specified OID.
- `-Tp` Display a graphical tree, rooted at the specified OID.
- `-Ta` Dump the loaded MIB in a trivial form.
- `-Tl` Dump a labeled form of all objects.
- `-To` Dump a numeric form of all objects.
- `-Ts` Dump a symbolic form of all objects.
- `-Tt` Dump a tree form of the loaded MIBs (mostly useful for debugging).
- `-V` Display version information for the application and then exit.
- `-w` *width* Specifies the width of `-Tp` and `-Td` output. The default is very large.

In addition to the preceding options, `snmptranslate` takes the OID input (`-I`), MIB parsing (`-M`) and OID output (`-O`) options described in the `INPUT OPTIONS`, `MIB PARSING OPTIONS` and `OUTPUT OPTIONS` sections of snmpcmd(1M).

Examples

EXAMPLE 1 Expanding `sysDescr`

The following command translates `sysDescr` to a more qualified form:

```
% snmptranslate -On -IR sysDescr

.1.3.6.1.2.1.1.1
```

The following command does further translation of `sysDescr`:

```
% snmptranslate -Onf -IR sysDescr

.iso.org.dod.internet.mgmt.mib-2.system.sysDescr
```

Again, the following command does further translates `sysDescr`:

```
% snmptranslate -Td -IR -OS system.sysDescr

SNMPv2-MIB::sysDescr
sysDescr OBJECT-TYPE
 -- FROM SNMPv2-MIB
 -- TEXTUAL CONVENTION DisplayString
 SYNTAX OCTET STRING (0..255)
 DISPLAY-HINT "255a"
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION "A textual description of the entity. This
 value should include the full name and
 version identification of the system's
 hardware type, software operating-system,
 and networking software."
::= { iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) system(1) 1 }
```

EXAMPLE 2 Displaying a Tree

The following command displays the tree shown below:

```
% snmptranslate -Tp -IR -OS system

+--system(1)
 |
 +-- -R-- String sysDescr(1)
 | Textual Convention: DisplayString
 | Size: 0..255
 +-- -R-- ObjID sysObjectID(2)
 +-- -R-- TimeTicks sysUpTime(3)
```

EXAMPLE 2 Displaying a Tree *(Continued)*

```
|  |
|  +-- sysUpTimeInstance(0)
|
+-- -RW- String sysContact(4)
| Textual Convention: DisplayString
| Size: 0..255
+-- -RW- String sysName(5)
| Textual Convention: DisplayString
| Size: 0..255
+-- -RW- String sysLocation(6)
| Textual Convention: DisplayString
| Size: 0..255
+-- -R-- INTEGER sysServices(7)
|        Range: 0..127
+-- -R-- TimeTicks sysORLastChange(8)
| Textual Convention: TimeStamp
|
+--sysORTable(9)
   |
   +--sysOREntry(1)
      |  Index: sysORIndex(1)
      |
      +-- ---- INTEGER sysORIndex(1)
      |        Range: 1..2147483647
      +-- -R-- ObjID sysORID(2)
      +-- -R-- String sysORDescr(3)
      | Textual Convention: DisplayString
      | Size: 0..255
      +-- -R-- TimeTicks sysORUpTime(4)
               Textual Convention: TimeStamp
```

EXAMPLE 3 Dumping MIB Contents

The commands shown below produce the dumps that follow.

```
% snmptranslate -Ta | head

dump DEFINITIONS ::= BEGIN
org ::= { iso 3 }
dod ::= { org 6 }
internet ::= { dod 1 }
directory ::= { internet 1 }
mgmt ::= { internet 2 }
experimental ::= { internet 3 }
private ::= { internet 4 }
```

EXAMPLE 3 Dumping MIB Contents *(Continued)*

```
security ::= { internet 5 }
snmpV2 ::= { internet 6 }
```

Here is use of the `-Tl` option:

```
% snmptranslate -Tl | head

.iso(1).org(3)
.iso(1).org(3).dod(6)
.iso(1).org(3).dod(6).internet(1)
.iso(1).org(3).dod(6).internet(1).directory(1)
.iso(1).org(3).dod(6).internet(1).mgmt(2)
.iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1)
.iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1)
.iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1).sysDescr(1)
.iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1).sysObjectID(2)
.iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).system(1).sysUpTime(3)
```

Here is the use of the `-To` option:

```
% snmptranslate -To | head

.1.3
.1.3.6
.1.3.6.1
.1.3.6.1.1
.1.3.6.1.2
.1.3.6.1.2.1
.1.3.6.1.2.1.1
.1.3.6.1.2.1.1.1
.1.3.6.1.2.1.1.2
.1.3.6.1.2.1.1.3
```

Here is the use of the `-Ts` option:

```
% snmptranslate -Ts | head

.iso.org
.iso.org.dod
.iso.org.dod.internet
.iso.org.dod.internet.directory
.iso.org.dod.internet.mgmt
.iso.org.dod.internet.mgmt.mib-2
.iso.org.dod.internet.mgmt.mib-2.system
.iso.org.dod.internet.mgmt.mib-2.system.sysDescr
.iso.org.dod.internet.mgmt.mib-2.system.sysObjectID
.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime
```

EXAMPLE 3 Dumping MIB Contents *(Continued)*

Here is the use of the `-Tt` option:

```
% snmptranslate -Tt | head

org(3) type=0
  dod(6) type=0
    internet(1) type=0
      directory(1) type=0
      mgmt(2) type=0
        mib-2(1) type=0
          system(1) type=0
            sysDescr(1) type=2 tc=4 hint=255a
            sysObjectID(2) type=1
            sysUpTime(3) type=8
```

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for matching object errors, after which an error message is displayed.

`2` An error occurred while executing the command. An error message is displayed.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also `snmpcmd(1M)`, `snmp_variables(4)`, `attributes(5)`

Name snmptrap, snmpinform – send an SNMP trap to a manager

Synopsis

```
snmptrap -v 1 [common options] [-Ci] enterprise-oid agent generic-trap specific-trap uptime
     [oid type value]...

snmptrap -v {2c | 3} [common options] [-Ci] uptime trap-oid
     [oid type value]...

snmptrap -v {2c | 3} [common options] uptime trap-oid
     [oid type value]...
```

Description The `snmptrap` utility is an SNMP application that uses the SNMP `TRAP` operation to send information to a network manager. You can specify one or more object identifiers (OIDs) on the command line. A type and a value must accompany each object identifier. Each variable name must be entered in the format specified in `snmp_variables(4)`.

When invoked as `snmpinform`, or when you specify the `-Ci` option to `snmptrap`, it sends an `INFORM-PDU`, expecting a response from the trap receiver, and retransmitting if required. Otherwise, it sends a `TRAP-PDU` or `TRAP2-PDU`.

If any of the required version 1 parameters—`enterprise-oid`, `agent`, and `uptime`—are specified as empty, they default to 1.3.6.1.4.1.3.1.1 (`enterprises.cmu.1.1`), `hostname`, and `host-uptime` respectively.

The *type* is a single character, one of:

```
i INTEGER
u UNSIGNED
c COUNTER32
s STRING
x HEX STRING
d DECIMAL STRING
n NULLOBJ
o OBJID
t TIMETICKS
a IPADDRESS
b BITS
```

For example, the following command sends a generic `linkUp` trap to `manager`, for interface 1:

```
# snmptrap -v 1 -c public manager enterprises.spider test-hub 3 0 ''
interfaces.iftable.ifentry.ifindex.1 i 1
```

Options The following option is supported:

`-Ci` Sends an `INFORM-PDU`, as described above. Applies only to `snmptrap`. This option provides the equivalent function of `snmpinform`.

In addition to this option, `snmptrap` takes the common options described in the `snmpcmd(1M)` manual page.

Exit Status `0` Successful completion.

1 A usage syntax error. A usage message is displayed.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also snmpcmd(1M), snmp_variables(4), attributes(5)

Name snmptrapd – receive and log SNMP trap messages

Synopsis `/usr/sfw/sbin/snmptrapd [`*options*`] [`*listening addresses*`]`

Description The `snmptrapd` utility is an SNMP application that receives and logs SNMP `TRAP` and `INFORM` messages.

The default is to listen on UDP port 162 on all IPv4 interfaces. Because 162 is a privileged port, `snmptrapd` must be be run as root.

Options This command supports the following options:

`-a`	Ignore `authenticationFailure` traps.
`-c` *file*	Read *file* as a configuration file.
`-C`	Do not read any configuration files except the one optionally specified by the `-c` option.
`-d`	Dump (in hexadecimal) the sent and received SNMP packets.
`-D` *token*[,...]	Turn on debugging output for the specified *token*(s). Use `ALL` for extremely verbose output.
`-e`	Print event numbers (rising/falling alarm, and so forth).
`-f`	Do not call `fork()` from the calling shell.
`-F` *format*	When logging to standard output, use the format in the string *format*. See *Format Specifications* below for more details.
`-h`, `--help`	Display a brief usage message and then exit.
`-H`	Display a list of configuration file directives understood by the trap daemon and then exit.
`-l` d | 0–7	Specifies the `syslog(3C)` facility to use when logging to syslog. `d` means `LOG_DAEMON`; `0` through `7` means `LOG_LOCAL0` through `LOG_LOCAL7`. `LOG_LOCAL0` is the default.
`-m` *miblist*	Specifies a colon-separated list of MIB modules to load for this application. This overrides the environment variable `MIBS`.
`-M` *dirlist*	Specifies a colon-separated list of directories to search for MIBs. This overrides the environment variable `MIBDIRS`.
`-n`	Do not attempt to translate source addresses of incoming packets into host names.
`-o` *file*	Log formatted incoming traps to *file*. Upon receipt of a `SIGHUP`, the daemon will close and reopen the log file. This feature is useful when rotating the log file with other utilities such as `logrotate`.

`-P`	Print formatted incoming traps to `stderr`.
`-s`	Log formatted incoming traps to `syslog(3C)`. These `syslog` messages are sent with a level of `LOG_WARNING` and facility as determined by the `-l` flag (`LOG_LOCAL0` by default). This is the default unless you use the `-o` or `-P` flag.
`-u` *file*	Save the process ID of the trap daemon in *file*.
`-v`, `--version`	Print version information for the trap daemon and then exit.

In addition to the preceding options, `snmptrapd` takes the same output formatting options as the other Net-SNMP commands. See the section *OUTPUT OPTIONS* in `snmpcmd(1M)`.

For extensibility and configuration information, see `snmptrapd.conf(4)`.

Format Specifications

`snmptrapd` interprets format strings similarly to `printf(3C)`. It interprets the following formatting sequences:

`%%`	A literal percent sign(`%`).
`%t`	Decimal number of seconds since the operating system's epoch, as returned by `time(2)`.
`%y`	Current year on the local system.
`%m`	Current (numeric) month on the local system.
`%l`	Current day of month on the local system.
`%h`	Current hour on the local system.
`%j`	Current minute on the local system.
`%k`	Current second on the local system.
`%T`	The value of the `sysUpTime.0` *varbind* in seconds.
`%Y`	The year field from the `sysUpTime.0` *varbind*.
`%M`	The numeric month field from the `sysUpTime.0` *varbind*.
`%L`	The day of month field from the `sysUpTime.0` *varbind*.
`%H`	The hour field from the `sysUpTime.0` *varbind*.
`%J`	The minute field from the `sysUpTime.0` *varbind*.
`%K`	The seconds field from the `sysUpTime.0` *varbind*.
`%a`	The contents of the `agent-addr` field of the PDU (v1 `TRAP`s only).
`%A`	The hostname corresponding to the contents of the `agent-addr` field of the PDU, if available. Otherwise the contents of the `agent-addr` field of the PDU (v1 `TRAP`s only).
`%b`	PDU source address (note that this is not necessarily an IPv4 address).

`%B` PDU source hostname if available, otherwise PDU source address (which is not necessarily an IPv4 address).

`%N` Enterprise string.

`%w` Trap type (numeric, in decimal).

`%W` Trap description.

`%q` Trap sub-type (numeric, in decimal).

`%P` Security information from the PDU (community name for v1/v2c, user and context for v3).

`%v` List of trap's variable-bindings.

In addition to these values, you can also specify an optional field width and precision, just as in `printf(3C)`, and a flag value. The following flags are valid:

`-` left justify

`0` use leading zeros

`#` use alternate form

The "use alternate form" flag changes the behavior of some format flags. Normally, the fields that display time information base it on the local timezone, but this flag tells them to use GMT instead. Also, the variable-binding list is normally a tab-separated list, but this flag changes it to a comma-separated one. The alternate form for the uptime is similar to "3 days, 0:14:34.65".

Listening Addresses

By default, `snmptrapd` listens for incoming SNMP `TRAP` and `INFORM` packets on UDP port 162 on all IPv4 interfaces. However, it is possible to modify this behavior by specifying one or more listening addresses as arguments to `snmptrapd`. See `snmpd(1M)` for more information about the format of listening addresses.

NOTIFICATION-LOG-MIB Support

As of Net-SNMP 5.0, the `snmptrapd` application supports the `NOTIFICATION-LOG-MIB`. It does this by opening an AgentX subagent connection to the master `snmpd` agent and registering the notification log tables. As long as the `snmpd` application is started first, it will attach itself to it. Thus you should be able to view the last recorded notifications by means of the `nlmLogTable` and `nlmLogVariableTable`. See `snmptrapd.conf(4)` and the `dontRetainLogs` token for turning off this support. See the `NOTIFICATION-LOG-MIB` for more details about the MIB itself.

Examples

EXAMPLE 1 Using `snmptrapd`

To get a message such as `14:03 TRAP3.1 from humpty.example.edu` you can use a command similar to:

```
# snmptrapd -P -F "%02.2h:%02.2j TRAP%w.%q from %A\n"
```

EXAMPLE 1 Using `snmptrapd` *(Continued)*

If you want the same effect, but in GMT rather than local time, use:

```
# snmptrapd -P -F "%#02.2h:%#02.2j TRAP%w.%q from %A\n"
```

EXAMPLE 2 Viewing Traps on the Host on Which You Invoke `snmptrapd`

To view traps on the host from which you invoke `snmptrapd`, enter:

```
# snmptrapd -P
```

The preceding command sends output to stdout rather than to a log file.

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmagt
Interface Stability	Stable

See Also

`snmpcmd(1M)`, `snmpd(1M)`, `printf(3C)`, `syslog(3C)`, `snmptrapd.conf(4)`, `snmp_variables(4)`, `attributes(5)`

Name snmpusm – create and maintain SNMPv3 users on a remote entity

Synopsis

```
snmpusm [common options] AGENT create user [clonefrom-user]

snmpusm [common options] AGENT delete user

snmpusm [common options] AGENT cloneFrom user clonefrom-user

snmpusm [common options] [-Co] [-Ca] [-Cx] AGENT passwd old-passphrase new-passphrase
```

Description The `snmpusm` utility is an SNMP application that can be used to do simple maintenance on an SNMP agent's User-based Security Module (USM) table. The user needs write access to the usmUserTable MIB table. You can create, delete, clone, and change the passphrase of users configured on a running SNMP agent.

The SNMPv3 USM specifications (see RFC 3414) dictate that users are created and maintained by adding and modifying rows to the `usmUserTable` MIB table. To create a new user you simply create the row using `snmpset(1M)`. User's profiles contain private keys that are never transmitted over the wire in clear text, regardless of whether the administration requests are encrypted.

The secret key for a user is initially set by cloning another user in the table, so that a new user inherits the cloned user's secret key. A user can be cloned only once, however, after which they must be deleted and re-created to be re-cloned. The authentication and privacy security types are also inherited during this cloning (for example, MD5 vs. SHA1). To change the secret key for a user, you must know the user's old passphrase as well as the new one. The `passwd` subcommand of the `snmpusm` command requires both the new and the old passphrases be supplied. After cloning from the appropriate template, you should immediately change the new user's passphrase.

The Net-SNMP agent must first be initialized so that at least one user is setup in it before you can use this command to clone new ones. See the `snmpd.conf(4)` manual page for a description of the `createUser` configuration parameter.

Passphrases must be a minimum of eight characters in length.

Options See `snmpcmd(1M)` for a description of *common options*.

Examples Assume for our examples that the following VACM and USM configurations lines are in the `snmpd.conf` file for a Net-SNMP agent. These lines set up a default user named `initial` with the authentication passphrase `setup_passphrase`. Establishing these parameters enables the initial setup of an agent.

```
# VACM configuration entries
rwuser initial
# The name of the new user that is going to be created
rwuser wes
# USM configuration entries
createUser initial MD5 setup_passphrase DES
```

Note that the `initial` user's setup should be removed after creating a real user to whom you grant administrative privileges. The real user is `wes` in this example.

EXAMPLE 1 Creating a New User

The following command creates a new user, `wes`, which is cloned from `initial`. `wes` inherits that user's passphrase, `setup_passphrase`.

```
# snmpusm -v3 -u initial -n "" -l authNoPriv -a MD5 -A setup_passphrase \
 localhost create wes initial
```

EXAMPLE 2 Changing the User's Passphrase

After creating the user `wes` with the same passphrase as the user `initial`, we need to change his passphrase for `wes`. The following command changes it from `setup_passphrase`, which was inherited from `initial`, to `new_passphrase`.

```
# snmpusm -v 3 -u wes -n "" -l authNoPriv -a MD5 -A setup_passphrase \
localhost passwd setup_passphrase new_passphrase
```

EXAMPLE 3 Testing the New User

If the preceding commands were successful, the following command should perform an authenticated SNMPv3 GET request to the agent.

```
# snmpget -v 3 -u wes -n "" -l authNoPriv -a MD5 -A new_passphrase \
localhost sysUpTime.0
```

Following a successful test, remove the VACM `group` `snmpd.conf` entry for the user `initial`. At this point, you have a valid user `wes` that you can use for future transactions.

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors.

`2` An error occurred while executing the command. An error message is displayed.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

See Also

`snmpcmd(1M)`, `snmpset(1M)`, `snmpd.conf(4)`, `attributes(5)`

RFC 3414

Name snmpvacm – perform maintenance on an SNMP agent's View-based Access Control Module (VACM) table

Synopsis `/usr/sfw/bin/snmpvacm [`*common options*`] [`*subcommand options*`] AGENT subcommand subcommand-ar`

Description `snmpvacm` is a SNMP application that can be used to do maintenance on an SNMP agent's View-based Access Control Module (VACM) table. The VACM table defines a set of services that can be used for checking access rights, that is, checking whether a specific type of access to a specific managed object is allowed. `snmpvacm` supports three types of entries--group, view, and access. The agent maintains these entries in memory and stores VACM groups, views, and access entries in the persistent configuration file upon agent shutdown.

Subcommands This section describes the `snmpvacm` subcommands.

`createSec2Group` Creates SNMPv3 security to group name entries. A group name is used to define an access control policy for a group of principals.

Creates SNMPv3 security to group name entries. A group name is used to define an access control policy for a group of principals.

`snmpvacm [`*common options*`] createSec2Group MODEL SECURITYNAME GROUPNAME`

`MODEL` An integer greater then zero representing a SNMPv3 security model, such as USM. The reserved values are as follows:

`1` reserved for SNMPv1

`2` reserved for SNMPv2c

`3` User-Based Security Model (USM)

`SECURITYNAME` A string representing a security name for the principal, represented in a security-model-independent format, which is mapped from this entry to a `GROUPNAME`.

`GROUPNAME` A string that identifies the group to which this table entry (the combination of `securityModel` and `securityName`) belongs.

`deleteSec2Group` Deletes SNMPv3 security to group name entries. The group entry to be deleted is indexed by the specified `MODEL` and `SECURITYNAME`.

`snmpvacm [`*common options*`] deleteSec2Group MODEL SECURITYNAME`

`MODEL` An integer greater then zero representing a SNMPv3 security model, such as USM. The reserved values are as follows:

`1` reserved for SNMPv1

2 reserved for SNMPv2c

3 User-Based Security Model (USM)

`SECURITYNAME` A string representing a security name for the principal, represented in a security-model-independent format, which is mapped from this entry to a GROUPNAME.

`createView` Creates a MIB view. A MIB view is a family of view subtrees, which are pairings of OID subtree values with bit string mask values.

Each MIB view is defined by two sets of view subtrees, included in or excluded from the MIB view.

`snmpvacm [`*common options*`] [-Ce] createView NAME SUBTREE MASK`

`-Ce` An optional flag used when the MIB view type needs to be "excluded" from the MIB view. If not used, the type is defaulted to "included".

`NAME` The OID subtree which when combined with the corresponding instance of MASK defines a family of view subtrees.

`SUBTREE` The OID subtree which when combined with the corresponding instance of MASK defines a family of view subtrees.

`MASK` The bit mask, a hex string, which, in combination with the corresponding instance SUBTREE, defines a family of view subtrees.

The mask indicates which sub-identifiers of the associated subtree OID are significant to a particular MIB view instance.

`deleteView` Deletes a MIB view. A MIB view is a family of view subtrees. A view subtree is a pairing of an OID subtree value with a bit string mask value.

`snmpvacm [`*common options*`] deleteView NAME SUBTREE`

`NAME` A string representing a MIB view name that is associated to a subtree/mask pairing.

`SUBTREE` The OID subtree which, when combined with the corresponding instance of MASK, defines a family of view subtrees.

`createAccess` Creates SNMPv3 access configuration entries. These entries are used to store the access rights defined for the groups. Each entry is indexed by a

group name, a context prefix, a security model, and a security level. A group and view needs to be defined in order to make use of the access check.

```
snmpvacm [common options] createAccess GROUPNAME
[CONTEXTPREFIX] SECURITYMODEL SECURITYLEVEL
CONTEXTMATCH READVIEWNAME WRITEVIEWNAME
NOTIFYVIEWNAME
```

`GROUPNAME` The name of the group to which this access right applies.

`CONTEXTPREFIX` A string representing a `contextName` must match the value of the instance of this object exactly when `CONTEXTMATCH` is set to “exact” or partially when `CONTEXTMATCH` is set to “prefix”.

If not specified, the value reverts to the default, an empty string, "".

`SECURITYMODEL` An integer representing the `securityModel` that must be used in order to gain access to this access right.

`SECURITYLEVEL` An integer representing the minimum security level that must be used to gain access to this access right. A security level of `noAuthNoPriv` is less than `authNoPriv` and `authNoPriv` is less than `authPriv`.

Integer values supported:

1 `noAuthNoPriv`

2 `authNoPriv`

3 `authPriv`

`CONTEXTMATCH` An integer whose value determines the type of match required. When set to "exact", the context name must exactly match the value in `CONTEXTPREFIX`. If set to "prefix", the context name must match the first few starting characters of the value in `CONTEXTPREFIX`.

Integer values supported:

1 exact

2 prefix

`READVIEWNAME` The authorized MIB view name used for read access. If the value is an empty string, then there is no active view configured for read access.

`WRITEVIEWNAME` The authorized MIB view name used for write access. If the value is an empty string, then there is no active view configured for write access.

`NOTIFYVIEWNAME` The authorized MIB view name used for notify access. If the value is an empty string, then there is no active view configured for notify access.

`deleteAccess` Deletes SNMPv3 access configuration entries, given a group name, context prefix, security model, and security level.

```
snmpvacm [common options] deleteAccess GROUPNAME
[CONTEXTPREFIX] SECURITYMODEL SECURITYLEVEL
```

`GROUPNAME` The name of the group to which this access right applies.

`CONTEXTPREFIX` A string representing a `contextName` must match the value of the instance of this object exactly when `CONTEXTMATCH` is set to "exact" or partially when `CONTEXTMATCH` is set to "prefix".

`SECURITYMODEL` An integer representing the `securityModel` that must be used to gain access to this access right.

`SECURITYLEVEL` An integer representing the minimum security level that must be used to gain access to this access right. A security level of `noAuthNoPriv` is less than `authNoPriv` and `authNoPriv` is less than `authPriv`.

The following integer values are supported:

1 `noAuthNoPriv`

2 `authNoPriv`

3 `authPriv`

Examples

For the following examples, the user is my_user and the password is my_password. Use `net-snmp-config` to create the first user (`my_user`). Then clone `my_user` to configure another SNMPv3 user, `my_user_2`. See snmpusm(1M).

EXAMPLE 1 Creating a VACM Group Entry

Create a VACM group entry, as follows:

EXAMPLE 1 Creating a VACM Group Entry *(Continued)*

```
snmpvacm -v 3 -u my_user -l authPriv -a MD5 -A
        my_password -x DES -X my_password localhost createSec2Group
        3 my_user_2 my_group
```

Run snmpwalk(1M) to verify the group name was created:

```
snmpwalk -v 3 -u my_user -l authPriv -a MD5 -A
        my_password -x DES -X my_password localhost
        SNMP-VIEW-BASED-ACM-MIB::vacmGroupName
```

In addition to other configured VACM group entries, you will note an entry such as the following:

```
SNMP-VIEW-BASED-ACM-MIB::vacmGroupName.3."my_user_2" = STRING: my_group
```

EXAMPLE 2 Creating a MIB View Entry

The command below creates a MIB view entry applicable only to the system group MIB.

```
snmpvacm -v 3 -u my_user -l authPriv -a MD5 -A
        my_password -x DES -X my_password localhost createView
        my_view .1.3.6.1.2.1.1 FF
```

Run snmpwalk(1M) to verify the `my_view` MIB view was created:

```
snmpwalk -v 3 -u my_user -l authPriv -a MD5 -A
        my_password -x DES -X my_password localhost
        SNMP-VIEW-BASED-ACM-MIB::vacmViewTreeFamilyTable
```

In `snmpwalk` output, observe the lines, such as those below, related to the `my_view` MIB view.

```
SNMP-VIEW-BASED-ACM-MIB::vacmViewTreeFamilyMask."my_view".2.1.3.6.1.2.1.1\
= Hex-STRING: FF
SNMP-VIEW-BASED-ACM-MIB::vacmViewTreeFamilyType."my_view".2.1.3.6.1.2.1.1\
= INTEGER: included(1)
SNMP-VIEW-BASED-ACM-MIB::vacmViewTreeFamilyStorageType.\
"my_view".2.1.3.6.1.2.1.1 = INTEGER: nonVolatile(3)
SNMP-VIEW-BASED-ACM-MIB::vacmViewTreeFamilyStatus.\
"my_view".2.1.3.6.1.2.1.1 = INTEGER: active(1)
```

EXAMPLE 3 Creating an Access Entry

The command below creates an access entry using the following components:

- the "`my_group`" entry created above
- an empty prefix string ("")
- the USM security model (3)
- the security level (3)

EXAMPLE 3 Creating an Access Entry *(Continued)*

- the context match (1)
- the read view name ("my_view")
- the write view name ("")
- the notify view name ("")

```
snmpvacm -v 3 -u my_user -l authPriv -a MD5 -A
        my_password -x DES -X my_password localhost createAccess
        my_group "" 3 3 1 my_view "" ""
```

Run snmpwalk(1M) to verify the access entry was created:

```
snmpwalk -v 3 -u my_user -l authPriv -a MD5 -A
        my_password -x DES -X my_password localhost
        SNMP-VIEW-BASED-ACM-MIB::vacmAccessTable

SNMP-VIEW-BASED-ACM-MIB::vacmAccessContextMatch."my_group"."".3.authPriv\
= INTEGER: exact(1)
SNMP-VIEW-BASED-ACM-MIB::vacmAccessReadViewName."my_group"."".3.authPriv\
= STRING: my_view
SNMP-VIEW-BASED-ACM-MIB::vacmAccessWriteViewName."my_group"."".3.authPriv\
= STRING:
SNMP-VIEW-BASED-ACM-MIB::vacmAccessNotifyViewName."my_group"."".3.authPriv\
= STRING:
SNMP-VIEW-BASED-ACM-MIB::vacmAccessStorageType."my_group"."".3.authPriv\
= INTEGER: nonVolatile(3)
SNMP-VIEW-BASED-ACM-MIB::vacmAccessStatus."my_group"."".3.authPriv\
= INTEGER: active(1)
```

EXAMPLE 4 Testing the Configuration

Test the preceding setup by verifying the access setup. You do this by accessing an object in the system group and another object outside this range. Note the use of the user name `my_user_2`.

```
snmpget -mALL -v 3 -u my_user_2 -l authPriv -a MD5
        -A my_password -x DES -X my_password localhost sysObjectID.0
```

At this point, when you to access an object outside the access range, the attempt fails with an appropriate error:

```
snmpgetnext -mALL -v 3 -u my_user_2 -l authPriv -a MD5
        -A my_password -x DES -X my_password localhost ifTable

    RFC1213-MIB::ifTable = No more variables left in this MIB View (It is
        past the end of the MIB tree)
```

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message displays. Also used for time out errors.

`2` An error occurred while executing the command. An error message displays.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	Stable

See Also

`snmpusm(1M)`, `snmpwalk(1M)`, `snmpd.conf(4)`, `attributes(5)`

RFC 3415

Name snmpwalk – communicate with a network entity using SNMP GETNEXT requests

Synopsis

```
snmpwalk [application options] [common options] [oid]
```

Description The `snmpwalk` utility is an SNMP application that uses SNMP `GETNEXT` requests to query a network entity for a tree of information.

You can specify an object identifier (OID) on the command line. This OID specifies which portion of the object identifier space will be searched using `GETNEXT` requests. All variables in the subtree below the specified OID are queried and their values returned. Each variable name must be entered in the format specified in `snmp_variables(4)`.

If no OID argument is present, `snmpwalk` searches MIB-2. If the network entity has an error processing the request packet, an error packet is returned and a message displayed, indicating t the way in which the request was malformed.

If the tree search causes attempts to search beyond the end of the MIB, the message "End of MIB" is displayed.

Options The following options are supported:

`-Cc` Do not check whether the returned OIDs are increasing. Some agents (for example, agents for Laser-Jet printers) return OIDs out of order, but can complete the walk anyway. Other agents return OIDs that are out of order and can cause `snmpwalk` to loop indefinitely. By default, `snmpwalk` tries to detect this behavior and warns you when it encounters an agent acting illegally. Use `-Cc` to turn off this feature.

`-Ci` Include the given OID in the search range. Normally, snmpwalk uses `GETNEXT` requests starting with the OID you specify and returns all results in the MIB tree beyond that OID. This option allows you to include the OID specified on the command line in the displayed results (assuming the OID is a valid OID in the tree).

`-Cp` Upon completion of the walk, display the number of variables found.

In addition to these options, `snmpwalk` takes the common options described in the `snmpcmd(1M)` manual page.

Examples **EXAMPLE 1** Retrieving All of the Variables Under `system`

The following command retrieves all of the variables under `system`:

```
# snmpwalk -Os -c public -v 1 zeus system
```

Output from this command will be similar to:

```
sysDescr.0 = STRING: "SunOS zeus.net.cmu.edu 4.1.3_U1 1 sun4m"
sysObjectID.0 = OID: enterprises.hp.nm.hpsystem.10.1.1
sysUpTime.0 = Timeticks: (155274552) 17 days, 23:19:05
sysContact.0 = STRING: ""
sysName.0 = STRING: "zeus.net.cmu.edu"
```

EXAMPLE 1 Retrieving All of the Variables Under `system` *(Continued)*

```
sysLocation.0 = STRING: ""
sysServices.0 = INTEGER: 72
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsmcmd
Interface Stability	External

Exit Status

`0` Successful completion.

`1` A usage syntax error. A usage message is displayed. Also used for timeout errors.

`2` An error occurred while executing the command. An error message is displayed.

See Also

`snmpbulkwalk(1M)`, `snmpcmd(1M)`, `snmp_variables(4)`, `attributes(5)`

Name snmpXdmid – Sun Solstice Enterprise SNMP-DMI mapper subagent

Synopsis

```
/usr/lib/dmi/snmpXdmid  -s hostname [-h] [-c config-dir]
 [-d debug-level]
```

Description The `snmpXdmid` utility is a subagent in the Solstice Enterprise Agent Desktop Management Interface package. It maps the SNMP requests forwarded by the Master Agent (`snmpdx(1M)`) into one or more equivalent DMI requests. Further, it remaps the DMI response into SNMP response back to `snmpdx`. By default, `snmpXdmid` also forwards the DMI indications as SNMP traps to `snmpdx`. The feature is configurable and can be disabled by setting `TRAP_FORWARD_TO_MAGENT=0` in the `snmpXdmid` configuration file, `snmpXdmid.conf`.

This subagent runs as a daemon in the system. The subagent uses a set of .MAP files located in `/var/dmi/map` to map the SNMP Object Identifier (OID) into a corresponding DMI component. The map files are generated using the MIF-to-MIB utility, `miftomib`. They are read by `snmpXdmid` when a corresponding MIF file gets registered with the DMI Service Provider (`dmispd(1M)`).

Note – The SMA (Systems Management Agent) is the default SNMP agent in Solaris. See `sma_snmp(5)`. `snmpdx` is Obsolete and may not be supported in a future release of Solaris.

The `snmpXdmid.conf` file is used for configuration information. Each entry in the file consists of a keyword followed by an equal sign (=), followed by a parameter string. The keyword must begin in the first position. A line beginning with a pound sign (#) is treated as a comment and the subsequent characters on that line are ignored. The keywords currently supported are:

`WARNING_TIMESTAMP`
Indication subscription expiration, warning time.

`EXPIRATION_TIMESTAMP`
Indication subscription expiration timestamp.

`FAILURE_THRESHOLD`
DMISP retries before dropping indication due to comm errors.

`TRAP_FORWARD_TO_MAGENT`

`0`	Drop indication at the subagent level.
non-zero	Forward indications as SNMP traps to `snmpdx`.

By default, the configuration file `snmpXdmid.conf` is located in the `/etc/dmi/conf` directory. You can specify an alternative directory with the `-c` option.

Options The following options are supported:

`-c` *config-dir*	Specify the directory where `snmpXdmid.conf` file is located.
`-d` *debug-level*	Debug. Levels from 1 to 5 are supported, giving various levels of debug information.

`-h` Help. Print the command line usage.

`-s` *hostname* Specify the host on which `dmispd` is running.

Files `/etc/dmi/conf/snmpXdmid.conf` DMI mapper configuration file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadmi
Interface Stability	Obsolete

See Also `dmispd(1M)`, `snmpdx(1M)`, `attributes(5)`, `sma_snmp(5)`

Name snmpXwbemd – SNMP Adapter Subagent for WBEM

Synopsis

```
/usr/sadm/lib/wbem/snmpXwbemd [-d] [-h] [-p port]
```

Description The `snmpXwbemd` daemon is a subagent in the Web-Based Enterprise Management (WBEM) services package.

This daemon maps the Simple Network Management Protocol (SNMP) requests forwarded by the Solstice Enterprise Agents (SEA) Master Agent `snmpdx(1M)` into one or more equivalent WBEM Common Information Model (CIM) properties and instances. Further, it remaps the response from the CIM Object Manager into a SNMP response, which it passes back to `snmpdx`(1M).

A mapping file contains the corresponding Object Identifier (OID), class name, property name, and Abstract Syntax Notation 1 (ASN.1) type for each object. You can also create your own mapping file.

Options The following options are supported:

`-d` Displays all debug information.

`-h` Displays help by printing the correct command line usage.

`-p` Specifies the port number to use.

Operands The following operand is supported:

port Specifies the port number you want to use.

Examples **EXAMPLE 1** An Example of a `050SUNWwbcou.map` File

This mapping file that Sun Microsystems provides contains definitions of objects, in this format:

```
#
#pragma ident    "@(#)050SUNWwbcou.map    1.0    01/04/03 SMI"
#
# Copyright (c) 2001 by Sun Microsystems, Inc.
# All rights reserved.
#
# *** Description of contents ***
#
# First non-commented non-blank line contains required Version label.
# Remaining non-commented non-blank lines are considered map entries
# used as described below:
#
# Column 1 - SNMP OID - Uniquely describes an SNMP variable
# Column 2 - CIM Class Name - CIM class associated with this variable
# Column 3 - CIM Property Name - CIM property that maps to SNMP OID variable
# Column 4 - ASN.1 type - SNMP datatype that dictates how data is mapped
#            to/from SNMP requests.  Supported types are: SnmpString, SnmpOid,
```

EXAMPLE 1 An Example of a `050SUNWwbcou.map` File *(Continued)*

```
#            SnmpTimeticks, SnmpCounter, SnmpInt, SnmpGauge, SnmpIpAddress,
#            SnmpOpaque)
# Column 5 and greater are ignored
#
Version 1.0

1.3.6.1.2.1.1.1.0 Solaris_ComputerSystem Description SnmpString
1.3.6.1.2.1.1.3.0 Solaris_OperatingSystem LastBootUpTime SnmpTimeticks
1.3.6.1.2.1.1.4.0 Solaris_ComputerSystem PrimaryOwnerContact SnmpString
1.3.6.1.2.1.1.5.0 Solaris_ComputerSystem Name SnmpString

1.3.6.1.2.1.25.1.5.0 Solaris_OperatingSystem NumberOfUsers SnmpGauge
1.3.6.1.2.1.25.1.6.0 Solaris_OperatingSystem NumberOfProcesses SnmpGauge
1.3.6.1.2.1.25.1.7.0 Solaris_OperatingSystem MaxNumberOfProcesses SnmpGauge
1.3.6.1.2.1.25.1.2.0 Solaris_OperatingSystem LocalDateTime SnmpString
```

Each definition of an object in this file contains an OID, its corresponding CIM class name, its corresponding CIM property name, and its corresponding ASN.1 type. Each of these elements is separated by a space character.

Files

`/var/sadm/wbem/snmp/map/050SUNWwbcou.map` — The SNMP Adapter Subagent for WBEM MIB–2 mapping file that Sun Microsystems provides contains SNMP Management Information Base (MIB) definitions for the CIM instrumentation that SNMP manages.

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbcou
CSI	Enabled
Interface Stability	Evolving
MT-Level	Safe

See Also snmpdx(1M), attributes(5)

Name snoop – capture and inspect network packets

Synopsis

```
snoop  [-aqrCDNPSvV] [-t [r |  a |  d]] [-c maxcount]
 [-d device] [-i filename] [-n filename] [-o filename]
 [-p first [, last]] [-s snaplen] [-x offset [, length]]
 [expression]
```

Description `snoop` captures packets from the network and displays their contents. `snoop` uses both the network packet filter and streams buffer modules to provide efficient capture of packets from the network. Captured packets can be displayed as they are received, or saved to a file (which is *RFC 1761*–compliant) for later inspection.

`snoop` can display packets in a single-line summary form or in verbose multi-line forms. In summary form, with the exception of certain VLAN packets, only the data pertaining to the highest level protocol is displayed. If a packet has a VLAN header and its VLAN ID is non-zero, then `snoop` will show that the packet is VLAN tagged. For example, an NFS packet will have only NFS information displayed. Except for VLAN information under the condition just described, the underlying RPC, UDP, IP, and Ethernet frame information is suppressed, but can be displayed if either of the verbose options are chosen.

In the absence of a name service, such as LDAP or NIS, `snoop` displays host names as numeric IP addresses.

`snoop` requires an interactive interface.

Options

`-C`
List the code generated from the filter expression for either the kernel packet filter, or `snoop`'s own filter.

`-D`
Display number of packets dropped during capture on the summary line.

`-N`
Create an IP address-to-name file from a capture file. This must be set together with the `-i` option that names a capture file. The address-to-name file has the same name as the capture file with `.names` appended. This file records the IP address to hostname mapping at the capture site and increases the portability of the capture file. Generate a `.names` file if the capture file is to be analyzed elsewhere. Packets are not displayed when this flag is used.

`-P`
Capture packets in non-promiscuous mode. Only broadcast, multicast, or packets addressed to the host machine will be seen.

`-S`
Display size of the entire link layer frame in bytes on the summary line.

`-V`
Verbose summary mode. This is halfway between summary mode and verbose mode in degree of verbosity. Instead of displaying just the summary line for the highest level protocol in a packet, it displays a summary line for each protocol layer in the packet. For

instance, for an NFS packet it will display a line each for the ETHER, IP, UDP, RPC and NFS layers. Verbose summary mode output may be easily piped through `grep` to extract packets of interest. For example, to view only RPC summary lines, enter the following: `example# snoop -i rpc.cap -V | grep RPC`

`-a`
Listen to packets on `/dev/audio` (warning: can be noisy).

`-c` *maxcount*
Quit after capturing *maxcount* packets. Otherwise keep capturing until there is no disk space left or until interrupted with Control-C.

`-d` *device*
Receive packets from the network using the interface specified by *device*, for example, `eri0` or `hme0`. The program netstat(1M), when invoked with the `-i` flag, lists all the interfaces that a machine has. Normally, `snoop` will automatically choose the first non-loopback interface it finds.

`-i` *filename*
Display packets previously captured in *filename*. Without this option, `snoop` reads packets from the network interface. If a *filename*`.names` file is present, it is automatically loaded into the `snoop` IP address-to-name mapping table (See `-N` flag).

`-n` *filename*
Use *filename* as an IP address-to-name mapping table. This file must have the same format as the `/etc/hosts` file (IP address followed by the hostname).

`-o` *filename*
Save captured packets in *filename* as they are captured. (This *filename* is referred to as the "capture file".) The format of the capture file is RFC 1761–compliant. During packet capture, a count of the number of packets saved in the file is displayed. If you wish just to count packets without saving to a file, name the file `/dev/null`.

`-p` *first* [, `last`]
Select one or more packets to be displayed from a capture file. The *first* packet in the file is packet number 1.

`-q`
When capturing network packets into a file, do not display the packet count. This can improve packet capturing performance.

`-r`
Do not resolve the IP address to the symbolic name. This prevents `snoop` from generating network traffic while capturing and displaying packets. However, if the `-n` option is used, and an address is found in the mapping file, its corresponding name will be used.

`-s` *snaplen*
Truncate each packet after *snaplen* bytes. Usually the whole packet is captured. This option is useful if only certain packet header information is required. The packet truncation is done within the kernel giving better utilization of the streams packet buffer. This means

less chance of dropped packets due to buffer overflow during periods of high traffic. It also saves disk space when capturing large traces to a capture file. To capture only IP headers (no options) use a *snaplen* of 34. For UDP use 42, and for TCP use 54. You can capture RPC headers with a *snaplen* of 80 bytes. NFS headers can be captured in 120 bytes.

`-t [ r | a | d ]`
Time-stamp presentation. Time-stamps are accurate to within 4 microseconds. The default is for times to be presented in `d` (delta) format (the time since receiving the previous packet). Option `a` (absolute) gives wall-clock time. Option `r` (relative) gives time relative to the first packet displayed. This can be used with the `-p` option to display time relative to any selected packet.

`-v`
Verbose mode. Print packet headers in lots of detail. This display consumes many lines per packet and should be used only on selected packets.

`-x`*offset* [, *length*]
Display packet data in hexadecimal and ASCII format. The *offset* and *length* values select a portion of the packet to be displayed. To display the whole packet, use an *offset* of 0. If a *length* value is not provided, the rest of the packet is displayed.

Operands *expression*
Select packets either from the network or from a capture file. Only packets for which the expression is true will be selected. If no expression is provided it is assumed to be true.

Given a filter expression, `snoop` generates code for either the kernel packet filter or for its own internal filter. If capturing packets with the network interface, code for the kernel packet filter is generated. This filter is implemented as a streams module, upstream of the buffer module. The buffer module accumulates packets until it becomes full and passes the packets on to `snoop`. The kernel packet filter is very efficient, since it rejects unwanted packets in the kernel before they reach the packet buffer or `snoop`. The kernel packet filter has some limitations in its implementation; it is possible to construct filter expressions that it cannot handle. In this event, `snoop` tries to split the filter and do as much filtering in the kernel as possible. The remaining filtering is done by the packet filter for `snoop`. The `-C` flag can be used to view generated code for either the packet filter for the kernel or the packet filter for `snoop`. If packets are read from a capture file using the `-i` option, only the packet filter for `snoop` is used.

A filter *expression* consists of a series of one or more boolean primitives that may be combined with boolean operators (`AND`, `OR`, and `NOT`). Normal precedence rules for boolean operators apply. Order of evaluation of these operators may be controlled with parentheses. Since parentheses and other filter expression characters are known to the shell, it is often necessary to enclose the filter expression in quotes. Refer to Example 2 for information about setting up more efficient filters.

The primitives are:

`host` *hostname*
: True if the source or destination address is that of `hostname`. The *hostname* argument may be a literal address. The keyword `host` may be omitted if the name does not conflict with the name of another expression primitive. For example, `pinky` selects packets transmitted to or received from the host `pinky`, whereas `pinky and dinky` selects packets exchanged between hosts `pinky AND dinky`.

  The type of address used depends on the primitive which precedes the `host` primitive. The possible qualifiers are `inet`, `inet6`, `ether`, or none. These three primitives are discussed below. Having none of the primitives present is equivalent to "inet host hostname or inet6 host hostname". In other words, snoop tries to filter on all IP addresses associated with hostname.

inet or *inet6*
: A qualifier that modifies the `host` primitive that follows. If it is *inet*, then `snoop` tries to filter on all IPv4 addresses returned from a name lookup. If it is *inet6*, `snoop` tries to filter on all IPv6 addresses returned from a name lookup.

ipaddr, *atalkaddr*, or *etheraddr*
: Literal addresses, IP dotted, AppleTalk dotted, and Ethernet colon are recognized. For example,

  - "`172.16.40.13`" matches all packets with that IP
  - "`2::9255:a00:20ff:fe73:6e35`" matches all packets with that IPv6 address as source or destination;
  - "`65281.13`" matches all packets with that AppleTalk address;
  - "`8:0:20:f:b1:51`" matches all packets with the Ethernet address as source or destination.

  An Ethernet address beginning with a letter is interpreted as a hostname. To avoid this, prepend a zero when specifying the address. For example, if the Ethernet address is `aa:0:45:23:52:44`, then specify it by add a leading zero to make it `0aa:0:45:23:52:44`.

`from` or `src`
: A qualifier that modifies the following `host`, `net`, *ipaddr*, *atalkaddr*, *etheraddr*, `port` or `rpc` primitive to match just the source address, port, or RPC reply.

`to` or `dst`
: A qualifier that modifies the following `host`, `net`, *ipaddr*, *atalkaddr*, *etheraddr*, `port` or `rpc` primitive to match just the destination address, port, or RPC call.

`ether`
: A qualifier that modifies the following `host` primitive to resolve a name to an Ethernet address. Normally, IP address matching is performed. This option is not supported on media such as IPoIB (IP over InfiniBand).

`ethertype` *number*
True if the Ethernet type field has value *number*. If *number* is not 0x8100 (VLAN) and the packet is VLAN tagged, then the expression will match the encapsulated Ethernet type.

`ip`, `ip6`, `arp`, `rarp`, `pppoed`, `pppoes`
True if the packet is of the appropriate ethertype.

`vlan`
True if the packet has `ethertype` VLAN and the VLAN ID is not zero.

`vlan-id` *id*
True for packets of ethertype VLAN with the id *id*.

`pppoe`
True if the ethertype of the packet is either `pppoed` or `pppoes`.

`broadcast`
True if the packet is a broadcast packet. Equivalent to `ether[2:4] = 0xffffffff` for Ethernet. This option is not supported on media such as IPoIB (IP over InfiniBand).

`multicast`
True if the packet is a multicast packet. Equivalent to "`ether[0] & 1 = 1`" on Ethernet. This option is not supported on media such as IPoIB (IP over InfiniBand).

`bootp`, `dhcp`
True if the packet is an unfragmented IPv4 UDP packet with either a source port of `BOOTPS (67)` and a destination port of `BOOTPC (68)`, or a source port of `BOOTPC (68)` and a destination of `BOOTPS (67)`.

`dhcp6`
True if the packet is an unfragmented IPv6 UDP packet with either a source port of `DHCPV6-SERVER` (547) and a destination port of `DHCPV6-CLIENT` (546), or a source port of `DHCPV6-CLIENT` (546) and a destination of `DHCPV6-SERVER` (547).

`apple`
True if the packet is an Apple Ethertalk packet. Equivalent to "`ethertype 0x809b or ethertype 0x80f3`".

`decnet`
True if the packet is a DECNET packet.

`greater` *length*
True if the packet is longer than *length*.

`less` *length*
True if the packet is shorter than *length*.

`udp`, `tcp`, `icmp`, `icmp6`, `ah`, `esp`
True if the IP or IPv6 protocol is of the appropriate type.

`net` *net*
True if either the IP source or destination address has a network number of *net*. The `from` or `to` qualifier may be used to select packets for which the network number occurs only in the source or destination address.

`port` *port*
True if either the source or destination port is *port*. The *port* may be either a port number or name from `/etc/services`. The `tcp` or `udp` primitives may be used to select TCP or UDP ports only. The `from` or `to` qualifier may be used to select packets for which the *port* occurs only as the source or destination.

`rpc` *prog* [, *vers* [, `proc`]]
True if the packet is an RPC call or reply packet for the protocol identified by *prog*. The *prog* may be either the name of an RPC protocol from `/etc/rpc` or a program number. The *vers* and `proc` may be used to further qualify the program *version* and *procedure* number, for example, `rpc nfs,2,0` selects all calls and replies for the NFS null procedure. The `to` or `from` qualifier may be used to select either call or reply packets only.

`ldap`
True if the packet is an LDAP packet on port 389.

`gateway` *host*
True if the packet used *host* as a gateway, that is, the Ethernet source or destination address was for *host* but not the IP address. Equivalent to "`ether host` *host* and not host *host*".

`nofrag`
True if the packet is unfragmented or is the first in a series of IP fragments. Equivalent to `ip[6:2] & 0x1fff = 0`.

expr relop expr
True if the relation holds, where *relop* is one of >, <, >=, <=, =, `!=`, and `expr` is an arithmetic expression composed of numbers, packet field selectors, the `length` primitive, and arithmetic operators +, –, *, &, |, ^, and %. The arithmetic operators within `expr` are evaluated before the relational operator and normal precedence rules apply between the arithmetic operators, such as multiplication before addition. Parentheses may be used to control the order of evaluation. To use the value of a field in the packet use the following syntax:

base`[expr [: size ] ]`

where `expr` evaluates the value of an offset into the packet from a *base* offset which may be `ether`, `ip`, `ip6`, `udp`, `tcp`, or `icmp`. The `size` value specifies the size of the field. If not given, 1 is assumed. Other legal values are 2 and 4. For example,

```
ether[0] & 1 = 1
```

is equivalent to `multicast`

`ether[2:4] = 0xffffffff`

is equivalent to `broadcast`.

`ip[ip[0] & 0xf * 4 : 2] = 2049`

is equivalent to `udp[0:2] = 2049`

`ip[0] & 0xf > 5`

selects IP packets with options.

`ip[6:2] & 0x1fff = 0`

eliminates IP fragments.

`udp and ip[6:2]&0x1fff = 0 and udp[6:2] != 0`

finds all packets with UDP checksums.

The `length` primitive may be used to obtain the length of the packet. For instance "`length > 60`" is equivalent to "`greater 60`", and "`ether[length − 1]`" obtains the value of the last byte in a packet.

`and`
Perform a logical AND operation between two boolean values. The AND operation is implied by the juxtaposition of two boolean expressions, for example "`dinky pinky`" is the same as "`dinky AND pinky`".

`or` or `,`
Perform a logical OR operation between two boolean values. A comma may be used instead, for example, "`dinky,pinky`" is the same as "`dinky OR pinky`".

`not` or `!`
Perform a logical NOT operation on the following boolean value. This operator is evaluated before AND or OR.

`slp`
True if the packet is an SLP packet.

`sctp`
True if the packet is an SCTP packet.

`ospf`
True if the packet is an OSPF packet.

Examples

EXAMPLE 1 Using the `snoop` Command

Capture all packets and display them as they are received:

```
example# snoop
```

Capture packets with host `funky` as either the source or destination and display them as they are received:

EXAMPLE 1 Using the snoop Command *(Continued)*

```
example# snoop funky
```

Capture packets between funky and pinky and save them to a file. Then inspect the packets using times (in seconds) relative to the first captured packet:

```
example# snoop -o cap funky pinky
example# snoop -i cap -t r | more
```

To look at selected packets in another capture file:

```
example# snoop -i pkts -p 99,108
 99   0.0027   boutique -> sunroof     NFS C GETATTR FH=8E6
100   0.0046   sunroof -> boutique     NFS R GETATTR OK
101   0.0080   boutique -> sunroof NFS C RENAME FH=8E6C MTra00192 to .nfs08
102   0.0102   marmot -> viper        NFS C LOOKUP FH=561E screen.r.13.i386
103   0.0072   viper -> marmot       NFS R LOOKUP No such file or directory
104   0.0085   bugbomb -> sunroof    RLOGIN C PORT=1023 h
105   0.0005   kandinsky -> sparky    RSTAT C Get Statistics
106   0.0004   beeblebrox -> sunroof  NFS C GETATTR FH=0307
107   0.0021   sparky -> kandinsky    RSTAT R
108   0.0073   office -> jeremiah      NFS C READ FH=2584 at 40960 for 8192
```

To look at packet 101 in more detail:

```
example# snoop -i pkts -v -p101
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 101 arrived at 16:09:53.59
ETHER:  Packet size = 210 bytes
ETHER:  Destination = 8:0:20:1:3d:94, Sun
ETHER:  Source      = 8:0:69:1:5f:e,  Silicon Graphics
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4, header length = 20 bytes
IP:   Type of service = 00
IP:         ..0. .... = routine
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 196 bytes
IP:   Identification 19846
IP:   Flags = 0X
IP:   .0.. .... = may fragment
IP:   ..0. .... = more fragments
IP:   Fragment offset = 0 bytes
IP:   Time to live = 255 seconds/hops
```

EXAMPLE 1 Using the snoop Command *(Continued)*

```
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 18DC
IP:   Source address = 172.16.40.222, boutique
IP:   Destination address = 172.16.40.200, sunroof
IP:
UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 1023
UDP:  Destination port = 2049 (Sun RPC)
UDP:  Length = 176
UDP:  Checksum = 0
UDP:
RPC:  ----- SUN RPC Header -----
RPC:
RPC:  Transaction id = 665905
RPC:  Type = 0 (Call)
RPC:  RPC version = 2
RPC:  Program = 100003 (NFS), version = 2, procedure = 1
RPC:  Credentials: Flavor = 1 (Unix), len = 32 bytes
RPC:     Time = 06-Mar-90 07:26:58
RPC:     Hostname = boutique
RPC:     Uid = 0, Gid = 1
RPC:     Groups = 1
RPC:  Verifier   : Flavor = 0 (None), len = 0 bytes
RPC:
NFS:  ----- SUN NFS -----
NFS:
NFS:  Proc = 11 (Rename)
NFS:  File handle = 000016430000000100080000305A1C47
NFS:                597A00000008000020463l4AFC450000
NFS:  File name = MTra00192
NFS:  File handle = 000016430000000100080000305A1C47
NFS:                597A00000008000020463l4AFC450000
NFS:  File name = .nfs08
NFS:
```

To view just the NFS packets between `sunroof` and `boutique`:

```
example# snoop -i pkts rpc nfs and sunroof and boutique
1   0.0000   boutique -> sunroof    NFS C GETATTR FH=8E6C
2   0.0046    sunroof -> boutique   NFS R GETATTR OK
3   0.0080   boutique -> sunroof   NFS C RENAME FH=8E6C MTra00192 to .nfs08
```

To save these packets to a new capture file:

```
example# snoop -i pkts -o pkts.nfs rpc nfs sunroof boutique
```

To view encapsulated packets, there will be an indicator of encapsulation:

EXAMPLE 1 Using the snoop Command *(Continued)*

```
example# snoop ip-in-ip
sunroof -> boutique ICMP Echo request    (1 encap)
```

If -V is used on an encapsulated packet:

```
example# snoop -V ip-in-ip
sunroof -> boutique  ETHER Type=0800 (IP), size = 118 bytes
sunroof -> boutique  IP D=172.16.40.222 S=172.16.40.200 LEN=104, ID=27497
sunroof -> boutique  IP  D=10.1.1.2 S=10.1.1.1 LEN=84, ID=27497
sunroof -> boutique  ICMP Echo request
```

EXAMPLE 2 Setting Up A More Efficient Filter

To set up a more efficient filter, the following filters should be used toward the end of the expression, so that the first part of the expression can be set up in the kernel: `greater`, `less`, `port`, `rpc`, `nofrag`, and `relop`. The presence of `OR` makes it difficult to split the filtering when using these primitives that cannot be set in the kernel. Instead, use parentheses to enforce the primitives that should be `OR`'d.

To capture packets between `funky` and `pinky` of type `tcp` or `udp` on `port` 80:

```
example# snoop funky and pinky and port 80 and tcp or udp
```

Since the primitive `port` cannot be handled by the kernel filter, and there is also an `OR` in the expression, a more efficient way to filter is to move the `OR` to the end of the expression and to use parentheses to enforce the `OR` between `tcp` and `udp`:

```
example# snoop funky and pinky and (tcp or udp) and port 80
```

Exit Status

`0` Successful completion.

`1` An error occurred.

Files

`/dev/audio`	Symbolic link to the system's primary audio device.
`/dev/null`	The null file.
`/etc/hosts`	Host name database.
`/etc/rpc`	RPC program number data base.
`/etc/services`	Internet services and aliases.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmdc

See Also netstat(1M), hosts(4), rpc(4), services(4), attributes(5), audio(7I), bufmod(7M), dlpi(7P), pfmod(7M), tun(7M)

Callaghan, B. and Gilligan, R. *RFC 1761, Snoop Version 2 Packet Capture File Format*. Network Working Group. February 1995.

Warnings The processing overhead is much higher for realtime packet interpretation. Consequently, the packet drop count may be higher. For more reliable capture, output raw packets to a file using the `-o` option and analyze the packets off-line.

Unfiltered packet capture imposes a heavy processing load on the host computer, particularly if the captured packets are interpreted realtime. This processing load further increases if verbose options are used. Since heavy use of `snoop` may deny computing resources to other processes, it should not be used on production servers. Heavy use of `snoop` should be restricted to a dedicated computer.

`snoop` does not reassemble IP fragments. Interpretation of higher level protocol halts at the end of the first IP fragment.

`snoop` may generate extra packets as a side-effect of its use. For example it may use a network name service (NIS or NIS+) to convert IP addresses to host names for display. Capturing into a file for later display can be used to postpone the address-to-name mapping until after the capture session is complete. Capturing into an NFS-mounted file may also generate extra packets.

Setting the `snaplen` (`-s` option) to small values may remove header information that is needed to interpret higher level protocols. The exact cutoff value depends on the network and protocols being used. For NFS Version 2 traffic using UDP on 10 Mb/s Ethernet, do not set `snaplen` less than 150 bytes. For NFS Version 3 traffic using TCP on 100 Mb/s Ethernet, `snaplen` should be 250 bytes or more.

`snoop` requires information from an RPC request to fully interpret an RPC reply. If an RPC reply in a capture file or packet range does not have a request preceding it, then only the RPC reply header will be displayed.

Name soconfig – configure transport providers for use by sockets

Synopsis

```
/sbin/soconfig -f file

/sbin/soconfig family type protocol [path]
```

Description The `soconfig` utility configures the transport provider driver for use with sockets. It specifies how the family, type, and protocol parameters in the `socket(3SOCKET)` call are mapped to the name of a transport provider such as `/dev/tcp`. This utility can be used to add an additional mapping or remove a previous mapping.

The `init(1M)` utility uses `soconfig` with the `sock2path(4)` file during the booting sequence.

Options The following options are supported:

`-f` *file* Set up the `soconfig` configuration for each driver according to the information stored in `file`. A `soconfig` file consists of lines of at least the first three fields listed below, separated by spaces:

family type protocol path

These fields are described in the `OPERANDS` section below.

An example of `file` can be found in the `EXAMPLES` section below.

Operands The following operands are supported:

family The protocol family as listed in the `/usr/include/sys/socket.h` file, expressed as an integer.

type The socket type as listed in the `/usr/include/sys/socket.h` file, expressed as an integer.

protocol The protocol number as specified in the family-specific `include` file, expressed as an integer. For example, for `AF_INET` this number is specified in `/usr/include/netinet/in.h`. An unspecified protocol number is denoted with the value zero.

path The string that specifies the path name of the device that corresponds to the transport provider. If this parameter is specified, the configuration will be added for the specified family, type, and protocol. If this parameter is not specified, the configuration will be removed.

Examples **EXAMPLE 1** Using `soconfig`

The following example sets up `/dev/tcp` for family `AF_INET` and type `SOCK_STREAM`:

```
example# soconfig 2 2 0 /dev/tcp
```

The following is a sample file used with the `-f` option. Comment lines begin with a number sign (#):

EXAMPLE 1 Using `soconfig` *(Continued)*

```
#     Family   Type  Protocol    Path
      2        2       0         /dev/tcp
      2        2       6         /dev/tcp

      2        1       0         /dev/udp
      2        1       17        /dev/udp

      1        2       0         /dev/ticotsord
      1        1       0         /dev/ticlts

      2        4       0         /dev/rawip
```

Files `/etc/sock2path` file containing mappings from sockets to transport providers

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

See Also `init(1M)`, `sock2path(4)`, `attributes(5)`

Network Interface Guide

Name soladdapp – add an application to the Solstice application registry

Synopsis

```
/usr/snadm/bin/soladdapp [-r registry] -n name -i icon -e executable
    [args]
```

Description `soladdapp` adds an application to the Solstice application registry. After it is added, the application is displayed in the Solstice Launcher main window (see `solstice(1M)`).

Options

`-r` *registry*	Define the full path name of the Solstice registry file.
`-n` *name*	Define the name of the tool to be registered.
`-i` *icon*	Define the full path name of the tool icon.
`-e` *executable*	Define the full path name of the tool.
args	Specify any arguments to use with the tool.

When executed without options, `soladdapp` uses `/opt/SUNWadm/etc/.solstice_registry` (the default registry path).

Return Values

`0`	on success
`1`	on failure
`2`	if the registry is locked
`3`	if the entry is a duplicate.

Examples **EXAMPLE 1** A sample display of the `soladdapp` command.

The following adds an application called Disk Manager to the Solstice application registry for display in the Solstice Launcher main window.

```
# soladdapp -r /opt/SUNWadm/etc/.solstice_registry -n "Disk Manager"
-i /opt/SUNWdsk/etc/diskmgr.xpm -e /opt/SUNWdsk/bin/diskmgr
```

Files

`/opt/SUNWadm/etc/.solstice_registry`	The default registry path.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadml

See Also `soldelapp(1M)`, `solstice(1M)`, `attributes(5)`

Notes Globally registered applications are used by local and remote users sharing the software in a particular `/opt` directory. They can be added only using `soladdapp`.

Name soldelapp – remove an application from the Solstice application registry

Synopsis

```
/usr/snadm/bin/soldelapp [-r registry] -n name
```

Description `soldelapp` removes an application from the Solstice application registry. After removal, the application is no longer displayed in the Solstice Launcher main window (see `solstice(1M)`).

Options

`-r` *registry* Define the full path name of the Solstice registry file.

`-n` *name* Define the name of the tool to be removed.

When executed without options, `soldelapp` uses `/opt/SUNWadm/etc/.solstice_registry` (the default registry path).

Return Values

`0` on success

`1` on failure

`2` if the registry is locked

`3` if *name* is not found in the registry

`4` if the named registry or default registry is not found

Examples **EXAMPLE 1** A sample display of the `soldelapp` command.

The following removes an application called Disk Manager from the Solstice application registry and the Solstice Launcher main window.

```
# soldelapp -r /opt/SUNWadm/etc/.solstice_registry -n "Disk Manager"
```

Files `/opt/SUNWadm/etc/.solstice_registry` The default registry file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadml

See Also `soladdapp(1M)`, `solstice(1M)`, `attributes(5)`

Notes Globally registered applications are used by local and remote users sharing the software in a particular `/opt` directory. They can be removed only using `soldelapp`.

Name solstice – access system administration tools with a graphical user interface

Synopsis

```
/bin/solstice
```

Description `solstice` used on a system presents the `Solstice Launcher`, a graphical user interface that provides access to the Solstice AdminSuite product family of system administration tools. The tools that appear in the launcher depend on what Solstice products you installed on your system.

Help is available by using the Help button.

Usage The `Solstice Launcher` allows you to do the following tasks:

Launch applications	Use the `Solstice Launcher` to launch system administration tools.
Register applications	Use the `Solstice Launcher` to add and register applications locally with the launcher.
Remove applications	Use the `Solstice Launcher` to remove locally registered applications.
Customize application properties	Use the `Solstice Launcher` to show, hide, or remove applications in the launcher, reorder the icons, change the launcher window width, modify applications properties, and add applications.

Files `/$HOME/.solstice_registry` Local registry information.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsadml

See Also `soladdapp(1M)`, `soldelapp(1M)`, `attributes(5)`

Notes The `Solstice Launcher` adds or removes local applications that are private to the user (not local to the system) only. The properties of globally registered applications that are used by local and remote users sharing the software from a particular `/opt` directory cannot be modified from the `Solstice Launcher`. To register global applications for use by local and remote users, use the `soladdapp(1M)` command. To remove globally registered applications, use the `soldelapp(1M)` command.

Name sppptun – PPP tunneling driver utility

Synopsis

```
sppptun plumb

sppptun plumb protocol device

sppptun unplumb interface

sppptun query
```

Description The `sppptun` utility is used to configure and query the Solaris PPP tunneling device driver, `/dev/sppptun`. Currently, only PPP over Ethernet (PPPoE) is supported, so the `plumb` and `unplumb` arguments are used to specify Ethernet interfaces that are to be used for PPPoE, and the `query` option lists the plumbed interfaces.

The use of `sppptun` to add interfaces is similar to the use of `ifconfig(1M)` to add interfaces to IP. The plumbing is done once for each interface, preferably at system start-up time, and is not normally manipulated on a running system. If multiple instances of PPP are run over a single interface, they share the plumbing to that interface. Plumbing for each session is not required (and not possible for PPPoE).

The proper way to plumb interfaces for PPPoE is to list the interfaces, one per line, in the `/etc/ppp/pppoe.if` file.

Usage

`sppptun plumb`	When specified with no additional arguments, the plumb argument lists the protocols that are supported by the utility. These are the strings that are used as the *protocol* argument below.
`sppptun plumb` *protocol device*	This plumbs a new interface into the driver. The *protocol* parameter is `pppoe` for the PPP-carrying "Session Stage" connection or `pppoed` for the PPPoE "Discovery Stage" connection. Both connections must be present for each Ethernet interface that is to be used for PPPoE. The *device* parameter is the path name of the Ethernet interface to use (use `ifconfig(1M)` to list available devices). If the path begins with `/dev/`, then this portion may be omitted.
`sppptun unplumb` *interface*	This removes an existing interface from the driver and terminates any PPP sessions that were using the interface. The *interface* parameter is the name of the interface as reported when the interface was plumbed.
`sppptun query`	Displays the canonical names of all interfaces plumbed into the `/dev/sppptun` device driver.

Examples **EXAMPLE 1** Setting up to Use PPPoE on `hme0`

Plumb the `hme0` interface.

EXAMPLE 1 Setting up to Use PPPoE on `hme0` *(Continued)*

```
# sppptun plumb pppoed hme0
hme0:pppoed
# sppptun plumb pppoe hme0
hme0:pppoe
```

Remove the `hme0` interface.

```
# sppptun unplumb hme0:pppoed
# sppptun unplumb hme0:pppoe
```

EXAMPLE 2 Script to Remove All Plumbed Interfaces

```
#!/bin/sh
for intf in ‘sppptun query‘
do
        sppptun unplumb $intf
done
```

Exit Status The following exit values are returned:

`0` Successful completion.

`1` One or more errors occurred.

Files

`/etc/ppp/pppoe.if`	list of Ethernet interfaces to be plumbed at boot time
`/usr/sbin/sppptun`	executable command
`/dev/sppptun`	Solaris PPP tunneling device driver

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpppdt

See Also `pppd(1M)`, `pppoec(1M)`, `pppoed(1M)`, `sppptun(7M)`

RFC 2516, Method for Transmitting PPP Over Ethernet (PPPoE), Mamakos et al, February 1999

Name spray – spray packets

Synopsis

```
/usr/sbin/spray [-c count] [-d delay] [-l length]
     [-t nettype] host
```

Description `spray` sends a one-way stream of packets to *host* using RPC, and reports how many were received, as well as the transfer rate. The *host* argument can be either a name or an Internet address.

`spray` is not useful as a networking benchmark, as it uses unreliable connectionless transports, UDP for example. `spray` can report a large number of packets dropped when the drops were caused by `spray` sending packets faster than they can be buffered locally, that is, before the packets get to the network medium.

Options

`-c` *count* Specify how many packets to send. The default value of *count* is the number of packets required to make the total stream size 100000 bytes.

`-d` *delay* Specify how many microseconds to pause between sending each packet. The default is `0`.

`-l` *length* The *length* parameter is the numbers of bytes in the Ethernet packet that holds the RPC call message. Since the data is encoded using XDR, and XDR only deals with 32 bit quantities, not all values of *length* are possible, and `spray` rounds up to the nearest possible value. When *length* is greater than 1514, then the RPC call can no longer be encapsulated in one Ethernet packet, so the *length* field no longer has a simple correspondence to Ethernet packet size. The default value of *length* is `86 bytes`, the size of the RPC and UDP headers.

`-t` *nettype* Specify class of transports. Defaults to `netpath`. See `rpc(3NSL)` for a description of supported classes.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWrcmdc

See Also `rpc(3NSL)`, `attributes(5)`

Name sshd – secure shell daemon

Synopsis

```
sshd [-deiqtD46] [-b bits] [-f config_file]
     [-g login_grace_time] [-h host_key_file]
     [-k key_gen_time] [-p port] [-V client_protocol_id]
```

Description The `sshd` (Secure Shell daemon) is the daemon program for `ssh(1)`. Together these programs replace `rlogin` and `rsh`, and provide secure encrypted communications between two untrusted hosts over an insecure network. The programs are intended to be as easy to install and use as possible.

`sshd` is the daemon that listens for connections from clients. It forks a new daemon for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange.

This implementation of `sshd` supports both SSH protocol versions 1 and 2 simultaneously. Because of security weaknesses in the v1 protocol, sites should run only v2, if possible. In the default configuration, only protocol v2 is enabled for the server. To enable v1 and v2 simultaneously, see the instructions in `sshd_config(4)`.

Support for v1 is provided to help sites with existing `ssh` v1 clients and servers to transition to v2. v1 might not be supported in a future release.

SSH Protocol Version 1

Each host has a host-specific RSA key (normally 1024 bits) used to identify the host. Additionally, when the daemon starts, it generates a server RSA key (normally 768 bits). This key is normally regenerated every hour if it has been used, and is never stored on disk.

Whenever a client connects the daemon responds with its public host and server keys. The client compares the RSA host key against its own database to verify that it has not changed. The client then generates a 256–bit random number. It encrypts this random number using both the host key and the server key, and sends the encrypted number to the server. Both sides then use this random number as a session key which is used to encrypt all further communications in the session. The rest of the session is encrypted using a conventional cipher, currently Blowfish or 3DES, with 3DES being used by default. The client selects the encryption algorithm to use from those offered by the server.

Next, the server and the client enter an authentication dialog. The client tries to authenticate itself using `.rhosts` authentication, `.rhosts` authentication combined with RSA host authentication, RSA challenge-response authentication, or password-based authentication.

Rhosts authentication is normally disabled because it is fundamentally insecure, but can be enabled in the server configuration file if desired. System security is not improved unless `rshd(1M)`, `rlogind(1M)`, `rexecd(1M)`, and `rexd(1M)` are disabled (thus completely disabling `rlogin(1)` and `rsh(1)` into the machine).

SSH Protocol Version 2

Version 2 works similarly to version 1: Each host has a host-specific DSA/RSA key. However, when the daemon starts, it does not generate a server key. Forward security is provided through a Diffie-Hellman key agreement. This key agreement results in a shared session key. The rest of the session is encrypted using a symmetric cipher, currently 128-bit AES, Blowfish, 3DES, or AES. The client selects the encryption algorithm to use from those offered by the server. Additionally, session integrity is provided through a cryptographic message authentication code (`hmac-sha1` or `hmac-md5`).

Protocol version 2 provides a public key based user authentication method (PubKeyAuthentication) GSS–API based user authentication, conventional password authentication, and a generic prompt/reply protocol for password-based authentication.

Command Execution and Data Forwarding

If the client successfully authenticates itself, a dialog for preparing the session is entered. At this time the client can request things like allocating a pseudo-tty, forwarding X11 connections, forwarding TCP/IP connections, or forwarding the authentication agent connection over the secure channel.

Finally, the client either requests a shell or execution of a command. The sides then enter session mode. In this mode, either side may send data at any time, and such data is forwarded to/from the shell or command on the server side, and the user terminal on the client side.

When the user program terminates and all forwarded X11 and other connections have been closed, the server sends command exit status to the client, and both sides exit.

`sshd` can be configured using command-line options or the configuration file `/etc/ssh/ssh_config`, described in `ssh_config(4)`. Command-line options override values specified in the configuration file.

`sshd` rereads its configuration file when it receives a hangup signal, `SIGHUP`, by executing itself with the name it was started as, that is, `/usr/lib/ssh/sshd`.

Host Access Control

The `sshd` daemon uses TCP Wrappers to restrict access to hosts. It uses the service name of `sshd` for `hosts_access()`. For more information on TCP Wrappers see `tcpd(1M)` and `hosts_access(3)` man pages, which are part of the `SUNWsfman` package (they are not SunOS man pages). TCP wrappers binaries, including `libwrap`, are in `SUNWtcpd`, a required package for `SUNWsshdu`, the package containing `sshd`.

Options

The options for `sshd` are as follows:

`-b` *bits*
Specifies the number of bits in the server key (the default is 768).

`-d`
Debug mode. The server sends verbose debug output to the system log, and does not put itself in the background. The server also will not fork and will only process one connection. This option is only intended for debugging for the server. Multiple `-d` options increase the debugging level. Maximum is 3.

`-e`
When this option is specified, `sshd` will send the output to standard error instead of to the system log.

`-f` *configuration_file*
Specifies the name of the configuration file. The default is `/etc/ssh/sshd_config`. `sshd` refuses to start if there is no configuration file.

`-g` *login_grace_time*
Gives the grace time for clients to authenticate themselves (the default is 300 seconds). If the client fails to authenticate the user within this number of seconds, the server disconnects and exits. A value of zero indicates no limit.

`-h` *host_key_file*
Specifies a file from which a host key is read. This option must be given if `sshd` is not run as root (as the normal host key files are normally not readable by anyone but root). The default is `/etc/ssh/ssh_host_key` for protocol version 1, and `/etc/ssh/ssh_host_rsa_key` and `/etc/ssh/ssh_host_dsa_key` for protocol version 2. It is possible to have multiple host key files for the different protocol versions and host key algorithms.

`-i`
Specifies that `sshd` is being run from `inetd`. `sshd` is normally not run from `inetd` because it needs to generate the server key before it can respond to the client, and this may take tens of seconds. Clients would have to wait too long if the key was regenerated every time. However, with small key sizes (for example, 512) using `sshd` from `inetd` may be reasonable.

`-k` *key_gen_time*
(SSHv1-specific) Specifies how often the server key is regenerated (the default is 3600 seconds, or one hour). The motivation for regenerating the key fairly often is that the key is not stored anywhere, and after about an hour, it becomes impossible to recover the key for decrypting intercepted communications even if the machine is cracked into or physically seized. A value of zero indicates that the key will never be regenerated.

`-o` *option*
Can be used to specify options in the format used in the configuration file. This is useful for specifying options for which there are no separate command-line flags.

`-p` *port*
Specifies the port on which the server listens for connections (the default is 22).

`-q`
Quiet mode. Nothing is sent to the system log. Normally the beginning, authentication, and termination of each connection is logged.

`-t`
Test mode. Check only the validity of the configuration file and the sanity of the keys. This is useful for updating sshd reliably as configuration options might change.

`-D`
When this option is specified `sshd` does not detach and does not become a daemon. This allows easy monitoring of `sshd`.

`-4`
Forces `sshd` to use IPv4 addresses only.

`-6`
Forces `sshd` to use IPv6 addresses only.

Extended Description

`authorized_keys` File Format

The `$HOME/.ssh/authorized_keys` file lists the public keys that are permitted for RSA authentication in protocol version 1 and for public key authentication (`PubkeyAuthentication`) in protocol version 2. The `AuthorizedKeysFile` configuration option can be used to specify an alternative file.

Each line of the file contains one key (empty lines and lines starting with a hash mark [#] are ignored as comments).

For each RSA key for protocol version 1, the file consists of the following space-separated fields:

options bits exponent modulus comment

For the public key for protocol version 2, the file consists of the following space-separated fields:

options key-type base64-encoding-key comment

For protocol version 2, *key-type* is one of `ssh-rsa` or `ssh-dsa`.

The options field is optional; its presence is determined by whether the line starts with a number. (The option field never starts with a number.) The bits, exponent, and modulus fields give the RSA key; the comment field is a convenient place for you to identify the key.

Lines in this file are usually several hundred bytes long (because of the size of the key modulus). You will find it very inconvenient to type them in; instead, copy the public key file and edit it.

Permissions of this file must be set so that it is not world or group writable. See the `StrictModes` option of `sshd_config(4)`.

The options (if present) consist of comma-separated option specifications. No spaces are permitted, except within double quotes. The following option specifications are supported:

`from="`*pattern-list*`"`
Specifies that, in addition to public key authentication, the canonical name of the remote host must be present in the comma-separated list of patterns ('`*`' and '`?`' serve as wildcards). The list can also contain negated patterns by prefixing the patterns with '`!`'. If the canonical host name matches a negated pattern, the key is not accepted.

The purpose of this option is to give you the option of increasing security: public key authentication by itself does not trust the network or name servers or anything but the key. However, if someone manages to steal the key, possession of the key would permit the intruder to log in from anywhere in the world. This option makes using a stolen key more difficult, because name servers and routers would have to be compromised, in addition to just the key.

`command="`*command*`"`
Specifies that the *command* is executed whenever this key is used for authentication. The command supplied by the user (if any) is ignored. The command is run on a `pty` if the client requests a `pty`; otherwise it is run without a `tty`. If an 8-bit clean channel is required, one must not request a `pty` or should specify `no-pty`. You can include a quote in the command by escaping it with a backslash. This option might be useful to restrict certain public keys from performing a specific operation. An example is a key that permits remote backups but nothing else. Note that the client can specify TCP/IP and/or X11 forwarding unless they are explicitly prohibited from doing so. Also note that this option applies to shell, command, or subsystem execution.

`environment="`*NAME*`=`*value*`"`
Specifies that the string *NAME*`=`*value* is to be added to the environment when logging in using this key. Environment variables set this way override other default environment values. Multiple options of this type are permitted. Environment processing is disabled by default and is controlled via the `PermitUserEnvironment` option.

`no-port-forwarding`
Forbids TCP/IP forwarding when this key is used for authentication. Any port forward requests by the client will return an error. This might be used, for example, in connection with the `command` option.

`no-X11-forwarding`
Forbids X11 forwarding when this key is used for authentication. Any X11 forward requests by the client will return an error.

`no-agent-forwarding`
Forbids authentication agent forwarding when this key is used for authentication.

`no-pty`
Prevents `tty` allocation (a request to allocate a `pty` will fail).

`permitopen="`*host*`:`*port*`"`
Limit local `ssh -L` port forwarding such that it can connect only to the specified host and port. IPv6 addresses can be specified with an alternative syntax: *host*/*port*. You can invoke multiple `permitopen` options, with each instance separated by a comma. No pattern matching is performed on the specified hostnames. They must be literal domains or addresses.

ssh_known_hosts File Format

The `/etc/ssh/ssh_known_hosts` and `$HOME/.ssh/known_hosts` files contain host public keys for all known hosts. The global file should be prepared by the administrator (optional), and the per-user file is maintained automatically: whenever the user connects from an unknown host its key is added to the per-user file.

For the RSA key for protocol version 1, these files consist of the following space-separated fields:

hostnames bits exponent modulus comment

For the public key for protocol version 2, these files consist of the following space-separated fields:

hostnames key-type base64-encoding-key comment

For protocol version 2, *key-type* is one of `ssh-rsa` or `ssh-dsa`.

Hostnames is a comma-separated list of patterns (* and ? act as wildcards); each pattern in turn is matched against the canonical host name (when authenticating a client) or against the user-supplied name (when authenticating a server). A pattern can also be preceded by ! to indicate negation: if the host name matches a negated pattern, it is not accepted (by that line) even if it matched another pattern on the line.

Alternately, hostnames can be stored in a hashed form, which hides host names and addresses should the file's contents be disclosed. Hashed hostnames start with a vertical bar (|) character. Only one hashed hostname can appear on a single line and none of the above negation or wildcard operators may be applied.

Bits, exponent, and modulus are taken directly from the RSA host key; they can be obtained, for example, from `/etc/ssh/ssh_host_rsa_key.pub`. The optional comment field continues to the end of the line, and is not used.

Lines starting with a hash mark (#) and empty lines are ignored as comments.

When performing host authentication, authentication is accepted if any matching line has the proper key. It is thus permissible (but not recommended) to have several lines or different host keys for the same names. This will inevitably happen when short forms of host names from different domains are put in the file. It is possible that the files contain conflicting information; authentication is accepted if valid information can be found from either file.

The lines in these files are typically hundreds of characters long. You should definitely not type in the host keys by hand. Rather, generate them by a script or by taking `/etc/ssh/ssh_host_rsa_key.pub` and adding the host names at the front.

Environment Variables

`sshd` sets the following environment variables for commands executed by `ssh` users:

`DISPLAY`
: Indicates the location of the X11 server. It is automatically set by `sshd` to point to a value of the form *hostname*:*n*, where *hostname* indicates the host where the shell runs, and *n* is an integer greater than or equal to 1. `ssh` uses this special value to forward X11 connections

over the secure channel. Unless you have important reasons to do otherwise, you should not set DISPLAY explicitly, as that will render the X11 connection insecure and will require you to manually copy any required authorization cookies.

HOME
Set to the path of the user's home directory.

LANG, LC_ALL, LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, LC_TIME
A locale setting. The locale defaults to that of sshd (usually the system-wide default locale), or is negotiated between the client and server during initial key exchange (as per RFC 4253).

Following initial key exchange, each of the variables can be overriden in the following sequence:

1. If a locale setting is set in a client's environment and that client supports "Environment Variable Passing" (see RFC 4254), then the setting will be passed over to the server side.
2. If the public key authentication method was used to authenticate the server and the PermitUserEnvironment variable in sshd_config(4) is set to yes on the server side, then the setting can be changed through the use of the environment option in the client's AuthorizedKeysFile file.
3. The setting can be change in the client's ~/.ssh/environment file on the server.

See PermitUserEnvironment in sshd_config(4) as to when the AuthorizedKeysFile and ~/.ssh/environment files are processed and used for setting the user environment.

LOGNAME
Synonym for USER. Set for compatibility with systems that use this variable.

MAIL
Set to point to the user's mailbox.

SSH_AUTH_SOCK
Indicates the path of a unix-domain socket used to communicate with the agent.

SSH_CONNECTION
Identifies the client and server ends of the connection. The variable contains four space-separated values: client IP address, client port number, server IP address and server port number.

SSH_CLIENT
Identifies the client end of the connection. The variable contains three space-separated values: client IP address, client port number, and server port number.

SSH_TTY
Set to the name of the tty (path to the device) associated with the current shell or command. If the current session has no tty, this variable is not set.

`TZ`
Indicates the present timezone, if `TIMEZONE` is set in `/etc/default/login` or if `TZ` was set when the daemon was started.

`HZ`
If set in `/etc/default/login`, the daemon sets it to the same value.

`SHELL`
The user's shell, if `ALTSHELL=YES` in `/etc/default/login`.

`PATH`
Set to the value of `PATH` or `SUPATH` (see `login(1)`) in `/etc/default/login`, or, if not set, to `/usr/bin:/bin`.

`USER`
Set to the name of the user logging in.

Additionally, `sshd` reads `$HOME/.ssh/environment` and adds lines of the format `VARNAME=`*value* to the environment.

Examples

In the following examples, certain lines might wrap due to line length limits in your display. You should nevertheless consider the wrapped line as a single line.

EXAMPLE 1 `authorized_key` File Entries

The following are examples of `authorized_key` file entries for protocol 1:

```
1024 33 12121...312314325 ylo@foo.bar

from="*.niksula.hut.fi,!pc.niksula.hut.fi" 1024 35 23...2334 ylo@niksula

command="dump /home",no-pty,no-port-forwarding 1024 33 23...2323 backup.hut.fi
```

EXAMPLE 2 `authorized_key` File Entries for Protocol 2

The following are examples of `authorized_key` file entries for protocol 2:

```
ssh-rsa AAAAB3NzaC1y.....EU88ovYKg4GfclWGCFYTuw8= ylo@foo.bar
from="*.niksula.hut.fi" ssh-rsa AAAAB3NzaC...uw8= ylo@niksula
command="dump /home",no-pty,no-port-forwarding ssh-rsa AA..8= backup.hut.fi
```

EXAMPLE 3 `ssh_known_hosts` File Entries for Protocol 1

The following are examples of `ssh_known_hosts` file entries for protocol 1:

```
closenet,closenet.hut.fi,...,130.233.208.41 1024 37 159...93 closenet.hut.fi
```

EXAMPLE 4 `ssh_known_hosts` File Entries for Protocol 2

The following are examples of `ssh_known_hosts` file entries for protocol 2:

```
closenet,closenet.hut.fi,...,130.233.208.41 ssh-rsa AA..8= closenet.hut.fi
```

Exit Status The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

Files `/etc/default/login`
Contains defaults for several `sshd_config` parameters, environment variables, and other environmental factors.

The following parameters affect environment variables (see `login(1)` and descriptions of these variables, above):

- `TIMEZONE`
- `HZ`
- `ALTSHELL`
- `PATH`
- `SUPATH`

The following `/etc/default/login` parameters supply default values for corresponding `sshd_config(4)` parameters:

- `CONSOLE` (see `PermitRootLogin` in `sshd_config(4)`)
- `PASSREQ` (see `PermitEmptyPasswords` in `sshd_config(4)`)
- `TIMEOUT` (see `LoginGraceTime` in `sshd_config(4)`)

The following `/etc/default/login` parameters:

- `UMASK`
- `ULIMIT`

...set the `umask(2)` and file size limit of, respectively, the shells and commands spawned by `sshd`.

Finally, two `/etc/default/login` parameters affect the maximum allowed login attempts per-connection using interactive user authentication methods (for example, `keyboard-interactive` but not `publickey`), as per `login(1)`:

- `RETRIES`
- `SYSLOG_FAILED_LOGINS`

`/etc/ssh/sshd_config`
Contains configuration data for `sshd`. This file should be writable by root only, but it is recommended (though not necessary) that it be world-readable.

`/etc/ssh/ssh_host_key`
`/etc/ssh/ssh_host_dsa_key`
`/etc/ssh/ssh_host_rsa_key`
Contains the private part of the host key. This file should only be owned by root, readable only by root, and not accessible to others. `sshd` does not start if this file is group/world-accessible.

`/etc/ssh/ssh_host_key.pub`
`/etc/ssh/ssh_host_dsa_key.pub`
`/etc/ssh/ssh_host_rsa_key.pub`
Contains the public part of the host key. This file should be world-readable but writable only by root. Its contents should match the private part. This file is not used for encryption; it is provided only for the convenience of the user so its contents can be copied to known hosts files. These two files are created using `ssh-keygen(1)`.

`/var/run/sshd.pid`
Contains the process ID of the `sshd` listening for connections. If there are several daemons running concurrently for different ports, this contains the pid of the one started last. The content of this file is not sensitive; it can be world-readable. You can use the `PidFile` keyword in `sshd_config` to specify a file other than `/var/run/sshd.pid`. See `sshd_config(4)`.

`/etc/ssh/ssh_known_hosts` and `$HOME/.ssh/known_hosts`
These files are consulted when using `rhosts` with public key host authentication to check the public key of the host. The key must be listed in one of these files to be accepted. The client uses the same files to verify that the remote host is the one it intended to connect. These files should be writable only by root or the owner. `/etc/ssh/ssh_known_hosts` should be world-readable, and `$HOME/.ssh/known_hosts` can but need not be world-readable.

`/etc/nologin`
If this file exists, `sshd` refuses to let anyone except root log in. The contents of the file are displayed to anyone trying to log in, and non-root connections are refused. The file should be world-readable.

`$HOME/.ssh/authorized_keys`
Lists the public keys (RSA or DSA) that can be used to log into the user's account. This file must be readable by root. This might, on some machines, imply that it is world-readable if the user's home directory resides on an NFS volume. It is recommended that it not be accessible by others. The format of this file is described above. Users will place the contents of their `identity.pub`, `id_dsa.pub` and/or `id_rsa.pub` files into this file, as described in `ssh-keygen(1)`.

`$HOME/.rhosts`
This file contains host-username pairs, separated by a space, one per line. The given user on the corresponding host is permitted to log in without password. The same file is used by `rlogind` and `rshd`. The file must be writable only by the user; it is recommended that it not be accessible by others. It is also possible to use `netgroups` in the file. Either host or user name may be of the form +@*groupname* to specify all hosts or all users in the group.

`$HOME/.shosts`
For `ssh`, this file is exactly the same as for `.rhosts`. However, this file is not used by `rlogin` and `rshd`, so using this permits access using SSH only.

`/etc/hosts.equiv`
This file is used during `.rhosts` authentication. In its simplest form, this file contains host names, one per line. Users on these hosts are permitted to log in without a password, provided they have the same user name on both machines. The host name can also be followed by a user name; such users are permitted to log in as any user on this machine (except root). Additionally, the syntax *+@group* can be used to specify netgroups. Negated entries start with a hyphen (`-`).

If the client host/user is successfully matched in this file, login is automatically permitted, provided the client and server user names are the same. Additionally, successful RSA host authentication is normally required. This file must be writable only by root; it is recommended that it be world-readable.

Warning: It is almost never a good idea to use user names in `hosts.equiv`. Beware that it really means that the named user(s) can log in as anybody, which includes `bin`, `daemon`, `adm`, and other accounts that own critical binaries and directories. For practical purposes, using a user name grants the user root access. Probably the only valid use for user names is in negative entries. This warning also applies to `rsh`/`rlogin`.

`/etc/ssh/moduli`
A private file.

`/etc/ssh/shosts.equiv`
This file is processed exactly as `/etc/hosts.equiv`. However, this file might be useful in environments that want to run both `rsh`/`rlogin` and `ssh`.

`$HOME/.ssh/environment`
This file is read into the environment at login (if it exists). It can contain only empty lines, comment lines (that start with `#`), and assignment lines of the form *name*=*value*. The file should be writable only by the user; it need not be readable by anyone else. Environment processing is disabled by default and is controlled by means of the `PermitUserEnvironment` option.

`$HOME/.ssh/rc`
If this file exists, it is run with `/bin/sh` after reading the environment files but before starting the user's shell or command. If X11 spoofing is in use, this will receive the `proto cookie` pair in standard input (and `DISPLAY` in environment). This must call `xauth` in that case.

The primary purpose of `$HOME/.ssh/rc` is to run any initialization routines that might be needed before the user's home directory becomes accessible; AFS is a particular example of such an environment. If this file exists, it is run with `/bin/sh` after reading the environment files, but before starting the user's shell or command. It must not produce any output on stdout; stderr must be used instead. If X11 forwarding is in use, it will receive the `proto cookie` pair in its standard input and `DISPLAY` in its environment. The script must call `xauth` because `sshd` will not run `xauth` automatically to add X11 cookies.

This file will probably contain some initialization code followed by something similar to:

```
if read proto cookie && [ -n "$DISPLAY" ]
then
  if [ `echo $DISPLAY | cut -c1-10`  =  'localhost:' ]
  then
    # X11UseLocalhost=yes
    echo add unix:`echo $DISPLAY |
    cut -c11-` $proto $cookie
  else
    # X11UseLocalhost=no
    echo add $DISPLAY $proto $cookie
  fi | xauth -q -
fi
```

If this file does not exist, `/etc/ssh/sshrc` is run, and if that does not exist, `xauth` is used to store the cookie. `$HOME/.ssh/rc` should be writable only by the user, and need not be readable by anyone else.

`/etc/ssh/sshrc`
Similar to `$HOME/.ssh/rc`. This can be used to specify machine-specific login-time initializations globally. This file should be writable only by root, and should be world-readable.

Security `sshd` supports the use of several user authentication mechanisms: a public key system where keys are associated with users (through users' `authorized_keys` files), a public key system where keys are associated with hosts (see the `HostbasedAuthentication` configuration parameter), a GSS-API based method (see the `GssAuthentication` and `GssKeyEx` configuration parameters) and three initial authentication methods: `none`, `password`, and a generic prompt/reply protocol, `keyboard-interactive`.

`sshd` negotiates the use of the GSS-API with clients only if it has a GSS-API acceptor credential for the "host" service. This means that, for GSS-API based authentication, the server must have a Kerberos V `keytab` entry (see below) or the equivalent for any other GSS-API mechanism that might be installed.

In order for Kerberos authentication to work, a `host/`<*FQDN*> Kerberos principal must exist for each Fully Qualified Domain Name associated with the `in.sshd` server. Each of these `host/`<*FQDN*> principals must have a `keytab` entry in the `/etc/krb5/krb5.keytab` file on the `in.sshd` server. An example principal might be:

```
host/bigmachine.eng.example.com
```

See `kadmin`(1M) or `gkadmin`(1M) for instructions on adding a principal to a `krb5.keytab` file. See *System Administration Guide: Security Services* for a discussion of Kerberos authentication.

GSS-API authorization is covered in `gss_auth_rules`(5).

`sshd` uses `pam`(3PAM) for the three initial authentication methods as well as for account management, session management, and password management for all authentication methods.

Specifically, `sshd` calls `pam_authenticate()` for the "none," "password" and "keyboard-interactive" SSHv2 `userauth` types, as well as for for the null and password authentication methods for SSHv1. Other SSHv2 authentication methods do not call `pam_authenticate()`. `pam_acct_mgmt()` is called for each authentication method that succeeds.

`pam_setcred()` and `pam_open_session()` are called when authentication succeeds and `pam_close_session()` is called when connections are closed.

`pam_open_session()` and `pam_close_session()` are also called when SSHv2 channels with `ptys` are opened and closed.

Each SSHv2 `userauth` type has its own PAM service name:

SSHv2 Userauth	PAM Service Name
none	sshd-none
password	sshd-password
keyboard-interactive	sshd-kbdint
pubkey	sshd-pubkey
hostbased	sshd-hostbased
gssapi-with-mic	sshd-gssapi
gssapi-keyex	sshd-gssapi

For SSHv1, `sshd -v1` is always used.

If `pam_acct_mgmt()` returns `PAM_NEW_AUTHTOK_REQD` (indicating that the user's authentication tokens have expired), then `sshd` forces the use of "keyboard-interactive" `userauth`, if version 2 of the protocol is in use. The "keyboard-interactive" `userauth` will call `pam_chauthtok()` if `pam_acct_mgmt()` once again returns `PAM_NEW_AUTHTOK_REQD`. By this means, administrators are able to control what authentication methods are allowed for SSHv2 on a per-user basis.

Setting up Host-based Authentication

To establish host-based authentication, you must perform the following steps:

- Configure the client.
- Configure the server.
- Publish known hosts.
- Make appropriate entries in `/etc/ssh/shosts.equiv` and `~/.shosts`.

These steps are expanded in the following paragraphs.

- On a client machine, in the system-wide client configuration file, `/etc/ssh/ssh_config`, you must have the entry:

  ```
  HostbasedAuthentication yes
  ```

  See `ssh_config(4)` and `ssh-keysign(1M)`.
- On the server, in the system-wide server configuration file, `/etc/ssh/sshd_config`, you must have the entry:

  ```
  HostbasedAuthentication yes
  ```

  If per-user `.shost` files are to be allowed (see last step), in the same file, you must have:

  ```
  IgnoreRhosts no
  ```

  See `sshd_config(4)` for a description of these keywords.
- To publish known hosts, you must have entries for the clients from which users will be allowed host-based authentication. Make these entries in either or both of the system-wide file (`/etc/ssh/ssh_known_hosts`) or the per-user file (`~/.ssh/known_hosts`).
- Note that `sshd` uses `.shosts`, not `.rhosts`. If you want the functionality provided by `.rhosts`, but do not want to use `rlogin` or `rsh` because of their security shortcomings, you can use `.shosts` in conjunction with `sshd`. To use this feature, make appropriate entries in `/etc/ssh/shosts.equiv` and `~/.shosts`, in the format specified in `rhosts(4)`.

  For the vast majority of network environments, `.shosts` is preferred over `.rhosts`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsshdu
Interface Stability	Evolving

The interface stability of `/etc/ssh/moduli` is Private.

See Also `login(1)`, `scp(1)`, `ssh(1)`, `ssh-add(1)`, `ssh-agent(1)`, `ssh-keygen(1)`, `svcs(1)`, `gkadmin(1M)`, `kadmin(1M)`, `sftp-server(1M)`, `ssh-keysign(1M)`, `svcadm(1M)`, `pam(3PAM)`, `rhosts(4)`, `ssh_config(4)`, `sshd_config(4)`, `attributes(5)`, `gss_auth_rules(5)`, `kerberos(5)`, `pam_roles(5)`, `smf(5)`

System Administration Guide: Security Services

Notes The `sshd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/ssh:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

sshd always sets PAM_RHOST and sets PAM_AUSER in the case of host-based userauth. This behavior allows for remote logins to roles using host-based authentication. See pam_roles(5).

Name ssh-keysign – ssh helper program for host-based authentication

Synopsis

```
ssh-keysign
```

Description `ssh-keysign` is used by `ssh(1)` to access the local host keys and generate the digital signature required during host-based authentication with SSH protocol version 2. This signature is of data that includes, among other items, the name of the client host and the name of the client user.

`ssh-keysign` is disabled by default and can be enabled only in the global client configuration file `/etc/ssh/ssh_config` by setting `HostbasedAuthentication` to `yes`.

`ssh-keysign` is not intended to be invoked by the user, but from `ssh`. See `ssh(1)` and `sshd(1M)` for more information about host-based authentication.

Files

`/etc/ssh/ssh_config`	Controls whether `ssh-keysign` is enabled.
`/etc/ssh/ssh_host_dsa_key` `/etc/ssh/ssh_host_rsa_key`	These files contain the private parts of the host keys used to generate the digital signature. They should be owned by root, readable only by root, and not accessible to others. Because they are readable only by root, `ssh-keysign` must be `set-uid` root if host-based authentication is used.

Security ssh-keysign will not sign host-based authentication data under the following conditions:

- If the `HostbasedAuthentication` client configuration parameter is not set to `yes` in `/etc/ssh/ssh_config`. This setting cannot be overriden in users' `~/.ssh/ssh_config` files.
- If the client hostname and username in `/etc/ssh/ssh_config` do not match the canonical hostname of the client where `ssh-keysign` is invoked and the name of the user invoking `ssh-keysign`.

In spite of `ssh-keysign`'s restrictions on the contents of the host-based authentication data, there remains the ability of users to use it as an avenue for obtaining the client's private host keys. For this reason host-based authentication is turned off by default.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsshu
Interface Stability	Evolving

See Also `ssh(1)`, `sshd(1M)`, `ssh_config(4)`, `attributes(5)`

Authors Markus Friedl, `markus@openbsd.org`

History `ssh-keysign` first appeared in Ox 3.2.

Name statd – network status monitor

Synopsis

```
/usr/lib/nfs/statd
```

Description `statd` is an intermediate version of the status monitor. It interacts with `lockd`(1M) to provide the crash and recovery functions for the locking services on NFS. `statd` keeps track of the clients with processes which hold locks on a server. When the server reboots after a crash, `statd` sends a message to the `statd` on each client indicating that the server has rebooted. The client `statd` processes then inform the `lockd` on the client that the server has rebooted. The client `lockd` then attempts to reclaim the lock(s) from the server.

`statd` on the client host also informs the `statd` on the server(s) holding locks for the client when the client has rebooted. In this case, the `statd` on the server informs its `lockd` that all locks held by the rebooting client should be released, allowing other processes to lock those files.

`lockd` is started by `automountd`(1M), `mount_nfs`(1M), and `share`(1M) if NFS automounts are needed.

Files

`/var/statmon/sm`	lists hosts and network addresses to be contacted after a reboot
`/var/statmon/sm.bak`	lists hosts and network addresses that could not be contacted after last reboot
`/var/statmon/state`	includes a number which changes during a reboot
`/usr/include/rpcsvc/sm_inter.x`	contains the rpcgen source code for the interface services provided by the statd daemon.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfscu

See Also `svcs`(1), `automountd`(1M), `lockd`(1M), `mount_nfs`(1M), `share`(1M), `svcadm`(1M), `attributes`(5), `smf`(5)

System Administration Guide: IP Services

Notes The crash of a server is only detected upon its recovery.

The `statd` service is managed by the service management facility, `smf`(5), under the service identifier:

```
svc:/network/nfs/status
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

If it is disabled, it will be enabled by mount_nfs(1M), share_nfs(1M), and automountd(1M) unless its application/auto_enable property is set to false.

Name stclient – Service Tag Administration Program

Synopsis

```
stclient -x [-r root_dir]

stclient -a [-i instance_URN] -p product_name -e product_version
     -t product_URN [-F parent_URN] -P product_parent
     [-I product_defined_instance_id] -m product_vendor -A platform_arch
     -z container -S source [-r root_dir]

stclient -u -i instance_URN -F parent_URN -I product_defined_instance_id
     [-r root_dir]

stclient -d -i instance_URN [-r root_dir]

stclient -g -i instance_URN [-r root_dir]

stclient -f -t product_URN [-r root_dir]

stclient -h

stclient -v
```

Description The `stclient` command displays, finds, adds, updates and deletes records in the Service Tag registry. The registry is in the XML file `/var/sadm/servicetag/registry/servicetag.xml`, and contains the inventory of the product instances installed in the machine. Each record has a unique product instance identifier which is generated when the service tag is added in the registry. This product instance identifier is used as a key when finding, updating or deleting the service tag records. The extract option prints out the registry contents in XML format in stdout.

The `stclient` command also runs in interactive mode. This mode is invoked by running `stclient` without any parameters. A menu of all the available options are displayed, and the user is prompted to enter different parameters depending on the option chosen.

Any user can extract or get the contents of the registry, but only users with the appropriate privileges, the "svctag" user, or the creator of the service tag is authorized to update or delete a service tag record.

Options The following options are supported:

`-p` *product_name*	Sets the product name of the service tag to be added. For example, "stclient -p "Solaris 10 Operating System"" sets the product name to the "Solaris 10 Operating System".
`-e` *product_version*	Sets the product version of the service tag to be added. For example, "stclient -r 5.10" sets the product version to "5.10".
`-t` *product_URN*	Sets the Sun product unique identifier of the service tag to be added. For example, "stclient -t urn:uuid:5005588c-36f3-11d6-9cec-fc96f718e113" sets

	the Sun product unique identifier to "urn:uuid:5005588c-36f3-11d6-9cec-fc96f718e113".
`-i` *instance_URN*	Sets the product instance unique identifier of the service tag. For example, "stclient -i 3a444ab2-1dd2-11b2-a69d-830020782a6b" sets the product instance unique identifier to "3a444ab2-1dd2-11b2-a69d-830020782a6b" This field may also be set when the service tag is added but is not typically used. The generation of the instance ID is normally left to the `stclient`.
`-F` *product_parent_URN*	Sets the Sun product unique identifier of parent product. For example, "stclient -F urn:uuid:5005588c-36f3-11d6-9cec-fc96f718e113" sets the Sun product unique identifier of parent product to "urn:uuid:5005588c-36f3-11d6-9cec-fc96f718e113".
`-P` *product_parent*	Sets the parent product of the service tag. For example, "stclient -P JES" sets the product parent to "JES".
`-I` *product_defined_instance_id*	Sets the product defined instance identifier. For example, "stclient -I ser:abc-klm-000-7190" sets the product defined instance identifier to "ser:abc-klm-000-7190".
`-m` *product_vendor*	Sets the vendor of the product. For example, "stclient –m Sun" sets the product vendor to "Sun".
`-A` *platform_arch*	Sets the platform architecture that the product is running on. For example, "stclient –A sparc" sets the platform architecture to "sparc"
`-z` *container*	Sets the container that the product is running on. For example, "stclient -z "global zone"" sets the container to "global zone".
`-S` *source*	Sets the source of this service tag, naming the entity that created it.
`-r` *root_dir*	Sets the root directory where the command searches for the registry. The command searches `/var/sadm/servicetag/registry` by default. When this option is used, the command looks for the Registry in the specified root directory, (for example, "`-r /home/user1`" searches `/home/user1/var/sadm/servicetag`). This is typically used for testing.

Operands

Function Letters The function portion of the key is specified by one of the following letters:

x Extract. Extracts and prints the contents of the Service Tag registry in XML format. An alternate root directory may be specified for testing purposes.

a Add. Adds a service tag in the registry. A unique product instance identifier may be supplied and is automatically generated if not provided. This field is returned by the command. The required fields for add are product name, product version, product URN, product parent, vendor, platform architecture, container and source.

u Update. Updates a service tag in the registry using the product instance URN as the key. The parent URN and product defined instance id fields can be updated.

d Delete. Deletes a service tag in the registry using the product instance URN as the key.

E Extract Environment. Enumerates to standard output the environmental or "agent"' information associated with the registered Service Tags on this system.

g Get. Gets and prints a service tag from the registry using the product instance URN as the key.

f Find. Finds and prints all the instance URNs of a given product URN.

h Help. Displays the command options.

v Version. Displays the version number of the command.

Exit Status

The following exit values are returned:

`0` Successful completion.

`> 0` An error occurred.

Examples

EXAMPLE 1 Adding a Service Tag in the Registry

To add a service tag, enter the following:

```
# stclient -a -p "Java Enterprise Web Server 6.0" -e 6.0 \
   -t urn:uuid:f2b8b59c-6eba-11d7-986f-9f5d47ec72fe  \
   -P Java Enterprise Server -m Sun -A sparc -z global -s patch
```

The screen displays the following:

```
Java Enterprise Web Server 6.0 6.0 added
Product instance URN=urn:st:7fc61914-1dd2-11b2-b992-830020782a6b
```

EXAMPLE 2 Updating a Service Tag in the Registry

To update a service tag, enter the following:

EXAMPLE 2 Updating a Service Tag in the Registry *(Continued)*

```
# stclient -u -i 7fc61914-1dd2-11b2-b992-830020782a6b \
   -I urn:st:product.defined.id
```

The screen displays the following:

```
Service tag updated
```

EXAMPLE 3 Extracting a Service Tag Registry

To extract a service tag, enter the following:

```
# stclient -x
```

The screen displays output similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<registry urn="urn:uuid:1234ab-00e1-11b3-98737646873" version="1.0">
   <service_tag>
  .
  .
  .
   </service_tag>
 </registry>
```

EXAMPLE 4 Finding all product instances

To find all product instances, enter the following:

```
# stclient -f -t urn:uuid:f2b8b59c-6eba-11d7-986f-9f5d47ec72fe \
   fc61914-1dd2-11b2-b992-830020782a6b
```

EXAMPLE 5 Listing the Environmental Information

To list the environmental information associated with the registered Service Tags on this system, enter the following:

```
# stclient -E
```

The screen displays output similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<agent>
   <agent_urn>urn:st:af15ee62-0bb3-ef2d-fa96-85a11996cc71</agent_urn>
  .
  .
  .
   </system_info>
 </agent>
```

Environment Variables See environ(5)for descriptions of the following environment variables that affect the execution of `stclient`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWservicetagu
Interface Stability	Committed

See Also in.stdiscover(1M), in.stlisten(1M)

Name stmsboot – administration program for the Solaris I/O multipathing feature

Synopsis

```
/usr/sbin/stmsboot [[-D (fp | mpt | mpt_sas | iscsi) ] -d | -e | -u]
      | -L | -l controller_number]
```

Description The Solaris I/O multipathing feature is a multipathing solution for storage devices that is part of the Solaris operating environment. This feature was formerly known as Sun StorEdge Traffic Manager (STMS) or MPxIO.

The `stmsboot` program is an administrative command to manage enumeration of multipath–capable devices with Solaris I/O multipathing. Solaris I/O multipathing-enabled devices are enumerated under `scsi_vhci(7D)`, providing multipathing capabilities. Solaris I/O multipathing-disabled devices are enumerated under the physical controller.

In the `/dev` and `/devices` trees, Solaris I/O multipathing-enabled devices receive new names that indicate that they are under Solaris I/O multipathing control. This means a device will have a different name from its original name (after enabling) when it is under Solaris I/O multipathing control. The `stmsboot` command automatically updates `/etc/vfstab` and dump configuration to reflect the device names changes when enabling or disabling Solaris I/O multipathing. One reboot is required for changes to take effect.

Options The following options are supported:

`-e [ -E fp | mpt | mpt_sas | iscsi ]`
Enables Solaris I/O multipathing on all supported multipath-capable controller ports, including `fp(7d)`, `mpt(7D)`, `mpt_sas(7D)`, and `iscsi(7D)` port drivers. Multipath-capable ports include fibre channel (`fp(7d)`) controller ports and SAS (`mpt(7D)` or `mpt_sas(7D)`) controller ports. Following this enabling, you are prompted to reboot. During the reboot, `vfstab` and the dump configuration will be updated to reflect the device name changes. Specifying `-D mpt`, `-D mpt_sas`, or `-D fp` limits the enabling operation to ports attached using the specified driver.

`-d [ -D fp | mpt | mpt_sas | iscsi ]`
Disables Solaris I/O multipathing on all supported multipath-capable controller ports, including `fp(7d)`, `mpt(7D)`, `mpt_sas(7D)`, and `iscsi(7D)` port drivers. Multipath-capable ports include fibre channel (`fp(7d)`) controller ports and SAS (`mpt(7D)` or `mpt_sas(7D)`) controller ports. Following this disabling, you are prompted to reboot. During the reboot, `vfstab` and the dump configuration will be updated to reflect the device name changes. Specifying `-D mpt`, `-D mpt_sas`, or `-D fp` limits the disabling operation to ports attached using the specified driver.

`-u [ -D fp | mpt | mpt_sas | iscsi ]`
Updates `vfstab` and the dump configuration after you have manually modified the configuration to have Solaris I/O multipathing enabled or disabled on specific `fp(7d)`, `mpt(7D)`, `mpt_sas(7D)`, and `iscsi(7D)` controller ports. This option prompts you to reboot. During the reboot, `vfstab` and the dump configuration will be updated to reflect the device name changes.

`-L`
Display the device name changes from non-Solaris I/O multipathing device names to Solaris I/O multipathing device names for multipath-enabled controller ports. If Solaris I/O multipathing is not enabled, then no mappings are displayed.

`-l` *controller_number*
Display the device name changes from non-Solaris I/O multipathing device names to Solaris I/O multipathing device names for the specified controller. If Solaris I/O multipathing is not enabled, then no mappings are displayed.

Note that `mpt_sas(7D)` has MPxIO turned on by default. This means that when using the `-L` or `-l` option with `-D mpt_sas`, `stmsboot` does not display any non-multipathed and multipathed device names.

Usage

The primary function of `stmsboot` is to control the enabling and disabling of Solaris I/O multipathing on the host. The utility automatically updates `vfstab(4)` and `dumpadm(1M)` configuration to reflect device name changes. The system administrator is responsible for modifying application configuration (for example, backup software, DBMS, and so forth) to reflect updated device names.

The `-L` and `-l` options display the mapping between multipathed and non-multipathed device names. These options function only after changes to the Solaris I/O multipathing configuration have taken effect, that is, following the reboot after invoking `stmsboot -e`.

ZFS datasets, including ZFS root datasets, are correctly handled by `stmsboot`.

Examples

EXAMPLE 1 Enabling Solaris I/O Multipathing

To enable Solaris I/O multipathing for all multipath-capable controllers, run:

```
# stmsboot -e
```

To enable Solaris I/O multipathing on multipath-capable `mpt(7D)` controller ports, enter:

```
# stmsboot -D mpt -e
```

To enable Solaris I/O multipathing on multipath-capable `mpt_sas(7D)` controller ports, enter:

```
# stmsboot -D mpt_sas -e
```

To enable Solaris I/O Multipathing on multipath-capable fibre channel controller ports, enter:

```
# stmsboot -D fp -e
```

To enable Solaris I/O Multipathing on multipath-capable iSCSI controller ports, enter:

```
# stmsboot -D iscsi -e
```

EXAMPLE 2 Disabling Solaris I/O Multipathing

To disable Solaris I/O multipathing on all multipath-capable controllers, enter:

```
# stmsboot -d
```

To disable Solaris I/O multipathing on multipath-capable mpt(7D) controller ports, enter:

```
# stmsboot -D mpt -d
```

To disable Solaris I/O multipathing on multipath-capable mpt_sas(7D) controller ports, enter:

```
# stmsboot -D mpt_sas -d
```

To disable Solaris I/O multipathing on multipath-capable iSCSI controller ports, enter:

```
# stmsboot -D iscsi -d
```

To disable Solaris I/O multipathing on multipath-capable fibre channel controller ports, enter:

```
# stmsboot -D fp -d
```

EXAMPLE 3 Enabling Solaris I/O Multipathing on Selected Ports

To enable Solaris I/O multipathing on specific fibre channel controller ports and disable the feature on others, manually edit the `/kernel/drv/fp.conf` file. (See fp(7d).) The following command will update vfstab(4) and dumpadm(1M) configurations to reflect the changed device names:

```
# stmsboot -u
```

A similar procedure involving the `/kernel/drv/mpt.conf` file should be followed for devices attached by means of the mpt(7D) driver. For devices attached by means of the iscsi(7D) driver, follow a similar procedure that uses the `/kernel/drv/iscsi.conf` file.

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs, system/library
Interface Stability	Obsolete

See Also

dumpadm(1M), fsck(1M), mpathadm(1M), ufsdump(1M), zfs(1M), zpool(1M), ufsdump(4), vfstab(4), emlxs(7D), fcp(7D), fp(7d), iscsi(7D), mpt(7D), mpt_sas(7D), qlc(7D), scsi_vhci(7D)

Solaris SAN Configuration and Multipathing Guide (see `http://docs.sun.com`)

Consult a particular storage product's system administrator's guide and release notes for further information specific to that product.

Notes

Solaris I/O multipathing is not supported on all devices. After enabling Solaris I/O multipathing, only supported devices are placed under Solaris I/O multipathing control. Non-supported devices remain unchanged.

For Solaris releases prior to the current release, the `-e` and `-d` options replace `mpxio-disable` property entries with a global `mpxio-disable` entry in `fp.conf`.

Enabling Solaris I/O Multipathing on a Sun StorEdge Disk Array

The following applies to Sun StoreEdge T3, 3910, 3960, 6120, and 6320 storage subsystems.

To place your Sun StorEdge disk subsystem under Solaris I/O multipathing control, in addition to enabling Solaris I/O multipathing, the `mp_support` of the subsystem must be set to `mpxio` mode. The preferred sequence is to change the subsystem's `mp_support` to `mpxio` mode, then run `stmsboot -e`. If Solaris I/O multipathing is already enabled but the subsystem's `mp_support` is not in `mpxio` mode, then change the `mp_support` to `mpxio` mode and run `stmsboot -u`.

Refer to the *Sun StorEdge Administrator's Guide* for your subsystem for more details.

Using `ufsdump`

The `ufsdump(1M)` command records details of filesystem dumps in `/etc/dumpdates` (see `ufsdump(4)`). Among other items, the entries contain device names. An effect of the "active" `stmsboot` options (`-e`, `-d`, and `-u`) is to change the device name of a storage device.

Because `stmsboot` does not modify `dumpdates`, entries will refer to obsolete device names, that is, device names that were in effect before Solaris I/O multipathing configuration changes were performed. In this situation `ufsdump` will behave as if no previous dump of the filesystem had been performed. A level 0 dump will be performed.

Procedure to Use `stmsboot` in Conjunction with Sun Cluster

If possible, invoke `stmsboot -e` before installing Sun Cluster software. After executing `stmsboot`, install Sun Cluster software normally.

If Sun Cluster software is installed before executing `stmsboot`, follow this procedure:

On each machine in the cluster where Solaris I/O multipathing is required, execute:

```
# stmsboot -e
```

...and allow the system to reboot.

When the system comes up, enter the following two commands:

1. # **/usr/cluster/bin/scdidadm -C**
2. # **/usr/cluster/bin/scdidadm -r**

   The preceding commands update `did` mappings with new device names while preserving `did` instance numbers for disks that are connected to multiple cluster nodes. `did` instance numbers of the local disks might not be preserved. For this reason, the `did` disk names for local disks might change.
3. Update `/etc/vfstab` to reflect any new `did` disk names for your local disks.

4. Reboot the system.

To disable the Solaris multipathing feature, use `stmsboot -d` (instead of `stmsboot -e`), then follow the procedure above.

To view mappings between the old and new device names, run `stmsboot -L`. To view `did` device name mappings, run `/usr/cluster/bin/scdidadm -L`.

With active-passive storage arrays, it is possible that while your host is rebooting the array controller could failover the path that a particular target is using. In this scenario, `fsck(1M)` will fail to open the physical path listed in `/etc/vfstab`. The `svc:/system/filesystem/local:default` SMF service will transition to a maintenance state as a result. To rectify this, consult the documentation for your storage array to failback the path. The `mpathadm(1M)` can assist with determining the active and passive path(s).

Limitations On x86 platforms, the current Solaris release does not support disabling Solaris I/O multipathing of boot devices attached by means of fibre channel. Solaris I/O multipathing is always enabled for supported fibre channel-attached boot devices. Disabling Solaris I/O multipathing in this situation must be performed on a per-port basis. See `fp(7d)`.

Executing `devfsadm -C` removes obsolete device entries that `stmsboot` relies on. This will prevent correct operation of the `-d` option for boot devices (regardless of platform type) and the `-L` option.

Name strace – print STREAMS trace messages

Synopsis `strace [`*mid sid level*`]...`

Description `strace` without arguments writes all STREAMS event trace messages from all drivers and modules to its standard output. These messages are obtained from the STREAMS log driver (see `log(7D)`). If arguments are provided, they must be in triplets of the form *mid, sid, level*, where *mid* is a STREAMS module ID number, *sid* is a sub-ID number, and *level* is a tracing priority level. Each triplet indicates that tracing messages are to be received from the given module/driver, sub-ID (usually indicating minor device), and priority level equal to, or less than the given level. The token `all` may be used for any member to indicate no restriction for that attribute.

The format of each trace message output is:

<*seq*> <`time`> <*ticks*> <*level*> <*flags*> <*mid*> <*sid*> <*text*>

<*seq*>	trace sequence number
<`time`>	time of message in *hh:mm:ss*
<*ticks*>	time of message in machine ticks since boot
<*level*>	tracing priority level
<*flags*>	E : message is also in the error log F : indicates a fatal error N : mail was sent to the system administrator (hardcoded as root)
<*mid*>	module ID number of source
<*sid*>	sub-ID number of source
<*text*>	formatted text of the trace message

Once initiated, `strace` will continue to execute until terminated by the user.

Examples **EXAMPLE 1** A sample output of the `strace` command:

The following example outputs all trace messages from the module or driver whose module ID is `41`:

```
strace  41 all all
```

The following example outputs those trace messages from driver or module ID `41` with sub-IDs `0`, `1`, or `2`:

```
strace  41 0 1  41 1 1  41 2 0
```

Messages from sub-IDs `0` and `1` must have a tracing level less than or equal to `1`. Those from sub-ID `2` must have a tracing level of `0`.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also attributes(5), log(7D)

STREAMS Programming Guide

Notes

- There is no restriction to the number of `strace` processes opening the STREAMS log driver at a time.
- The log-driver records the list of the triplets specified in the command invocation, and compares each potential trace message against this list to decide if it should be formatted and sent up to the `strace` process. Hence, long lists of triplets will have a greater impact on overall STREAMS performance. Running `strace` will have the most impact on the timing of the modules and drivers generating the trace messages that are sent to the `strace` process. If trace messages are generated faster than the `strace` process can handle them, some of the messages will be lost. This last case can be determined by examining the sequence numbers on the trace messages output.

Name strclean – STREAMS error logger cleanup program

Synopsis

```
strclean [-a age] [-d logdir]
```

Description `strclean` is used to clean up the STREAMS error logger directory on a regular basis (for example, by using `cron`. By default, all files with names matching `error.*` in `/var/adm/streams` that have not been modified in the last three days are removed.

Options The following options are supported:

`-a` *age* The maximum age in days for a log file can be changed using the `-a` option.

`-d` *logdir* A directory other than `/var/adm/streams` can be specified using the `-d` option.

Examples **EXAMPLE 1** A sample of using the `strclean` command.

This example has the same result as running `strclean` with no arguments:

```
example%  strclean -d /var/adm/streams -a 3
```

Files `/var/adm/streams/error.*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `cron(1M)`, `strerr(1M)`, `attributes(5)`

STREAMS Programming Guide

Notes `strclean` is typically run from `cron` on a daily or weekly basis.

Name strerr – STREAMS error logger daemon

Synopsis

```
strerr
```

Description `strerr` receives error log messages from the STREAMS-based log driver (see `log(7D)`) and appends them to a log file. The resultant error log files reside in the directory `/var/adm/streams`, and are named `error.`*mm*`-dd`, where *mm* is the month and `dd` is the day of the messages contained in each log file.

The format of an error log message is:

<*seq*> <`time`> <*ticks*> <*flags*> <*mid*> <*sid*> <*text*>

<*seq*> error sequence number

<`time`> time of message in hh:mm:ss

<*ticks*> time of message in machine ticks since boot priority level

<*flags*> `T` : the message was also sent to a tracing process `F` : indicates a fatal error `N` : send mail to the system administrator (hardcoded as root)

<*mid*> module ID number of source

<*sid*> sub-ID number of source

<*text*> formatted text of the error message

Messages that appear in the error log are intended to report exceptional conditions that require the attention of the system administrator. Those messages which indicate the total failure of a STREAMS-based driver or module should have the `F` flag set. Those messages requiring the immediate attention of the administrator will have the `N` flag set, which causes the error logger to send the message to the system administrator using `mail`. The priority level usually has no meaning in the error log but will have meaning if the message is also sent to a tracer process.

Once initiated, `strerr` continues to execute until terminated by the user. It is commonly executed asynchronously.

Files `/var/adm/streams/error.`*mm*`-dd` error log file.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `attributes(5)`, `log(7D)`

STREAMS Programming Guide

Notes There is no restriction to the number of `strerr` processes opening the STREAMS–based log driver at a time.

If a module or driver is generating a large number of error messages, running the error logger will cause a degradation in STREAMS performance. If a large burst of messages are generated in a short time, the log driver may not be able to deliver some of the messages. This situation is indicated by gaps in the sequence numbering of the messages in the log files.

Name sttydefs – maintain line settings and hunt sequences for TTY ports

Synopsis

```
/usr/sbin/sttydefs -a ttylabel [-b] [-f final-flags]
     [-i initial-flags] [-n nextlabel]

/usr/sbin/sttydefs -l [ttylabel]

/usr/sbin/sttydefs -r ttylabel
```

Description `sttydefs` is an administrative command that maintains the line settings and hunt sequences for the system's TTY ports by making entries in, and deleting entries from the `/etc/ttydefs` file.

`sttydefs` with a `-a` or `-r` option may be invoked only by the super-user. `sttydefs` with `-l` may be invoked by any user on the system.

Options The following options are supported:

`-a` *ttylabel* — Add a record to the `ttydefs` file, using *ttylabel* as its label. The following describes the effect of the `-b`, `-n`, `-i`, or `-f` options when used in conjunction with the `-a` option:

`-b` — Enable autobaud. Autobaud allows the system to set the line speed of a given TTY port to the line speed of the device connected to the port without the user's intervention.

`-f` *final-flags* — Specify the value to be used in the *final-flags* field in `/etc/ttydefs`. *final-flags* must be in a format recognized by the `stty` command. *final-flags* are the `termio(7I)` settings used by `ttymon` after receiving a successful connection request and immediately before invoking the service on the port. If this option is not specified, `sttydefs` will set *final-flags* equal to the `termio(7I)` flags `9600` and `sane`.

`-i` *initial-flags* — Specify the value to be used in the *initial-flags* field in `/etc/ttydefs`. *initial-flags* must be in a format recognized by the `stty` command. These flags are used by `ttymon` when searching for the correct baud rate. They are set prior to writing the prompt. If this option is not specified, `sttydefs` will set *initial-flags* equal to the `termio(7I)` flag `9600`.

`-n` *nextlabel* — Specify the value to be used in the *nextlabel* field in `/etc/ttydefs`. If this option is not specified, `sttydefs` will set *nextlabel* equal to *ttylabel*.

`-l`[*ttylabel*] — If a *ttylabel* is specified, `sttydefs` displays the record from `/etc/ttydefs` whose TTY label matches the specified *ttylabel*. If no *ttylabel* is specified, `sttydefs` displays the entire contents of `/etc/ttydefs`. `sttydefs` verifies that each entry it displays is correct and that the entry's *nextlabel* field references an existing

`-r` *ttylabel* — Remove any record in the `ttydefs` file that has *ttylabel* as its label.

Output If successful, `sttydefs` will exit with a status of `0`. `sttydefs -l` will generate the requested information and send it to standard output.

Examples **EXAMPLE 1** A sample of `sttydefs` command.

The following command lists all the entries in the `ttydefs` file and prints an error message for each invalid entry that is detected.

```
example# sttydefs -l
```

The following shows a command that requests information for a single label and its output:

```
example# sttydefs -l 9600
----------------------------------------------------------------
9600:9600 hupcl erase ^h:9600 sane ixany tab3 hupcl erase ^h::4800
----------------------------------------------------------------
ttylabel:    9600
initial flags:    9600 hupcl erase ^h
final flags:    9600 sane ixany tab3 hupcl erase ^h
autobaud:    no
nextlabel:      4800
```

The following sequence of commands will add the labels `1200`, `2400`, `4800`, and `9600` and put them in a circular list:

```
sttydefs -a 1200 -n 2400 -i 1200 -f "1200 sane"
sttydefs -a 2400 -n 4800 -i 2400 -f "2400 sane"
sttydefs -a 4800 -n 9600 -i 4800 -f "4800 sane"
sttydefs -a 9600 -n 1200 -i 9600 -f "9600 sane"
```

Files `/etc/ttydefs`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `attributes(5)`, `termio(7I)`

Name su – become superuser or another user

Synopsis

```
su [-] [username [arg...]]
```

Description The su command allows one to become another user without logging off or to assume a role. The default user *name* is root (superuser).

To use su, the appropriate password must be supplied (unless the invoker is already root). If the password is correct, su creates a new shell process that has the real and effective user ID, group IDs, and supplementary group list set to those of the specified *username*. Additionally, the new shell's project ID is set to the default project ID of the specified user. See getprojent(3PROJECT), setproject(3PROJECT). The new shell will be the shell specified in the shell field of *username*'s password file entry (see passwd(4)). If no shell is specified, /usr/bin/sh is used (see sh(1)). If superuser privilege is requested and the shell for the superuser cannot be invoked using exec(2), /sbin/sh is used as a fallback. To return to normal user ID privileges, type an EOF character (CTRL-D) to exit the new shell.

Any additional arguments given on the command line are passed to the new shell. When using programs such as sh, an *arg* of the form -c *string* executes *string* using the shell and an *arg* of -r gives the user a restricted shell.

To create a login environment, the command "su –" does the following:

- In addition to what is already propagated, the LC* and LANG environment variables from the specified user's environment are also propagated.
- Propagate TZ from the user's environment. If TZ is not found in the user's environment, su uses the TZ value from the TIMEZONE parameter found in /etc/default/login.
- Set MAIL to /var/mail/*new_user*.

If the first argument to su is a dash (-), the environment will be changed to what would be expected if the user actually logged in as the specified user. Otherwise, the environment is passed along, with the exception of $PATH, which is controlled by PATH and SUPATH in /etc/default/su.

All attempts to become another user using su are logged in the log file /var/adm/sulog (see sulog(4)).

Security su uses pam(3PAM) with the service name su for authentication, account management, and credential establishment.

Examples **EXAMPLE 1** Becoming User bin While Retaining Your Previously Exported Environment

To become user bin while retaining your previously exported environment, execute:

```
example% su bin
```

EXAMPLE 2 Becoming User bin and Changing to bin's Login Environment

To become user bin but change the environment to what would be expected if bin had originally logged in, execute:

```
example% su - bin
```

EXAMPLE 3 Executing command with user bin's Environment and Permissions

To execute command with the temporary environment and permissions of user bin, type:

```
example% su - bin -c "command args"
```

Environment Variables

Variables with LD_ prefix are removed for security reasons. Thus, su bin will not retain previously exported variables with LD_ prefix while becoming user bin.

If any of the LC_* variables (LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY) (see environ(5)) are not set in the environment, the operational behavior of su for each corresponding locale category is determined by the value of the LANG environment variable. If LC_ALL is set, its contents are used to override both the LANG and the other LC_* variables. If none of the above variables are set in the environment, the "C" (U.S. style) locale determines how su behaves.

LC_CTYPE — Determines how su handles characters. When LC_CTYPE is set to a valid value, su can display and handle text and filenames containing valid characters for that locale. su can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. su can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

LC_MESSAGES — Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

Files

$HOME/.profile — user's login commands for sh and ksh

/etc/passwd — system's password file

/etc/profile — system-wide sh and ksh login commands

/var/adm/sulog — log file

/etc/default/su — the default parameters in this file are:

- SULOG — If defined, all attempts to su to another user are logged in the indicated file.
- CONSOLE — If defined, all attempts to su to root are logged on the console.

PATH	Default path. (/usr/bin:)
SUPATH	Default path for a user invoking su to root. (/usr/sbin:/usr/bin)
SYSLOG	Determines whether the syslog(3C) LOG_AUTH facility should be used to log all su attempts. LOG_NOTICE messages are generated for su's to root, LOG_INFO messages are generated for su's to other users, and LOG_CRIT messages are generated for failed su attempts.

/etc/default/login — the default parameters in this file are:

TIMEZONE	Sets the TZ environment variable of the shell.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also csh(1), env(1), ksh(1), login(1), roles(1), sh(1), syslogd(1M), exec(2), getprojent(3PROJECT), setproject(3PROJECT), pam(3PAM), pam_authenticate(3PAM), pam_acct_mgmt(3PAM), pam_setcred(3PAM), pam.conf(4), passwd(4), profile(4), sulog(4), syslog(3C), attributes(5), environ(5)

Name sulogin – access single-user mode

Synopsis

```
sulogin
```

Description The `sulogin` utility is automatically invoked by `init` when the system is first started. It prompts the user to type the root password to enter system maintenance mode (single-user mode) or to type EOF (typically CTRL-D) for normal startup (multi-user mode). The user should never directly invoke `sulogin`.

The `sulogin` utility can prompt the user to enter the root password on a variable number of serial console devices, in addition to the traditional console device. See `consadm(1m)` and `msglog(7D)` for a description of how to configure a serial device to display the single-user login prompt.

Files

`/etc/default/sulogin` Default value can be set for the following flag:

PASSREQ — Determines if login requires a password. Default is `PASSREQ=YES`.

`/etc/default/login` Default value can be set for the following flag:

`SLEEPTIME` — If present, sets the number of seconds to wait before login failure is printed to the screen and another login attempt is allowed. Default is `4` seconds. Minimum is `0` seconds. Maximum is `5` seconds.

Both `su(1M)` and `login(1)` are affected by the value of `SLEEPTIME`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

See Also `login(1)`, `consadm(1m)`, `init(1M)`, `su(1M)`, `attributes(5)`, `msglog(7D)`

Name suninstall – install the Solaris operating system

Synopsis

```
suninstall
```

Description `install-solaris(1M)` is now the preferred command for starting and restarting the Solaris Installation program. It should be used instead of `suninstall`. `suninstall` is symbolically linked to `install-solaris`.

`suninstall` is a forms-based and graphical subsystem for installing the operating system.

`suninstall` exists only on the Solaris installation media (CD or DVD) and should only be invoked from there. Refer to the *Solaris 10 Installation Guide: Basic Installations* for more details.

suninstall allows installation of the operating system onto any standalone system. suninstall loads the software available on the installation media. Refer to the *Solaris 10 Installation Guide: Basic Installations* for disk space requirements.

Usage Refer to the *Solaris 10 Installation Guide: Basic Installations* for more information on the various menus and selections.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcdrom (Solaris installation media)

See Also `pkginfo(1)`, `install(1M)`, `install-solaris(1M)`, `pkgadd(1M)`, `attributes(5)`

Solaris 10 Installation Guide: Basic Installations

Notes It is advisable to exit `suninstall` by means of the exit options in the `suninstall` menus.

Name SUNWgfb_config – fbconfig module for configuring Sun XVR-1000 Graphics Accelerator

Synopsis

```
fbconfig [-dev device-filename]
     [-active a | b | both | auto]
     [-res video-mode [now | try] [noconfirm]]
     [-file machine | system]
     [-doublewide enable | disable | reverse]
     [-doublehigh enable | disable | reverse]
     [-multisample enable | disable | auto [static | dynamic]]
     [-samples samples-per-pixel]
     [-g gamma-correction-value] [-master a | b | input]
     [-clearpixel0 | 255]

fbconfig [-dev device-filename] -stream a | b
     [-port hd15 | dvid | svideo | auto]
     [-res video-mode [now | try] [noconfirm]]
     [-file machine | system]
     [-offset xoff-value [yoff-value]]
     [-slave [enable | disable] [internal | external]]

fbconfig [-dev device-filename] -defaults

fbconfig [-dev device-filename]
     [-propt | -prconf | -res \?]

fbconfig [-help | -list]
```

Description `SUNWgfb_config` is the Sun XVR-1000 Graphics Accelerator device dependent layer for `fbconfig(1M)`. It configures the Sun XVR-1000 Graphics Accelerator and some of the X11 window system defaults. The device can drive two monitors, each with a unique video stream (called stream a and stream b).

The first form of `SUNWgfb_config` shown in the synopsis above sets card options, which are options common to both stream a and stream b, and apply to the entire card.

The second form is used to set stream options, which are options specific to either stream a or stream b. The second form usage requires the `-stream` option to define which stream is affected.

Both the first form and the second form store the specified options in the `OWconfig` file. These options will be used to initialize the device the next time the window system is run on that device. Updating options in the `OWconfig` file provides persistence of these options across window system sessions and system reboots. For `-res` with `now` or `try`, `-slave`, `-master`, `-port`, and `-g` , the device will be immediately programmed.

The third form, which invokes the `-defaults` option, sets all card options and all stream options to their default values and saves these defaults in the `OWconfig` file.

The fourth form, which invokes the `-prconf`, `-propt`, and `-res \?` options, queries the device for status that is card-specific.

The fifth form, which invokes the `-help`, and `-list` options, provides instruction on using `SUNWgfb_config` and a list of available devices. Additionally for the fifth form, all other options are ignored.

You can specify options for only one device at a time. Specifying options for multiple devices requires multiple invocations of `SUNWgfb_config`.

Only Sun XVR-1000 Graphics Accelerator-specific options can be specified through `SUNWgfb_config`. The normal window system options for specifying default depth, default visual class, and so forth are still specified as device modifiers on the command line when the X server is started.

You can also specify the `OWconfig` file that is to be updated. By default, the machine-specific file in the `/etc/openwin` directory tree is updated. You can use the `-file` option to specify an alternate file. For example, the system-global `OWconfig` file in the `/usr/openwin` directory tree can be updated instead.

General Options

`-dev` *device-filename*

Specifies the device's special file. The default is `/dev/fb`.

`-file machine|system`

Specifies which `OWconfig` file to update. If *machine* is specified, the machine-specific `OWconfig` file in the `/etc/openwin` directory tree is updated. If *system* is specified, the global `OWconfig` file in the `/usr/openwin` directory tree is updated. If the specified file does not exist, it is created. This option has no effect unless other options are specified. The default is *machine*.

`-res` *video-mode*`[now | try] [noconfirm]`

Specifies the video mode used to drive the monitor connected to the specified device. If `-res` is invoked with `now` or `try`, you must specify a `-stream` option or a device, such as `/dev/fbs/gfb0a`. If `-active` is set to `both` or `auto`, then both stream *video-mode* values will be modified.

The *video-mode* argument specifies resolution and timing information for the display (for example, `SUNW_STD_1280x1024x76`). The naming convention for the *video-mode* specifier is:

origin_type_widthxheightxrate

The elements of the specifier are described as follows:

origin This can be one of:

- `SUNW`, Sun derived resolution
- `VESA`, Video Electronics Standards Association-derived resolution
- `other`, other source

type This can be one of:

- `STD`, normal resolution, usable by most display devices
- `DIG`, resolution tuned only for LCD flat panels
- `INT`, interlaced
- `STEREO`, stereo

`width` screen width in pixels

`height` screen height in pixels

`rate` vertical frequency of the screen refresh

Note that some video-modes supported by the device, might not be supported by the monitor. The list of video-modes supported by the device and the monitor can be obtained by running `SUNWgfb_config` with the `-res \?` option (the fourth form shown in the command synopsis above).

The `-res` option also accepts additional, optional arguments, listed below, immediately following the video mode specification. Either `now` or `try` (`try` subsumes `now`) and `noconfirm` can be present.

`now` If present, not only is the video mode updated in the `OWconfig` file, but the device is immediately programmed to display this video mode. This is useful for changing the video mode before starting the window system.

Note – It is recommended that you not use this suboption with `SUNWgfb_config` while the configured device is being used (for example, while running the window system). Unpredictable results can occur. To run `SUNWgfb_config` with the `now` suboption, first bring the window system down. If the `now` suboption is used within a window system session, the video mode is changed immediately, but the width and height of the affected screen do not change until the window system is exited and reentered. In addition, the system might not recognize changes in stereo mode.

`noconfirm` Using the `-res` option, the user can put the system into an unusable state, with no video output. To reduce the chance of this, the default behavior of `SUNWgfb_config` is to display a warning message and to ask the user whether to continue. The `noconfirm` bypasses this confirmation. This option is useful when `SUNWgfb_config` is being run from a shell script.

`try` If present, the specified video mode will be programmed on a trial basis. The user is asked to confirm the video mode by typing **y** within 10 seconds. Alternatively, the user can

terminate the trial before 10 seconds elapse by typing any character other than **y** or carriage return. Such input is considered a no and the previous video mode is restored. With a negative response, `SUNWgfb_config` does not change the video mode in the `OWconfig` file; other options specified still take effect. If a carriage return is typed, the user is asked (**y** or **n**) whether to keep the new video mode.

This sub-option should not be used with `SUNWgfb_config` while the configured device is being used (for example, while running the window system) as unpredictable results may occur. To run `SUNWgfb_config` with the `try` sub-option, the window system should be brought down first.

`-defaults` — Resets all option values to their default values. Writes these values to the `OWconfig` file.

`-propt` — Displays the current values of all options in the `OWconfig` file specified by the `-file` option for the device specified by the `-dev` option. Displays the values of options as they will be in the `OWconfig` file after the call to `SUNWgfb_config` completes. The following is an example display:

```
--- OpenWindows Configuration for /dev/fbs/gfb0 ---
OWconfig: machine
Active Streams: both
Samples Per Pixel: 2
Multisample Allocation Model: static
```

```
Multisample Mode: auto
Doublewide: disable
Gamma Correction Value: 2.22

--- OpenWindows Configuration for Stream a ---
Video Mode: SUNW_STD_1280x1024x76

--- OpenWindows Configuration for Stream b ---
Video Mode: VESA_STD_640x480x60
```

`-prconf` Displays the XVR-1000 hardware configuration. The following is an example display:

```
--- Hardware Configuration for /dev/fb (SUNWgfb0) ---
Type:   Sun Graphics Accelerator
Part:   501-5865
Memory:
        MAJC:           32MB
        Texture:        256MB total
        3DRAM64:        5.0M pixels

Versions:  FCode 1.14  MCode 0.19  MAJC 2.1  FBC3 3.0  XChip 2.0

Video Streams:
   Stream a
         Current resolution Setting: SUNW_STD_1280x1024x76
         Monitor/EDID data (13W3)
                 Monitor Manufacturer: SUN
                 Monitor Name:  GDM-5410
                 EDID:  Version 1, Revision 2
   Stream b
         Current resolution Setting: VESA_STD_640x480x60
         Port: svideo
```

`-help` Displays a list of the `SUNWgfb_config` command line options, along with a brief explanation of each.

`-res \?` Displays list of defined *video-mode* names.

Card Options

`-active a | b | both | auto`

Specifies which streams are enabled. `both` select both streams. The default is `auto`, which means whichever stream is chosen by the console.

`-multisample` `enable` | `disable` | `auto` [`static` | `dynamic`]

The suboptions for `-multisample` are described as follows:

`disable` No multisample is possible.

`enable` Multisample is possible but is selected on a per-application basis.

`auto` All Sun OpenGL applications are rendered using multisampling.

`static` Multisample allocation occurs at X startup/config load time. The config *samples-per-pixel* or `max` parameter specifies the depth that is pre-allocated.

`dynamic` OpenGL tasks allocate buffers themselves.

`-samples` *samples-per-pixel*
Specifies the number of samples/pixel to pre-allocate in static mode. Provides a hint to OpenGL in dynamic mode. The allowable choices for *samples-per-pixel* are 2, 3, 4, 5, 6, 8, 10, 16, and `max`. The default is `max`, which means to use the maximum number of samples that can be supported with the amount of memory available.

`-doublewide` `enable` | `disable` | `reverse`
This option makes it easy for you to combine both streams into one side-by-side virtual display. If you specify `enable`, stream a is to the left of stream b. If `reverse` is specified, stream b is to the left of stream b. Both will be the same resolution defined with the `-res` option. If you specify `disable`, only one stream will be enabled. `-doublewide` precludes `-doublehigh`.

`-doublehigh` `enable` | `disable` | `reverse`
This option makes it easy for you to combine both streams into one virtual display, with one monitor on a shelf above the other. If you specify `enable`, stream a is above stream b. If `reverse` is specified, stream b is above stream a. Both will be the same resolution defined with the `-res` option. If you specify `disable`, only one stream will be enabled. `-doublehigh` precludes `-doublewide`.

`-g` *gamma-correction value*
This option changes the gamma correction value. By default the gamma correction value is 2.22. Any value less than zero is illegal. This option can be used while the window system is running. Changing the gamma correction value will affect all the windows being displayed using gamma-corrected visuals. The gamma correction value is also saved in the `OWconfig` file for the next time the window system starts.

`-master` `a` | `b` | `input`
This option controls the setting frame pins on the stereo/sync connector on the device. It also controls which stream drives stereo glasses, which attach to the same connector.

If `a` (the default) or `b` is selected, the card is setup to be a sync master, and the frame sync signal from the corresponding stream will be sent out this connector.

If you select `input`, the card is setup to take its frame sync from another card through this connector. This sync can then be used to sync either or both streams by setting the stream-specific `-slave` option(s) to `external`.

If stereo glasses are used, the `a` or `b` options select which stream is used for the sync signal to the glasses.

`-clearpixel 0 | 255`
Selects the overlay transparent color. This is the pixel value (color index) used by the transparent overlay visual to display the underlay (RGB) pixel contents. The default is 255 (all bits 1), but some applications require 0. All other color indices display a colormap color.

Stream Options

`-stream a | b`
Specifies for which stream options will be set. It is a required option for each of the other options in this section. It is optional for `-res`. Only one `-stream` option can be specified.

`-port hd15 | dvid | svideo | auto`
Directs stream b to the appropriate output connector: `hd-15`, `dvid`, or `svideo`. If `auto`, then the output connector for stream b is selected by the console. Stream a is always output through the 13W3 connector.

`-offset` *xoff-value* [*yoff-value*]
Offsets the display of the stream (specified by `-stream`) relative to the adjoining edge of the other stream when doublewide or doublehigh is enabled. This can be used to cause an overlap.

xoff-value Number of pixels offset in horizontal direction for the right-hand stream when doublewide is enabled. Positive direction is to the right (create a gap); negative is to the left (overlap the streams). Default is 0, which means the two edges touch.

yoff-value Number of pixels offset in vertical direction for the bottom stream when doublehigh is enabled. Positive direction is down (create a gap); negative is up (overlap the streams). Default is 0, which means the two edges touch.

`-slave [ enable | disable ] [ internal | external ]`
This option allows you to specify the sync source for the specified stream. `internal` indicates that the sync source is the other stream of this device. `external` indicates the sync is taken from a source outside the device. If you use `external`, you also need to use the card option `-master input`.

Defaults

For a given invocation of `SUNWgfb_config`, if an option does not appear on the command line, the corresponding `OWconfig` option is not updated. It retains its previous value.

When the window system is run, if an option has never been specified through `SUNWgfb_config`, a default value is used. The option defaults are as follows:

Option	Default
–dev	/dev/fb
–file	machine
–res	none

–samples	max
–multisample	enable/dynamic
–clearpixel	255
–master	a
–slave	disable/external
–doublewide	not set
–g	2.22
-active	auto
-port	auto
-offset	0/0

The default for the `-res` option of `none` means that, when the window system is run, the screen resolution will be the video mode that is currently programmed in the device. This provides compatibility for users who are used to specifying the device resolution through the PROM.

Examples

EXAMPLE 1 Switching Resolution of a Monitor

The following example switches to the resolution of 1280 by 1024 at 76 Hz:

```
example% fbconfig -stream a -res -SUNW_STD_1280x1024x76
```

Files

`/usr/lib/fbconfig/SUNWgfb_config` device special file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWgfbcf

See Also

`fbconfig(1M)`, `attributes(5)`

See the `dtlogin(1)` man page in the CDE man page collection. Also useful is the `Xsun(1)` man page in the OpenWindows man page collection.

Name SUNWifb_config – configure the Sun Expert3D Graphics Accelerator

Synopsis

```
/usr/lib/fbconfig/SUNWifb_config [-dev device-filename]
     [-res video-mode [now | try] [noconfirm | nocheck]]
     [-file machine | system] [-deflinear true | false]
     [-defoverlay true | false]
     [-linearorder first | last]
     [-overlayorder first | last]
     [-expvis enable | disable] [-slave enable | disable]
     [-accum enable | disable] [-g gamma-correction-value]
     [-gfile gamma-correction-file] [-propt] [-prconf]
     [-defaults] [-slave] [] [-samples 1 | 2 | 4 | 8 | 16]
     [-multisample enable | disable | auto]

/usr/lib/fbconfig/SUNWifb_config [-propt] [prconf]

/usr/lib/fbconfig/SUNWifb_config [-help] [-res \?]
```

Description `SUNWifb_config` configures the Sun Expert3D Graphics Accelerator, Sun Expert3D-Lite, and Sun XVR-500 Graphics Accelerators, and some of the X11 window system defaults for the graphics accelerator.

The first form of `SUNWifb_config` shown in the synopsis above stores the specified options in the `OWconfig` file. These options will be used to initialize the Sun Expert3D device the next time the window system is run on that device. Updating options in the `OWconfig` file provides persistence of these options across window system sessions and system reboots.

The second and third forms of `SUNWifb_config`, which invoke only the `-prconf`, `-propt`, `-help`, and `-res \?` options, do not update the `OWconfig` file. Additionally, for the third form of the command, all other options are ignored.

Options may be specified for only one Sun Expert3D device at a time. Specifying options for multiple Sun Expert3D devices requires multiple invocations of `SUNWifb_config`.

Only options specific to the Sun Expert3D device can be specified through `SUNWifb_config`. The normal window system options for specifying default depth, default visual class and so forth are still specified as device modifiers on the `openwin` command line (see the `Xsun`(1) manual page in the *OpenWindows Desktop Reference Manual*).

Options The following options are supported:

`-dev` *device-filename*
: Specifies the Sun Expert3D special file. The default is `/dev/fbs/ifb0`.

`-file`*machine*| *system*
: Specifies which `OWconfig` file to update. If *machine* is specified, the machine-specific `OWconfig` file in the `/etc/openwin` directory tree is updated. If *system* is specified, the global `OWconfig` file in the `/usr/openwin` directory tree is updated. If the

specified file does not exist, it is created. This option has no effect unless other options are specified. The default is *machine*.

`-res` *video-mode*

Specifies the video mode used to drive the monitor connected to the specified Sun Expert3D device.

The format of these built-in video modes is:

*width*x*height*x*rate*

where *width* is the screen width in pixels, *height* is the screen height in pixels, and *rate* is the vertical frequency of the screen refresh. The `s` suffix of `960x680x112s` and `960x680x108s` means that these are stereo video modes. The `i` suffix of `640x480x60i` and `768x575x50i` designates interlaced video timing. If absent, non-interlaced timing will be used. As a convenience, `-res` also accepts formats with `@` (at sign) in front of the refresh rate instead of `x`. For example: `1280x1024@76`. Note that some video-modes supported by the Sun Expert3D device might not be supported by the monitor. The list of video-modes supported by the Sun Expert3D device and the monitor can be obtained by running `SUNWifb_config` with the `-res \?` option (shown in the command synopsis above). The following is a list of all possible video-modes supported on the Sun Expert3D device:

	1024x768x60 1024x768x70 1024x768x75 1024x768x75 1024x768x77 1024x800x84 1152x900x66 1152x900x76 1280x800x76 1280x1024x60 1280x1024x67 1280x1024x76 1280x1024x85 1280x1024x112s (Stereo) 960x680x112s (Stereo) 960x680x108s (Stereo) 640x480x60 640x480x60i (Interlaced) 768x575x50i (Interlaced) 1440x900x76 1600x1000x66 1600x1000x76 1600x1280x76 1792x1344x75 1920x1080x72 1920x1200x70 1920x1200x75
Symbolic names	For convenience, some of the above video modes have symbolic names defined for them. Instead of the form *width* x *height* x *rate*, one of these names may be supplied as the argument to `-res`. The meaning of the symbolic name none is that when the window system is run the screen resolution will be the

video mode that is currently programmed in the device.

Name	Corresponding Video Mode
svga	1024x768x60
1152	1152x900x76
1280	1280x1024x76
stereo	960x680x112s
ntsc	640x480x60i
pal	768x575x50i
none	(see text above)

The `res` option also accepts additional, optional arguments immediately following the video mode specification. Any or all of the following might be present.

`now`

If present, not only will the video mode be updated in the `OWconfig` file, but the Sun Expert3D device will be immediately programmed to display this video mode. (This is useful for changing the video mode before starting the window system).

Note that it is inadvisable to use this suboption with `SUNWifb_config` while the configured device is being used (for example, while running the window system); unpredictable results might occur. To run `SUNWifb_config` with the `now` suboption, first bring the window system down. If the `now` suboption is used within a window system

session, the video mode will be changed immediately, but the width and height of the affected screen will not change until the window system is exited and reentered again. In addition, the system may not recognize changes in stereo mode. Consequently, this usage is strongly discouraged.

`noconfirm` Using the `-res` option, the user could potentially put the system into an unusable state, a state where there is no video output. This can happen if there is ambiguity in the monitor sense codes for the particular code read. To reduce the chance of this, the default behavior of `SUNWifb_config` is to print a warning message to this effect and to prompt the user to find out if it is okay to continue. The `noconfirm` option instructs `SUNWifb_config` to bypass this confirmation and to program the requested video mode anyway. This option is useful when `SUNWifb_config` is being run from a shell script.

`nocheck` If present, the normal error checking based on the monitor sense code (described above) will be suspended. The video mode specified by the user

will be accepted regardless of whether it is appropriate for the currently attached monitor. (This option is useful if a different monitor is to be connected to the Sun Expert3D device). Use of this option implies `noconfirm` as well.

`try`

If present, the specified video mode will be programmed on a trial basis. The user will be asked to confirm the video mode by typing **y** within 10 seconds. Or the user may terminate the trial before 10 seconds are up by typing any character. Any character other than **y** or carriage return is considered a no and the previous video mode will be restored and `SUNWifb_config` will not change the video mode in the `OWconfig` file (other options specified will still take effect). If a carriage return is typed, the user is prompted for a yes or no answer on whether to keep the new video mode.

This sub-option should not be used with `SUNWifb_config` while the configured device is being used (for example, while running the window system) as unpredictable results may occur. To run `SUNWifb_config` with the

`try` sub-option, the window system should be brought down first.

`-deflinear true | false`

The Sun Expert3D device possesses two types of visuals: linear and nonlinear. Linear visuals are gamma corrected and nonlinear visuals are not. There are two visuals that have both linear and nonlinear versions: 24-bit TrueColor and 8-bit StaticGray. If `true`, the default visual is set to the linear visual that satisfies other specified default visual selection options (specifically, the `Xsun`(1) `-defdepth` and `-defclass` options described in the *OpenWindows Desktop Reference Manual*). If `false`, or if there is no linear visual that satisfies the other default visual selection options, the non-linear visual specified by these other options will be chosen to be the default. This option cannot be used when the `-defoverlay` option is present, because the Sun Expert3D does not possess a linear overlay visual.

`-defoverlay true | false`

The Sun Expert3D device provides an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the Sun Expert3D visuals. This is called the overlay visual. Windows created in this visual will not damage windows created in other visuals. The converse, however, is not true. Windows created in other visuals will damage overlay windows. If the value of this option is `true`, the overlay visual will be made the default visual. If `false`, the nonoverlay visual that satisfies the other default visual selection options, such as `-defdepth` and `-defclass`, will be chosen as the default visual. See the `Xsun`(1) manual page in the *OpenWindows Desktop Reference Manual*. Whenever `-defoverlay true` is used, the default depth and class chosen on the `openwin` command line must be 8-bit PseudoColor. If not, a warning message will be printed and the `-defoverlay` option will be treated as `false`. This option cannot be used when the `-deflinear` option is present, because the Sun Expert3D device does not possess a linear overlay visual.

`-linearorder first | last`

If `first`, linear visuals will come before their non-linear counterparts on the X11 screen visual list for the Sun Expert3D screen. If `last`, the nonlinear visuals will come before the linear ones.

`-overlayorder first | last`

If `-first`, the depth 8 PseudoColor Overlay visual will come before the non-overlay visual on the X11 screen visual list for the Sun Expert3D screen. If `last`, the non-overlay visual will come before the overlay one.

`-expvis enable | disable`

If enabled, OpenGL Visual Expansion will be activated. Multiple instances of selected visual groups (8-bit PseudoColor, 24-bit TrueColor, and so forth) can be found in the screen visual list.

`-slave enable | disable`

If enabled, the video for this frame buffer will be synced with the video of the display which is connected to it. For applications which support it buffers will also be swapped synchronously.

`-accum enable | disable`

If enabled, frame buffer memory is allocated for accelerated accumulation buffer for windows. If disabled, software accumulation buffering will be done for windows. Accelerated accumulation buffers for pBuffers are always available as memory allows.

`-g` *gamma-correction_value*

This option allows changing the gamma correction value. All linear visuals provide gamma correction. By default the gamma correction value is 2.22. Any value less than zero is illegal. The gamma correction value is applied to the linear visual, which then has an effective gamma value of 1.0, which is the value returned by `XSolarisGetVisualGamma()`. See `XSolarisGetVisualGamma`(3) for a description of that function.This option can be used while the window system is running. Changing the gamma correction value will affect all the windows being displayed using the linear visuals.

`-gfile` *gamma-correction_file*

This option loads gamma correction table from the specified file. This file should be formatted to provide the gamma correction values for R, G and B channels on each line. Each of these values

should be in hexadecimal format and separated from each other by at least one space. Also, this file should provide 1024 such triplets. An example of this file is as follows.

```
0x00 0x00 0x000
0x01 0x01 0x001
0x02 0x02 0x002
...
...
0x3ff 0x3ff 0x3ff
```

Using this option, the gamma correction table can be loaded while the window system is running. The new gamma correction will affect all the windows being displayed using the linear visuals. Note that, when gamma correction is being done using a user-specified table, the gamma correction value is undefined. By default, the window system assumes a gamma correction value of 2.22 and loads the gamma table it creates corresponding to this value.

`-defaults` Resets all option values to their default values.

`-propt` Prints the current values of all Sun Expert3D options in the `OWconfig` file specified by the `-file` option for the device specified by the `-dev` option. Prints the values of options as they will be in the `OWconfig` file after the call to `SUNWifb_config` completes. This is a typical display:

```
--- OpenWindows Configuration for /dev/fbs/ifb0 ---
OWconfig: machine
Video Mode: 1280x1024x76
Accum: Disabled (do not allocate an accumulation buffer)
Default Visual: Non-Linear Normal Visual
Visual Ordering: Linear Visuals are last
                 Overlay Visuals are last
OpenGL Visual Expansion: enabled
Gamma Correction Value: 2.22
Gamma Correction Table: Available
```

`-prconf` Prints the Sun Expert3D hardware configuration. This is a typical display:

```
--- Hardware Configuration for /dev/fbs/ifb0 ---
PROM Information: @(#)ifb.fth 1.25 99/10/12 SMI
```

```
EDID Data: Available - EDID version 1 revision 1
Monitor possible resolutions: 1024x768x60, 1024x768x70, 1024x768x75,
1152x900x66, 1152x900x76, 1280x1024x67, 1280x1024x76, 960x680x112s,
640x480x60
Current resolution setting: 1280x1024x76
```

`-help` — Prints a list of the `SUNWifb_config` command-line options, along with a brief explanation of each.

`-samples` `1` | `2` | `4` | `8` | `16` — Requested number of samples to compute per display pixel. The requested number of samples per pixel will be used if `-multisample` is not disabled and resouces exist for the request.

Query the number of samples used with `-propt` (see above) or the `xglinfo` utility. The `xglinfo` utility can return the number of multisamples after you specify the option `-multisample enable`.

The default is `16`.

`-multisample` `enable` | `disable` | `auto` — If set to `disable`, no multisample is possible. If set to `enable`, multisample is possible but is selected on a per-window basis using a library interface. If set to `auto`, all Sun OpenGL windows are rendered using multisampling.

Query the number of samples used with `-propt` (see above) or the `xglinfo` utility. The `xglinfo` utility can return the number of multisamples if `-multisample` is set to `enable`.

The default is `disable`.

The `xglinfo` utility is shipped with the Sun OpenGL package, `SUNWglrt`. The man page for `xglinfo` is part of another Sun OpenGL package, `SUNWgldoc`.

Defaults For a given invocation of `SUNWifb_config` command line if an option does not appear on the command line, the corresponding `OWconfig` option is not updated; it retains its previous value. When the window system is run, if an Sun Expert3D option has never been specified via `SUNWifb_config`, a default value is used. The option defaults are as follows:

Option	Default
-dev	/dev/fbs/ifb0
-file	machine
-res	none
-deflinear	false

```
-defoverlay     false
-linearorder    last
-overlayorder   last
-expvis         enable
-slave          disable
-accum          enable
-g              2.22
-samples        16
-multisample    disable
```

The default for the `-res` option of `none` means that, when the window system is run, the screen resolution will be the video mode that is currently programmed in the device. This design choice provides compatibility for users who are used to specifying the device resolution through the PROM. On some devices (for example, GX), this is the only way of specifying the video mode. This means that the PROM ultimately determines the default Sun Expert3D video mode.

Examples

EXAMPLE 1 Changing Monitor Resolution

The following example switches the monitor type to the resolution of 1280 x 1024 at 76 Hz:

```
example% /usr/lib/fbconfig/SUNWifb_config -res 1280x1024x76
```

Files

`/dev/fbs/ifb0` device special file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWifbcf

See Also

`attributes(5)`, `mmap(2)`, `ifb(7d)`, `fbio(7I)`

Name SUNWjfb_config – fbconfig module for configuring the Sun XVR-600 and XVR-1200 Graphics Devices

Synopsis

```
fbconfig [-dev device-filename]
     [-res video-mode [now | try] [noconfirm | nocheck]]
     [-file machine | system] [-deflinear true | false]
     [-defoverlay true | false]
     [-linearorder first | last]
     [-outputs swapped | direct]
     [-slave disable | multiview | stereo | bnc]
     [-accum enable | disable] [-g gamma-correction-value]
     [-gfile gamma-correction-file]
     [-fake8 enable | disable]
     [-doublewide enable | disable]
     [-doublehigh enable | disable]
     [-multisample available | disable | forceon]
     [-samples 1 | 2 | 4 | 8 | 16 | adaptive]
     [-defdepth 8 | 24] [-offset xval yval] [-propt]
     [-prconf] [-defaults]

fbconfig [-propt] [-prconf]

fbconfig [-help] [-res ?]
```

Description `SUNWjfb_config` configures the Sun XVR-600 and XVR-1200 Graphics Accelerator and some of the X11 window system defaults for the Sun XVR-600 and XVR-1200.

You can specify options for only one device at a time. If you want to specify options for multiple devices, you must invoke the `SUNWjfb_config` utility multiple times.

You can use the `SUNWjfb_config` utility to specify Sun XVR-600 or XVR-1200 specific options. Use the normal window system options to specify default depth, default visual class and so forth. Specify these as device modifiers on the command line. See `Xsun`(1)

You can also specify which `OWconfig` file to update. By default, `SUNWjfb_config` updates the machine-specific file in the `/etc/openwin`. You can specify an alternate file by using the `-file` option. For example, the system-global `OWconfig` file in the `/usr/openwin` directory tree can be updated instead.

Both of these standard `OWconfig` files can only be written by root. Consequently, the `SUNWjfb_config` program, which is owned by the root user, always runs with setuid root permission.

The first form of `SUNWjfb_config` shown in the `SYNOPSIS` section stores the specified options in the OWconfig file. These options initialize the Sun XVR-600 and XVR-1200 device the next time the window system is run on that device. Updating options in the OWconfig file provides persistence of these options across window system sessions and system reboots.

The second and third forms which invoke only the `-prconf`, `-propt`, `-help`, and `-res ?` options, do not update the OWconfig file. Additionally, for the third form all other options are ignored.

Options The following options are supported:

`-accum enable | disable`
Enable or disable frame buffer memory. If enabled, frame buffer memory is allocated for the accelerated accumulation buffer for windows. If disabled, software accumulation buffering for windows is used. Accelerated accumulation buffers for pBuffers are always available as memory allows.

`-dev` *device-filename*
Specify the SunSun XVR-600 or XVR-1200 special file. The default is `/dev/fbs/jfb0`.

`-defaults`
Reset all option values except `-dev` to their default values.

This option sets the resolution override to `none` which implies the last video mode setting is preserved. If no monitor is attached, the video mode is set to `1152x900x66`.

`-deflinear true | false`
Both the Sun XVR-600 and XVR-1200 possess two types of visuals: linear and nonlinear. Linear visuals are gamma corrected and nonlinear visuals are not. There are two visuals that have both linear and nonlinear versions: 24-bit TrueColor and 8-bit StaticGray.

If the value of this option is `true`, the default visual is set to the linear visual that satisfies other specified default visual selection options (specifically, the `Xsun` `defdepth` and `defclass` options; see `Xsun`(1)). If `false`, or if there is no linear visual that satisfies the other default visual selection options, the non-linear visual specified by these other options is chosen as the default visual. This option cannot be used when the `-defoverlay` option is present, because the Sun XVR-600 and XVR-1200 do not possess a linear overlay visual.

`-defoverlay true | false`
Both the Sun XVR-600 and XVR-1200 provide an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the visuals. This is called the overlay visual. Windows created in this visual will not damage windows created in other visuals. The converse, however, is not true. Windows created in other visuals will damage overlay windows.

If the value of this option is `true`, the overlay visual is the default visual. If `false`, the nonoverlay visual that satisfies the other default visual selection options, such as `defdepth` and `defclass`, is chosen as the default visual. See `Xsun`(1). Whenever `-defoverlay true` is used, the default depth and class chosen on the `openwin` command line must be 8-bit PseudoColor. If not, a warning message is printed and the `-defoverlay` option is treated as `false`. This option cannot be used when the `-deflinear` option is present, because the Sun XVR-600 and XVR-1200 do not possess a linear overlay visual.

`-defdepth 8 | 24`
Set the depth (bits per pixel) on the device. Possible values for the `-defdepth` option are `8` or `24`. You must log out of the current window system session and log back in again for the change to take effect. Any depth setting in the Xserver command line takes precedence over what is set using `fbconfig`. The default is determined by the X Server `smf(5)` default depth property. You can query this property with the following command:

```
% /usr/sbin/svccfg -s svc:/application/x11/x11-server listprop \
'options/default_depth'
```

`-doublehigh enable | disable`
Configure the two outputs of the Sun XVR-1200 into one vertical virtual display. The default is `disable`.

This option is not applicable to the Sun XVR-600 device.

`-doublewide enable | disable`
Configure the two outputs of the Sun XVR-1200 into one horizontal virtual display. The default is `disable`.

This option is not applicable to the Sun XVR-600 device.

`-fake8 enable | disable`
Enable or disable simple 8 bit X windows to be rendered without a hardware colormap to reduce colormap flashing. You might notice performance reductions. The default is `disable`.

`-file machine|system`
Specifies which `OWconfig` file to update. If *machine* is specified, the machine-specific `OWconfig` file in the `/etc/openwin` directory tree is updated. If *system* is specified, the global `OWconfig` file in the `/usr/openwin` directory tree is updated. If the specified file does not exist, it is created. This option has no effect unless other options are specified. The default is *machine*.

`-g` *gamma-correction value*
Change the gamma correction value. All linear visuals provide gamma correction. By default, the gamma correction value is `2.22`. Any value less than zero is illegal. This option can be used while the window system is running. Changing the gamma correction value affects all of the windows displayed by linear visuals.

`-gfile` *gamma-correction file*
Load the gamma correction table from the file specified by *gamma-correction file*. You should format this file to provide the gamma correction values for `R`, `G`, and `B` channels on each line. Specify each value in hexadecimal format and separate it from another value by at least 1 space. The *gamma-correction file* file should provide 1024 such triplets.

You can load the gamma correction table with this option can while the window system is running. The new gamma correction affects all the window being displayed using the linear visuals. When gamma correction is done using a user specified table, the gamma correction

value is undefined. By default, the window system assumes a gamma correction value of `2.22` and loads the gamma table it creates corresponding to this value.

The following is an example of a *gamma-correction file* file:

```
0x000 0x000 0x000
0x001 0x001 0x001
0x002 0x002 0x002
...
...
0x3ff 0x3ff 0x3ff
```

`-help`
Print a list of the `SUNWjfb_config` command line options, along with a brief explanation of each.

`-multisample available | disable |forceon`
If set to disable, no multisample is possible. If set to `available`, multisample is possible but is selected on a per-window basis using a library interface. If set to `forceon`, all Sun OpenGL windows are rendered using multisampling. Query the number of samples used with `-propt` or the `xglinfo`(1) utility. The `xglinfo` utility can return the number of multisamples if `-multisample` is set to enable. The default is `disable`.

`-offset` *xval yval*
Adjust the position of the specified stream by the value specified. This option is only implemented in `-doublewide` and `-doublehigh` modes. For `-doublewide`, use the *xval* to position the rightmost stream. Negative is left (overlaps with the left stream). For `-doublehigh`, use the *yval* to position the bottom stream. Negative is up (overlaps with top stream). The default is `[0, 0]`.

`-outputs swapped | direct`
If either `-doublewide` or `-doublehigh` are enabled, reverse the position of the two outputs relative to each other. The default is `direct`.

`-propt`
Print the current values of all Sun XVR-600 and XVR-1200 options in the OWconfig file specified by the `-file` option for the device specified by the `-dev` option. Print the values of options as they will be in the OWconfig file after the call to `SUNWjfb_config` completes.

This is a typical display:

```
# fbconfig -dev jfb2 -propt

--- OpenWindows Configuration for /dev/fbs/jfb2 ---

OWconfig: machine
Video Mode: 1280x1024x60
Accum: Enabled (allocate an accumulation buffer if possible)

Multisample Information:
```

```
    Multisample Mode: Disabled (multisample visuals
       will not be available)
    Samples Per Pixel: N/A (multisampling disabled)

   Screen Information:
    DoubleWide: Disabled
    DoubleHigh: Disabled
    Output Configuration: Direct
    Offset/Overlap: [0, 0]

   Visual Information:
    Default Visual: Non-Linear Normal Visual
    Visual Ordering: Linear Visuals are last
    Gamma Correction Value: 2.22
    Gamma Correction Table: Available
    Fake8 rendering: disabled
    Default Visual Depth (defdepth): 8
```

`-prconf`

Print the Sun XVR-1200 hardware configuration. This is a typical display:

```
# fbconfig -dev jfb2 -prconf

--- Hardware Configuration for /dev/fbs/jfb2 ---

Type: XVR-1200
Sun Serial Number: 3753101000022
Hardware Revision: -01 rev01
Manufacture Date: Thu Aug 8  12:54:16 2002
PROM Information: @(#)jfb.fth 1.8 02/10/18 SMI

Monitor/Resolution Information:
   EDID Data: Available - EDID version 1 revision 3
   Monitor type: Sun P/N 365-1415 S/N 0216ME0353
   Current resolution setting: 1280x1024x60 (custom)
   Monitor possible resolutions: 1024x768x60, 1024x768x70,
   1024x768x75, 1152x900x66, 1280x1024x60, 1280x1024x75,
   1280x1024x76, 640x480x60, 800x600x75

Framelock Configuration:
   Slave Mode: Disabled

Memory Information:
   Total Video Memory: 134217728
   Video Memory Used: 15728640
   Total Texture Memory: 268435456
   Texture Memory Used: 0
   Total Display List Memory: 33554432
```

`-res` *video-mode* `[ now |try [ noconfirm | nocheck ]]`

Specify the video mode of the monitor connected to the specified Sun XVR-600 or XVR-1200 device.

The `-res` option requires you to specify the video-mode. You can specify *video-mode* in the format of *width*x*height*x*rate* or as a symbolic name.

*width*x*height*x*rate*

Specify *video-mode* in the format of *width*x*height*x*rate*, where *width* is the screen width in pixels, *height* is the screen height in pixels, and *rate* is the vertical frequency of the screen refresh.

You can use the `s` suffix to specify stero video modes. The `s` suffix of `960x680x112s` and `960x680x108s` means that these are stereo video modes.

The `-res` option also accepts formats with `@` (at sign) in front of the refresh rate instead of `x`, (`280x1024@76`). Some video-modes, supported by Sun XVR-600 and XVR-1200, might not be supported by the monitor.

The list of video-modes supported by the Sun XVR-600 and XVR-1200 device and the monitor can be obtained by running `SUNWjfb_config` with the `-res ?` option. See `SYNOPSIS`.

Symbolic Names

Some video modes have symbolic names defined for them. Instead of using the *width*x*height*x*rate* format, you can specify one of the symbolic names as the argument to the `-res` option. The meaning of the symbolic name `none` is that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

The following symbolic names and their corresponding video modes are supported:

```
svga          1024x768x60
1152          1152x900x76
1280          1280x1024x76
stereo        960x680x112s
ntsc          640x480x60i
pal           768x575x50i
none          Programmed video mode
```

The `-res` option accepts additional, optional arguments immediately following the video mode specification. The following additional, optional arguments are supported:

`now`

If present, updates the video mode be updated in the `OWconfig file`. Programs the Sun XVR-600 or XVR-1200 device to display the video mode. This is useful for changing the video mode before starting the window system.

Do not use this suboption with `SUNWjfb_config` while the configured device is being used, for example, while running the window system. Unpredictable results can occur.

If you want to run SUNWjfb_config with the now suboption, first bring the window system down. If you use the now suboption within a window system session, the video mode is changed immediately. The width and height of the affected screen will not change until the window system is exited and re-entered again. Additionally, the system might not recognize changes in stereo mode. This usage is discouraged.

noconfirm

Using the -res option, the user could potentially put the system into an unusable state, a state where there is no video output. This can happen if there is ambiguity in the monitor sense codes for the particular code read. To reduce the chance of this, the default behavior of SUNWjfb_config is to print a warning message to this effect and to prompt the user to find out if it is okay to continue. The -noconfirm option instructs SUNWjfb_config to bypass this confirmation and to program the requested video mode anyway. This option is useful when SUNWjfb_config is run from a shell script.

nocheck

If present, the normal error checking based on the monitor sense code is suspended. The video mode specified is accepted regardless of whether it is appropriate for the currently attached monitor. This option is useful if a different monitor is to be connected to the Sun XVR-600 or XVR-1200 device. Use of this option implies noconfirm.

try

If present, programs the specified video mode on a trial basis. You are asked to confirm the video mode by entering a y within 10 seconds. You can terminate the trial before 10 seconds by entering any character but y or RETURN.

Terminating the trial by entering a character other than y or RETURN restores the previous video mode. SUNWjfb_config does not change the video mode in the OWconfig file. Other specified options still take effect. If a carriage return is typed, the user is prompted for a yes or no answer on whether to keep the new video mode. This option implies the now suboption.

If you want to run SUNWjfb_config with the now suboption, first bring the window system down. If you use the now suboption within a window system session, the video mode is changed immediately. The width and height of the affected screen will not change until the window system is exited and re-entered again. Additionally, the system might not recognize changes in stereo mode. This usage is strongly discouraged.

-samples 1 | | 4 | 8 | 16 | adaptive

Request the number of samples to compute per display pixel. The requested number of samples per pixel will be used if -multisample is not disabled and resources exist for the request. If set to adaptive, variable multisampling is possible, whith the minimum samples per pixel set to 2. Query the number of samples used with -propt or the xglinfo(1) utility. The xglinfo utility can return the number of multisamples after you specify the option -multisample enable. The default is 16.

`-slave disable | multiview | stereo | bnc`
Enable or disable a specific port. Specify `multiview` for the DB9 connector, `stereo` for the mini-DIN connector, or `bnc` for the bnc connector. Specify disable to disable a specific port. If set, the option is an exclusive slave instance in the Framelock Configuration Setup. The default is `disable`.

`multiview` and `bnc` are only applicable to the Sun XVR-1200 device.

Defaults For a given invocation of `SUNWjfb_config` command line if an option does not appear on the command line, the corresponding OWconfig option is not updated; it retains its previous value.

When the window system is run, if an option has never been specified by way of `SUNWjfb_config`, a default value is used. The options and their corresponding defaults are as follows:

Option	Default
`-dev`	`/dev/fbs/jfb0`
`-file`	`machine`
`-res`	`none`
`-deflinear`	`false`
`-defoverlay`	`false`
`-linearorder`	`last`
`-slave`	`disable`
`-accum`	`enable`
`-g`	`2.22`
`-samples`	`16`
`-multisample`	`disable`
`-doublewide`	`disable`
`-doublehigh`	`disable`
`-outputs`	`direct`
`-offset`	`[0,0]`

The default for the `-res` option of none means that when the window system is run the screen resolution will be the video mode that is currently programmed in the device.

This provides compatibility for users who are used to specifying the device resolution through the PROM. This means that the PROM ultimately determines the default video mode.

Examples

EXAMPLE 1 Switching the Resolution of the Monitor Type

The following example switches the monitor type to 1280 × 1024 at 76 Hz resolution:

```
example% fbconfig –dev jfb0 –res 1280x1024x76
```

Files

`/dev/fbs/jfb`*n*
Device special file for the XVR-600 or XVR-1200 high performance single screen

`/dev/fbs/jfb`*n*`a`
Device special file for the XVR-1200 first video out

`/dev/fbs/jfb`*n*`b`
Device special file for the XVR-1200 second video out

`/usr/lib/jfb.ucode`
`jfb` microcode file

`/usr/sbin/jfbdaemon`
`jfb` microcode loader file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWjfbcf

See Also

`fbconfig(1M)`, `svccfg(1M)`, `mmap(2)`, `attributes(5)`, `smf(5)`, `fbio(7I)`, `jfb(7D)`

The `Xsun(1)` man page, which is not part of the SunOS man page collection.

Name SUNWkfb_config – fbconfig module for configuring the Sun XVR-2500 Graphics Device

Synopsis

```
/usr/lib/fbconfig/SUNWkfb_config [-dev device-filename]
     [-res video-mode] [-file machine | system]
     [-deflinear true | false] [-defoverlay true | false]
     [-deftransparent true | false]
     [-slave disable | multiview]
     [-g gamma-correction-value]
     [-gfile gamma-correction-file]
     [-doublewide enable | disable]
     [-doublehigh enable | disable]
     [-multisample available | disable | forceon]
     [-samples 1 | 2 | 4 | 8 | 16 ] [-offset xval yval]
     [-propt] [-prconf] [-defaults]

/usr/lib/fbconfig/SUNWkfb_config [-propt] [-prconf]

/usr/lib/fbconfig/SUNWkfb_config [-help] [-res ?]
```

Description `SUNWkfb_config` configures the Sun XVR-2500 Graphics Accelerator and some of the X11 window system defaults for the Sun XVR-2500.

The first form of `SUNWkfb_config` shown in the `SYNOPSIS` section stores the specified options in the `OWconfig` file. These options initialize the Sun XVR-2500 device the next time the window system is run on that device. Updating options in the `OWconfig` file provides persistence of these options across window system sessions and system reboots.

The second and third forms which invoke only the `-prconf`, `-propt`, `-help`, and `-res ?` options, do not update the `OWconfig` file. For the third form, all other options are ignored.

You can specify options for only one device at a time. If you want to specify options for multiple devices, invoke the `SUNWkfb_config` utility multiple times.

Only the Sun XVR-2500 options can be specified through the `SUNWkfb_config` utility. Use the normal window system options to specify default depth (see `svccfg(1M)`), default visual class and so forth. Specify these as device modifiers on the command line. See `Xsun`(1).

You can also specify which `OWconfig` file to update. By default, `SUNWkfb_config` updates the machine-specific file in the `/etc/openwin` directory tree. Use the `-file` option to specify an alternate file. For example, the system-global `OWconfig` file in the `/usr/openwin` directory tree can be updated instead.

Both of these standard `OWconfig` files can only be written by root. Consequently, the `SUNWkfb_config` program, which is owned by the root user, always runs with setuid root permission.

Options The following options are supported:

`-defaults`

Resets all option values except `-dev` to their default values. This option sets the resolution override to `none` which implies that the last video mode setting is preserved. If no monitor is attached, the video mode is set to `1152x900x66`.

`-deflinear true | false`

This option selects the default X visual. The Sun XVR-2500 device supports two types of visuals: linear and nonlinear. Linear visuals are gamma corrected and nonlinear visuals are not.

If the value of this option is `true`, the default visual is set to default depth 24 and the default class is TrueColor with gamma correction enabled. If `false`, a non-linear visual that satisfies the other default visual selection options, such as the default depth and default class, is chosen as the default visual. See `Xsun`(1).

The `-deflinear`, `-defoverlay`, and `-deftransparent` options all select the default X visual. Only one option may be enabled at the same time. If more than one of these options is entered, the last selection takes precedence and the other options are disabled.

`-defoverlay true | false`

This option selects the default X visual. The Sun XVR-2500 device provides an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the visuals. This is called the overlay visual. Windows created in this visual will not damage windows created in other visuals. The converse, however, is not true. Windows created in other visuals will damage overlay windows.

If the value of this option is `true`, the overlay visual is the default visual. The default depth is 8 bit and the default class is PseudoColor. If `false`, the nonoverlay

visual that satisfies the other default visual selection options, such as the default depth and the default class, is chosen as the default visual. See `Xsun`(1).

The `-deflinear`, `-defoverlay`, and `-deftransparent` options all select the default X visual. Only one option may be enabled at the same time. If more than one of these options is entered, the last selection takes precedence and the other options are disabled.

`-deftransparent` `true` | `false`

This option selects the default X visual. The Sun XVR-2500 device provides an 8-bit PseudoColor visual whose pixels are disjoint from the rest of the visuals. This is called the overlay visual. Windows created in this visual will not damage windows created in other visuals.

If the value of this option is `true`, the overlay visual used as the default is a transparent overlay visual. A visual with transparency supports a colormap with 255 colors and one transparent pixel. The default depth is 8 bit and the default class is PseudoColor. If `false`, the nonoverlay visual that satisfies the other default visual selection options, such as the default depth and the default class, is chosen as the default visual. See `Xsun`(1).

The `-deflinear`, `-defoverlay`, and `-deftransparent` options all select the default X visual. Only one option may be enabled at the same time. If more than one of these options is entered, the last selection takes precedence and the other options are disabled.

`-dev` *device-filename*

Specifies the SunXVR-2500 special file. The default is `/dev/fbs/kfb0`.

`-doublehigh enable | disable`

Configures the two outputs of the Sun XVR-2500 device into one vertical virtual display. The default is `disable`.

`-doublewide enable | disable`

Configures the two outputs of the Sun XVR-2500 device into one horizontal virtual display. The default is `disable`.

`-file machine|system`

Specifies which `OWconfig` file to update. If *machine* is specified, the machine-specific `OWconfig` file in the `/etc/openwin` directory tree is updated. If *system* is specified, the global `OWconfig` file in the `/usr/openwin` directory tree is updated. If the specified file does not exist, it is created. This option has no effect unless other options are specified. The default is *machine*.

`-g` *gamma-correction value*

Changes the gamma correction value. All linear visuals provide gamma correction. By default, the gamma correction value is `2.22`. Any value less than zero is illegal. This option can be used while the window system is running. Changing the gamma correction value affects all of the windows displayed by linear visuals.

`-gfile` *gamma-correction file*

Loads the gamma correction table from the file specified by *gamma-correction file*. Format this file to provide the gamma correction values for `R`, `G`, and `B` channels on each line. Specify each value in hexadecimal format, separated from another value by at least one space. The *gamma-correction file* file should provide 1024 such triplets.

You can load the gamma correction table with this option while the window system is running. The new gamma correction affects all the windows being displayed using the linear visuals. When gamma correction is done using a user specified table, the gamma correction value is undefined. By default, the window system

assumes a gamma correction value of `2.22` and loads the gamma table it creates corresponding to this value.

The following is an example of a *gamma-correction file* file:

```
0x000 0x000 0x000
0x001 0x001 0x001
0x002 0x002 0x002
...
...
0x3ff 0x3ff 0x3ff
```

`-help` — Prints a list of the `SUNWkfb_config` command line options, along with a brief explanation of each.

`-multisample available | disable | forceon` — If set to `disable`, no multisample is possible. If set to `available`, multisample is possible but is selected on a per-window basis using a library interface. If set to `forceon`, all Sun OpenGL windows are rendered using multisampling. Query the number of samples used with `-propt` or the `xglinfo`(1) utility. The `xglinfo` utility can return the number of multisamples if `-multisample` is set to `available`. The default is `disable`.

`-offset` *xval yval* — Adjusts the position of the specified stream by the value specified. This option is only implemented in `-doublewide` and `-doublehigh` modes. For `-doublewide`, use the *xval* to position the rightmost stream. Negative is left (overlaps with the left stream). For `-doublehigh`, use the *yval* to position the bottom stream. Negative is up (overlaps with top stream). The default is `[0, 0]`.

`-prconf` — Prints the Sun XVR-2500 hardware configuration. This is a typical display:

```
--- Hardware Configuration for /dev/fbs/kfb2 ---

Type: XVR-2500
PROM Information: @(#) xvr2500.fth 14.106 05/08/15 SMI
```

```
Monitor/Resolution Information:
   Monitor Manufacturer: SUN
   Product code: 1352
   Serial #: 7225675
   Manufacture date: 1997, week 51
   Monitor dimensions: 48x31 cm
   Monitor preferred resolution: 1920x1200x70
   Separate sync supported: yes
   Composite sync supported: yes
   EDID: Version 1, Revision 1
   Supported resolutions: 1920x1200x70, SUNW_STD_1600x1000x76,
    SUNW_STD_1600x1000x66, SUNW_STD_1440x900x76, 1920x1080x70,
    1600x900x76, 1600x900x66, 1440x810x76, 1280x720x76,
    1920x1080x72, VESA_STD_720x400x70, VESA_STD_720x400x88,
    VESA_STD_640x480x60, VESA_STD_640x480x67,
    VESA_STD_800x600x56, VESA_STD_800x600x60,
    VESA_STD_800x600x72, VESA_STD_800x600x75,
    VESA_STD_1024x768x70, VESA_STD_1280x1024x75,
   Current resolution setting: SUNW_STD_1280x1024x76

Framelock Configuration:
 Slave Mode: Disabled
```

`-propt`

Prints the current values of all Sun XVR-2500 options in the `OWconfig` file specified by the `-file` option for the device specified by the `-dev` option. Prints the values of options as they will be in the `OWconfig` file after the call to `SUNWkfb_config` completes.

This is a typical display:

```
--- OpenWindows Configuration for /dev/fbs/kfb2 ---

OWconfig: machine
Video Mode: SUNW_STD_1280x1024x76

Multisample Information:
 Multisample Mode: Disabled (multisample visuals
    will not be available)
 Samples Per Pixel: N/A (multisampling disabled)

Screen Information:
 DoubleWide: Disabled
 DoubleHigh: Disabled
 Offset/Overlap: [0, 0]
```

```
Visual Information:
 Default Visual: Non-Linear Normal Visual
 Gamma Correction Value: using gamma value 2.22
```

`-res` *video-mode*

Specifies the video mode of the monitor connected to the specified Sun XVR-2500 device.

You must specify the video mode with the `-res` option. You can specify *video-mode* in the format of *width*x*height*x*rate* or as a symbolic name. The format of these built-in video modes is:

*width*x*height*x*rate*

width is the screen width in pixels, *height* is the screen height in pixels, and *rate* is the vertical frequency of the screen refresh.

You can use the `s` suffix to specify stereo video modes. The `s` suffix of `960x680x112s` and `960x680x108s` means that these are stereo video modes.

As a convenience, the `-res` option also accepts formats with `@` (at sign) in front of the refresh rate instead of `x`, as for example, `1280x1024@76`. Some video-modes supported by the Sun XVR-2500 device may not be supported by the monitor.

You can obtain the list of video-modes supported by the Sun XVR-2500 device and the monitor by running `SUNWkfb_config` with the `-res ?` option. See `SYNOPSIS`.

Symbolic Names

Some video modes have symbolic names defined for them. Instead of using the *width*x*height*x*rate* format, you can specify one of the symbolic names as the argument to the `-res` option. The meaning of the symbolic name `none` is that when the window system is run, the screen resolution is the video mode that is currently programmed in the device.

The following symbolic names and their corresponding video modes are supported:

```
Name          Cprresponding Video Mode
svga          1024x768x60
1152          1152x900x76
1280          1280x1024x76
stereo        960x680x112s
none          (see text above)
```

`-samples 1 | 2 | 4 | 8 | 16`	Requests the number of samples to compute per display pixel. The requested number of samples per pixel is used if `-multisample` is not disabled and resources exist for the request. Query the number of samples used with `-propt` or the `xglinfo`(1) utility. The `xglinfo` utility can return the number of multisamples after you specify the option `-multisample available`. The default is `4`.
`-slave disable | multiview`	If you set the `multiview` argument for the `-slave` option, the device synchronizes video with a master through the multiview ribbon cable. Both devices should be running the same resolution and the option should be issued when the window system is running. The default is `disable`.

Defaults For a given invocation of the `SUNWkfb_config` command line, if an option does not appear on the command line, the corresponding `OWconfig` option is not updated and retains its previous value.

When the window system is run, if a Sun XVR-2500 option has never been specified by way of `SUNWkfb_config`, a default value is used. The options and their corresponding defaults are as follows:

Option	Default
`-dev`	`/dev/fbs/kfb0`
`-file`	`machine`
`-res`	`none`
`-deflinear`	`false`
`-defoverlay`	`false`
`-deftransparent`	`false`
`-slave`	`disable`
`-g`	`2.22`
`-samples`	`4`
`-multisample`	`disable`

Option	Default
-doublewide	disable
-doublehigh	disable
-outputs	direct
-offset	[0,0]

The default for the -res option of none means that when the window system is run, the screen resolution is the video mode that is currently programmed in the device.

Examples

EXAMPLE 1 Switching the Resolution of the Monitor Type

The following example switches the monitor type to 1280 × 1024 at 76 Hz resolution:

```
example% fbconfig –dev kfb0 –res 1280x1024x76
```

Files

/dev/fbs/kfb*n*	Device special file for the XVR-2500 high performance single screen
/usr/sbin/kfbdaemon	The kfb memory manager

Attributes

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWkfbcf
Interface Stability	Evolving

See Also

fbconfig(1M), svccfg(1M), mmap(2), attributes(5), kfb(7D), fbio(7I)

See the Xsun(1) man page in the OpenWindows man page collection and the xglinfo(1) man page in the Sun Open GL man page collection.

Name SUNWnfb_config – fbconfig module for configuring XVR-300 Graphics Accelerator

Synopsis

```
fbconfig [-dev device-filename]
     [-res video-mode [now | try] [noconfirm | nocheck]]
     [-file machine | system] [-fake8 enable | disable]
     [-doublewide enable | disable]
     [-doublehigh enable | disable]
     [-clone enable | disable] [-outputs swapped | direct]
     [-depth 8 | 24] [-offset xval yval] [-defaults]

fbconfig [-propt] [-prconf]

fbconfig [-help] [-res ?]
```

Description `SUNWnfb_config` is the XVR-300 device dependent layer for `fbconfig`. It configures the XVR-300 Graphics Accelerator and some of the X11 window system defaults for XVR-300. The XVR-300 provides the capability to drive two monitors, each with a unique video stream (Stream #1 and Stream #2).

The first form of `SUNWnfb_config` shown in the `SYNOPSIS` section sets options for the XVR-300. This form stores the specified options in the `OWconfig` file. These options will be used to initialize the XVR-300 device the next time the window system is run on that device. Updating options in the `OWconfig` file provides persistence of these options across window system sessions and system reboots. For `-res now`, the XVR-300 device will be immediately programmed.

The second form, which invokes the `-prconf` and `-propt` options, queries the XVR-300 for status.

The third form, which invokes the `-help` and `-res ?` options, provides instruction on using `SUNWnfb_config`. Additionally, for the third form all other options are ignored.

Options may be specified for only one XVR-300 device at a time. Specifying options for multiple XVR-300 devices requires multiple invocations of `SUNWnfb_config`.

Only XVR-300-specific options can be specified through `SUNWnfb_config`. The normal window system options for specifying default depth, default visual class and so forth can still be specified as device modifiers on the command line when the X Server is started. See the `dtlogin`(1) reference to Xservers File.

The user can also specify the `OWconfig` file that is to be updated. By default, the machine-specific file in the `/etc/openwin` directory tree is updated. The `-file` option can be used to specify an alternate file to use. For example, the system-global `OWconfig` file in the `/usr/openwin` directory tree can be updated instead.

Options `-dev` *device-filename*

Specifies the XVR-300 special file. The default is `/dev/fb`.

`-file machine | system`
Specifies which `OWconfig` file to update. If `machine`, the machine-specific `OWconfig` file in the `/etc/openwin` directory tree is used. If `system`, the global `OWconfig` file in the `/usr/openwin` directory tree is used. If the file does not exist, it is created.

`-res` *video-mode* [`now` | `try`] [`noconfirm` | `nocheck`]
Specifies the video mode used to drive the monitor connected to the specified XVR-300 device. Video modes are built-in.

video-mode Specifies resolution and timing information for the display (for example, 1280x1024x76). The format of the *video-mode* specifier is: *width*x*height*x*rate*. *width* is the screen width in pixels, *height* is the screen height in pixels, and *rate* is the vertical frequency of the screen refresh. As a convenience, `-res` also accepts formats with `@` preceding the refresh rate instead of `x`. For example, `1280x1024@76`.

`Symbolic names`. For convenience, some video modes have symbolic names defined for them. Instead of the form *width*x*height*x*rate*, one of these names may be supplied as the argument to `-res`. The meaning of the symbolic name `none` is that when the window system is run, the screen resolution will be the video mode that is currently programmed in the device.

Some video-modes supported by XVR-300 may not be supported by the monitor. The list of video-modes supported by the XVR-300 device and the monitor can be obtained by running `SUNWnfb_config` with the `-res ?` option (the third form shown in the command `SYNOPSIS` section).

`-res` option arguments. The `-res` option also accepts additional, optional arguments immediately following the video mode specification. Any or all of these may be present.

`now` If present, not only will the video mode be updated in the `OWconfig` file, but the XVR-300 device will be immediately programmed to display this video mode. (This is useful for changing the video mode before starting the window system).

It is inadvisable to use this suboption with `SUNWnfb_config` while the configured device is being used (for example, while running the window system) because unpredictable results may occur. To run `SUNWnfb_config` with the `now` suboption, first bring the window system down. If the `now` suboption is used within a window system session, the video mode will be changed immediately, but the width and height of the affected screen won't change until the

window system is exited and reentered again. Consequently, this usage is discouraged.

`noconfirm` Using the `-res` option, the user could put the system into an unusable state, which has no video output. To reduce the chance of this, the default behavior of `SUNWnfb_config` is to print a warning message and to ask the user whether to continue. The `noconfirm` suboption bypasses this confirmation. This suboption is useful when `SUNWnfb_config` is being run from a shell script.

`nocheck` If present, the normal error checking based on the monitor sense code will be suspended. The video mode specified by the user will be accepted regardless of whether it is appropriate for the currently attached monitor. This suboption is useful if a different monitor is to be connected to the XVR-300 device. Use of this suboption implies `noconfirm` as well.

`try` If present, the specified video mode will be programmed on a trial basis. The user will be asked to confirm the video mode by typing 'y' within 10 seconds. Or the user may terminate the trial before 10 seconds are up by typing any character. Any character other than 'y' or carriage return is considered a no and the previous video mode will be restored and `SUNWnfb_config` will not change the video mode in the `OWconfig` file (other options specified will still take effect). If a carriage return is typed, the user is prompted for a yes or no answer on whether to keep the new video mode. This option implies the `now` suboption (see the warning note on the `now` suboption).

`-doublewide enable | disable`
This option allows you to configure the two outputs of the Sun XVR-300 into one horizontal virtual display. Both video outputs will be set to the same resolution. Default is `disable`.

`-doublehigh enable | disable`
This option allows you to configure the two outputs of the Sun XVR-300 into one vertical virtual display. Both video outputs will be set to the same resolution. Default is `disable`.

`-clone enable | disable`
If this option is set to `enable`, the two outputs will display identically. Default is `disable`.

`-outputs swapped | direct`
If either `doublewide` or `doublehigh` is enabled, this option allows you to reverse the positions of the two outputs relative to each other. Default is `direct`.

`-fake8 enable | disable`
If enabled, simple 8 bit X windows will be rendered without a hardware colormap to reduce colormap flashing. Some performance reductions may be observed. Default is `disable`.

`-depth 8 | 24`
Sets the default depth for the window system to start with. Possible values are `8` or `24`. Log out of the current window system session and log back in for the change to take effect. Any defdepth setting in the Xserver command line takes precedence over what is set using `fbconfig`. Default is `24`. This option is only supported on Solaris 9. For Solaris 10 and above, use `/usr/sbin/svccfg` to configure depth.

`-offset` *xval yval*
Adjusts the position of the specified stream by the value specified. Currently only implemented in `-doublewide` and `-doublehigh` modes. For `-doublewide`, the *xval* is used to position the rightmost stream. Negative is left (overlaps with the left stream). For `-doublehigh`, the *yval* is used to position the bottom stream. Negative is up (overlaps with top stream). Positive values are treated as 0. Default is [0, 0]

`-defaults`
Resets all option values to their default values.

`-propt`
Prints the current values of all XVR-300 options in the `OWconfig` file specified by the `-file` option for the device specified by the `-dev` option. Prints the values of options as they will be in the `OWconfig` file after the call to `SUNWnfb_config` completes. This is a typical display:

```
--- Graphics Configuration for /dev/fbs/nfb0 ---

OWconfig: machine
Video Mode: NONE

Screen Information:
        Doublewide: Disable
        Doublehigh: Disable
        Clone: Disable
        Offset/Overlap: [0, 0]
        Output Configuration: Direct
        Fake8 Rendering: Disable
```

`-prconf`
Prints the XVR-300 hardware configuration. This is a typical display:

```
--- Hardware Configuration for /dev/fbs/nfb0 ---

Type: XVR-300
ASIC: version 0x5b64  REV : version 0x3800080
PROM: version 1.6

Monitor/Resolution Information:
```

```
Monitor 1:
       Monitor Manufacturer: SUN
       Product code: 1383
       Serial #: 18499659
       Manufacture date: 1999, week 36
       Monitor dimensions: 39x29 cm
       Monitor preferred resolution: SUNW_STD_1280x1024x76
       Separate sync supported: yes
       Composite sync supported: yes
       EDID:  Version 1, Revision 1
       Monitor Supported resolutions from EDID: SUNW_STD_1280x1024x76,
        SUNW_STD_1152x900x76, 1600x1200x75, SUNW_STD_1280x1024x76,
        SUNW_STD_1152x900x76, VESA_STD_1280x1024x75,
        SUNW_STD_1280x1024x67, SUNW_STD_1152x900x66,
        VESA_STD_1024x768x75, SUNW_STD_1600x1200x75, 960x720x112,
        VESA_STD_720x400x70, VESA_STD_720x400x88, VESA_STD_640x480x60,
        VESA_STD_640x480x67, VESA_STD_640x480x72, VESA_STD_640x480x75,
        VESA_STD_800x600x56, VESA_STD_800x600x60, VESA_STD_800x600x72,
        VESA_STD_800x600x75, VESA_STD_832x624x75, VESA_STD_1024x768x60,
        VESA_STD_1024x768x70, VESA_STD_1024x768x75,
        VESA_STD_1280x1024x75, APPLE_1152x870x75
       Current resolution setting: 1280x1024x76
Monitor 2:
       Monitor Manufacturer: SUN
       Product code: 1352
       Serial #: 7225675
       Manufacture date: 1997, week 51
       Monitor dimensions: 48x31 cm
       Monitor preferred resolution: 1920x1200x70
       Separate sync supported: yes
       Composite sync supported: yes
       EDID:  Version 1, Revision 1
       Monitor Supported resolutions from EDID: 1920x1200x70,
      SUNW_STD_1600x1000x76, SUNW_STD_1600x1000x66,
        SUNW_STD_1440x900x76, 1920x1080x70,
        1600x900x76, 1600x900x66, 1440x810x76, 1280x720x76,
        1920x1080x72, VESA_STD_720x400x70, VESA_STD_720x400x88,
        VESA_STD_640x480x60, VESA_STD_640x480x67, VESA_STD_640x480x72,
        VESA_STD_640x480x75, VESA_STD_800x600x56, VESA_STD_800x600x60,
        VESA_STD_800x600x72, VESA_STD_800x600x75, VESA_STD_832x624x75,
        VESA_STD_1024x768x60, VESA_STD_1024x768x70,
        VESA_STD_1024x768x75, VESA_STD_1280x1024x75, APPLE_1152x870x75
       Current resolution setting: 1152x900x66
```

`-help`

Prints a list of the `SUNWnfb_config` command line options, along with a brief explanation of each.

`-res ?`
Prints list of defined *video-mode* names.

Defaults For a given invocation of a `SUNWnfb_config` command line if an option does not appear on the command line, the corresponding `OWconfig` option is not updated and retains its previous value.

When the window system is run, if an XVR-300 option has never been specified via `SUNWnfb_config`, a default value is used. The option defaults are as follows:

Option	Default
–dev	/dev/fb
–file	machine
–res	none
–fake8	disable
–doublewide	disable
–doublehigh	disable
–clone	disable
–outputs	direct
–offset	[0,0]

The default for the `-res` option of `none` means that when the window system is run, the screen resolution will be the video mode that is currently programmed in the XVR-300 PROM.

This provides compatibility for users who are used to specifying the device resolution through the XVR-300 PROM. On some devices (for example, GX) this is the only way of specifying the video mode. This means that the PROM ultimately determines the default XVR-300 video mode.

Examples **EXAMPLE 1** Switching the monitor type

The following example switches the monitor type to the resolution of 1280 × 1024 at 76 Hz:

```
example% fbconfig -res 1280x1024x76
```

Files

`/dev/fbs/nfb`*n*	Device special file for XVR-300 single screen
`/dev/fbs/nfb`*n*`a`	Device special file for the XVR-300 first video out
`/dev/fbs/nfb`*n*`b`	Device special file for the XVR-300 second video out
`/usr/lib/fbconfig/SUNWnfb_config`	Device dependent configuration module

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfbcf
Interface Stability	Evolving

See Also fbconfig(1M), attributes(5), nfb(7D)

See the dtlogin(1) man page in the CDE man page collection. Also useful is the XSun(1) man page in the OpenWindows man page collection.

Name SUNWpfb_config – fbconfig module for configuring the XVR-50 and the Sun XVR-100 Graphics Accelerator

Synopsis

```
For XVR-50:
/usr/lib/fbconfig/SUNWpfb_config [-dev device-filename]
     [-res video-mode [now | try] [noconfirm | nocheck]]
     [-file  machine | system]
     [-fake8  enable | disable]
     [-g  enable | disable | gamma-value]
     [-defaults]

For Sun XVR-100:
/usr/lib/fbconfig/SUNWpfb_config [-dev device-filename]
     [-res video-mode [now | try] [noconfirm | nocheck]]
     [-file  machine | system]
     [-fake8  enable | disable]
     [-g  enable | disable | gamma-value]
     [-doublewide  enable | disable]
     [-doublehigh  enable | disable]
     [-outputs  swapped | direct]
     [-depth  8 | 24]
     [-offset  xval yval]
     [-defaults]

For Sun XVR-100 and XVR-50:
/usr/lib/fbconfig/SUNWpfb_config [-pt] [-prconf]

/usr/lib/fbconfig/SUNWpfb_config [-help] [-res \?]
```

Description `SUNWpfb_config` is the XVR-50 and the Sun XVR-100 device dependent layer for `fbconfig(1M)`. It configures the XVR-50 and the Sun XVR-100 Graphics Accelerator and some of the X11 window system defaults for the XVR-50 and Sun XVR-100. The Sun XVR-100 provides the capability to drive two monitors, each with a unique video stream (Stream #1 and Stream #2). The XVR-50 drives one video stream.

The first two forms of `SUNWpfb_config` shown in the `SYNOPSIS` section sets options for the XVR-50 and the Sun XVR-100 devices. It stores the specified options in the `OWconfig` file. These options initialize the XVR-50 and the Sun XVR-100 devices the next time the window system is run on that device. Updating options in the `OWconfig` file provides persistence of these options across window system sessions and system reboots. The XVR-50 and the Sun XVR-100 devices are immediately programmed if you specify the `-res now`.

The third form, which invokes the `-prconf` and `-propt` options, queries the XVR-50 and the Sun XVR-100 device for status.

The fourth form, which invokes the `-help`, and `-res \?` options, provides instruction on using `SUNWpfb_config`. Additionally, for the fourth form all other options are ignored.

You can only specify options for only one XVR-50 or Sun XVR-100 device at a time. If you want to specify options for multiple XVR-50 or Sun XVR-100 devices, you must use multiple invocations of the `SUNWpfb_config` command.

You can only use `SUNWpfb_config` to specify XVR-50 or Sun XVR-100 specific options. You can use the normal window system options to specify the default depth, default visual class and so forth as device modifiers on the command line when the X Server is started. See `dtlogin`(1) for information regarding the Xservers File.

You can also specify the `OWconfig` file to update. The machine-specific file in the `/etc/openwin` directory tree is updated by default. You can used the `-file` option to specify an alternate file. For example, you can update the system-global `OWconfig` file in the `/usr/openwin` directory tree instead.

Options The following options are supported:

`-defaults`
Reset all option values to their default values.

`-depth` 8 | 24
Set the depth (bits per pixel) for the window system. Possible values for the `-depth` option are `8` or `24`. You must log out of the current window system session and log back in again for the change to take effect. Any depth setting in the Xserver command line takes precedence over what is set using `fbconfig`. The default is `8`. This option is only available with Solaris 8 or Solaris 9.

`-dev` *device-filename*
Specify the XVR-50 or the Sun XVR-100 special file. The default is `/dev/fb`.

`-doublehigh enable | disable`
Configure the two outputs of the Sun XVR-100 into one vertical virtual display. You must log out of the current window system session and log back in again for the change to take effect. The default is `disable`.

`-doublewide enable | disable`
Configure the two outputs of the Sun XVR-100 into one horizontal virtual display. You must log out of the current window system session and log back in again for the change to take effect. The default is `disable`.

`-fake8` enable | disable
Enable or disable simple 8 bit X windows to be rendered without a hardware colormap to reduce colormap flashing. You might notice performance reductions. You must log out of the current window system session and log back in again for the change to take effect. The default is `disable`.

`-file machine|system`
Specifies which `OWconfig` file to update. If machine is specified, the machine-specific `OWconfig` file in the `/etc/openwin` directory tree is updated. If system is specified, the

global `OWconfig` file in the `/usr/openwin` directory tree is updated. If the specified file does not exist, it is created. This option has no effect unless other options are specified. The default is machine.

`-g` enable | disable | *gamma-value*
This option allows you to change the gamma correction value. All linear visuals provide gamma correction. By default , the gamma correction value is 2.22. Any value less than zero is illegal.

This option can be used while the window system is running. Changing the gamma correction value will affect all the windows being displayed using the linear visuals. You must log out of the current window system session and log back in again for the change to take effect. The default is `disable`.

`-help`
Print a list of the `SUNWpfb_config` command line options, along with a brief explanation of each option.

`-offset` *xval yval*
Adjust the position of the secondary stream on the Sun XVR-100. This option is currently implemented only for the `-doublewide` and `-doublehigh` modes. For `-doublewide`, use the *xval* to position the DVI stream, if `-outputs` are direct (VGA stream if `-outputs` are swapped). A negative value specifies the overlapped region with the primary stream. This is similar for `-doublehigh`, except the *yval* is used. Positive values are treated as 0. You must log out of the current window system session and log back in again for the change to take effect. The default is `[0, 0]`.

`-outputs` swapped | direct
Reverse the positions of the `-doublewide` or `-doublehigh` outputs relative to each other on the Sun XVR-100. You must log out of the current window system session and log back in again for the change to take effect. The default is `direct`.

`-propt`
Print the current values of all XVR-50 and Sun XVR-100 options in the OWconfig file specified by the `-file` option for the device specified by the `-dev` option. Print the values of options as they will be in the OWconfig file after the call to `SUNWpfb_config` completes.

This is a typical display for the Sun XVR-100:

```
--- Graphics Configuration for /dev/fbs/pfb0 ---

OWconfig: machine
Video Mode: not set
Depth: not set

Screen Information:
        Doublewide: Disable
        Doublehigh: Disable
        Offset/Overlap: [0.0]
```

```
        Output Configuration: Direct
        Fake8 Rendering: Disable
        Gamma Configuration: Disable
```

`-prconf`
Print the XVR-50 and Sun XVR-100 hardware configuration.

This is a typical display:

```
--- Hardware Configuration for /dev/fbs/pfb0 ---

Type: XVR-100
ASIC: version 0x5159            REV : version 0x3000000
PROM: version 4.2

Monitor/Resolution Information:
        EDID Data: Not Available
        Current resolution setting: 1152x900x66

Depth Information:
        Possible depths:  8, 24
        Current depth: 8
```

`-res` *video-mode* `[ now | try [ noconfirm | nocheck ]]`
Specify the video mode that drives the monitor connected to the specified XVR-50 or Sun XVR-100 device.

Video modes are built-in. The `-res` option requires you to specify the video-mode. You can specify *video-mode* in the format of *width*x*height*x*rate* or as a symbolic name.

video-mode
Specify *video-mode* in the format of *width*x*height*x*rate*, where *width* is the screen width in pixels, *height* is the screen height in pixels, and *rate* is the vertical frequency of the screen refresh. An example video mode specified in this format is `1280x1024x60`.

The `-res` option also accepts formats with `@` preceding the refresh rate instead of `x` (`1280x1024@60`).

Symbolic Names
Some video modes have symbolic names defined for them. Instead of the form *width*x*height*x*rate* format, you can specify one of the symbolic names as the argument to the `-res` option. The meaning of the symbolic name "`none`" is that when the window system is run the screen resolution is the video mode that is currently programmed in the device.

The following symbolic names and their corresponding video modes are supported:

```
Name    Corresponding Video Mode
svga          1024x768x60
```

```
1152          1152x900x76
1280          1280x1024x75
none          Programmed video mode
```

Some video-modes, supported by XVR-50 and Sun XVR-100, might not be supported by the monitor. Use the `-res \?` option to obtain the list of video-modes supported by the XVR-50 and Sun XVR-100 device and the monitor.

The `–res` option also accepts additional, optional arguments immediately following the video mode specification. The following additional, optional arguments are supported:

`nocheck` If present, the normal error checking based on the monitor sense code is suspended. The video mode specified by the user is accepted regardless of whether it is appropriate for the currently attached monitor. This option is useful if a different monitor is to be connected to the XVR-50 or the Sun XVR-100 device. Use of this option implies `noconfirm` as well.

`noconfirm` You could put the system into an unusable state using the `-res` option, which has no video output. To reduce the chance of this, the default behavior of `SUNWpfb_config` is to print a warning message and to ask the user whether to continue. The `noconfirm` option bypasses this confirmation. This option is useful when you are running `SUNWpfb_config` from a shell script.

`now` If present, updates the video mode in the `OWconfig` file and immediately programs the XVR-50 or Sun XVR-100 device to display this video mode. This is useful for changing the video mode before starting the window system.

Do not use this suboption with `SUNWpfb_config` while the configured device is being used (for example, while running the window system) as unpredictable results can occur.

If you want to run `SUNWpfb_config` with the `now` suboption, first bring the window system down, and executed from the console command line. If you use the `now` option within a window system session, the video mode is changed immediately. The width and height of the affected screen will not change until the window system is exited and re-entered again. Additionally, the system might not recognize changes in stereo mode. This usage is discouraged.

`try` If present, programs the specified video mode on a trial basis. You are asked to confirm the video mode by entering a `y` within 10 seconds. You can terminate the trial before 10 seconds by entering any character but `y` or <RETURN>.

This sub-option should not be used with `SUNWpfbconfig` while the configured device is being used (for example, while running the window system) as unpredictable results may occur. To run `SUNWpfbconfig` with the `try` sub-option, the window system should be brought down first, and executed from the console command line.

`-res \?`
Print a list of defined *video-mode* names.

Defaults For a given invocation of `SUNWpfb_config` command line if an option does not appear on the command line, the corresponding OWconfig option is not updated; it retains its previous value.

When the window system is run, if a XVR-50 or Sun XVR-100 option has never been specified by `SUNWpfb_config`, a default value is used. The options and their corresponding defaults for a XVR-50 are as follows:

```
Option                  Default

-dev                    /dev/fb
-file                   machine
-res                    none
-fake8                  disable
-g                      disable
```

The options and their corresponding defaults for a Sun XVR-100 are as follows:

```
Option                  Default

-dev                    /dev/fb
-file                   machine
-res                    none
-fake8                  disable
-g                      disable
-depth                  8 (Solaris 8 and Solaris 9)
-doublewide             disable
-doublehigh             disable
-outputs                direct
-offset                 [0,0]
```

The default for the `-res` option `none` means that when the window system is run, the screen resolution will be the video mode that is currently programmed in the XVR-50 or Sun XVR-100 PROM.

This provides compatibility for users who are used to specifying the device resolution through the XVR-50 or the Sun XVR-100 PROM. On some devices (e.g. GX) this is the only way of specifying the video mode. This means that the PROM ultimately determines the default XVR-50 or Sun XVR-100 video mode.

Examples

EXAMPLE 1 Switching the Monitor Type

The following example switches the monitor type to the resolution of 1280 × 1024 at 60 Hz:

```
example% fbconfig –res 1280x1024x60
```

Files

`/dev/fbs/pfb`*n*	Device special file for XVR-50 or Sun XVR-100 single screen
`/dev/fbs/pfb`*n*`a`	Device special file for the Sun XVR-100 first video out
`/dev/fbs/pfb`*n*`b`	Device special file for the Sun XVR-100 second video out
`/usr/lib/fbconfig/SUNWpfb_config`	Device special file

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpfbcf

See Also

`fbconfig(1M)`, `svccfg(1M)`, `attributes(5)`, `pfb(7D)`

`XSun(1)`, `dtlogin(1X)`

Name SUNWzulu_config – fbconfig module for configuring SunXVR-4000 Graphics Accelerator

Synopsis

```
fbconfig [-dev device-filename] [-file machine | system]
     [options... | -defaults]

fbconfig [-dev device-filename] [-propt] [-prconf]

fbconfig [-dev device-filename] [-list | -help | -res \?]

fbconfig [-dev device-filename]
     [-doublewide enable | disable]
     [-doublehigh enable | disable]
     [-outputs direct | swapped | streamA | streamB]
     [-master a | b] [-clearpixel 0 | 255]

fbconfig [-dev device-filename] [-res video_mode]
     [-multisample available | disable | forceon]
     [-samples samples-per-pixel]
     [-jitter regular | random | permuted | auto]

fbconfig [-dev device-filename] [-stream a | b]
     [-filter cylinder | gaussian | mitchell | catmull]
     -filter_file filter_filename
     [-offset xoff-value yoff-value]
     [-g gamma-correction-value]
     [-slave [ enable | disable] [framelock [internal | external] |
      genlock | bothlock]]
     [-genlock [defaults] [hphase ±hphs] [vphase ±vphs]
     [sync [auto |tip | tri | slice]] [pol [auto | pos | neg]]]
```

Description `SUNWzulu_config` is the Sun XVR-4000 Graphics Accelerator device-dependent layer for `fbconfig(1M)`. It configures the Sun XVR-4000 Graphics Accelerator and some of the X11 window system defaults and some interactions with 3D-accelerated graphics (through OpenGL).

The first through third synopses, above, show the general form of a `SUNWzulu_config` command. The fourth synopsis (with `-res` as the first option) shows card options. The fifth synopsis is for managed-area options. The sixth and last synopsis shows stream options. These option categories—card, managed-area, and streams—are used mainly to explain the `SUNWzulu_config` functions. Where appropriate, you can use options of different types on the same command line.

The Sun XVR-4000 device can support one or two unique video streams (called `stream a` and `stream b`), each of which can drive a display device.

Option Classes The many options that `fbconfig` can select on the Sun XVR-4000 Graphics Accelerator are divided into the following categories:

general options — Shared among different invocation forms or used for query without selecting device settings.

card options	Of the entire XVR-4000 Graphics Accelerator, shared between up to two video streams.
managed area options	Pertain to an area of the frame buffer managed by X and possibly shared between two video streams.
stream options	Specific to a video stream.

Device Usage and Invocation Forms

To use the device to provide a single X managed area with one video stream, use a stream-independent device argument (for example, `-dev zulu0`) without a trailing `a` or `b`. The device name (for example, `/dev/fb` or `/dev/fbs/zulu0`), without any trailing stream indicator should appear on the `Xsun` command line. Stream options will control `stream a` (the only stream used).

To enable two streams from a single X managed area (without needing X's `+xinerama` option), use the device name (for example, `-dev zulu0`) without any trailing `a` or `b`. Enable card option `-doublewide` or `-doublehigh`. Without using the `-stream` option, any stream options you specify are applied to both streams. Stream options can differ between the video streams if `fbconfig` is be run separately for each stream, using the `-stream a | b` option, as shown in EXAMPLES (second example), below.

To use the device's two streams as independent X screens, run `fbconfig` separately for each stream (`-dev zulu0a` and `-dev zulu0b`), as shown in EXAMPLES (third example). Card options `-doublehigh` and `-doublewide` are not available. The device names with trailing stream indicators (for example, `/dev/fbs/zulu0a` and `/dev/fbs/zulu0b`) must be added to the `Xsun` command line to use these independent X screens. The `-stream` option is not needed; the stream is implied by the stream-specific device name.

The `fbconfig` utility checks settings for the two stream devices to assure X can use them simultaneously. Therefore, you might need to use `fbconfig` to reduce resouce consumption (for example, `-samples`) used by one stream's device (for example, `zulu0a`) before you can use `fbconfig` to increase consumption by the other stream's device (for example, `zulu0b`).

The first form of `SUNWzulu_config` shown in SYNOPSIS, above, stores the specified options in the `OWconfig` file associated with the device and (for stream options) the stream. These options are used to initialize the device the next time the window system is started on that device-filename. Updating options in the `OWconfig` file provides persistence of these options across window system sessions and system reboots. You can select the `OWconfig` file that is to be updated using the `-file` option. For `-jitter` and all stream options, the device will also be immediately programmed.

The second form, which invokes any of the `-prconf` and `-propt` options, queries the device for status that is card-specific.

The third form, which invokes the `-help`, `-list`, or `-res \?` options, provides instruction on using `SUNWzulu_config`, a list of available devices, or a list of available resolutions. When using this form, all other options are ignored.

You can specify options for only one device at a time. Specifying options for multiple devices (or multiple independent X managed areas or streams) requires multiple invocations of `fbconfig`.

Only Sun XVR-4000 Graphics Accelerator-specific options can be specified through `SUNWzulu_config`. Window system options for specifying default depth, default visual class, `-nobanner`, and so forth are still specified as device modifiers on the `Xsun` command line when the X server is started, probably in CDE's `Xservers` file. See the `Xsun(1)` man page in the OpenWindows man page collection and `/usr/dt/config/Xservers`.

Options This section is subdivided into general, card, managed area, and stream options.

General Options

`-dev` *device-filename* — Specifies the device's special file, such as `/dev/fbs/zulu0` or the basename such as `zulu0` as a shorthand. The default is `/dev/fb`. See "Device Usage and Invocation Forms," above.

`-file machine|system` — Specifies which `OWconfig` file to update. If *machine* is specified, the machine-specific `OWconfig` file in the `/etc/openwin` directory tree is updated. If *system* is specified, the global `OWconfig` file in the `/usr/openwin` directory tree is updated. If the specified file does not exist, it is created. This option has no effect unless other options are specified. The default is *machine*.

`-defaults` — Resets all option values to their default values, listed in the DEFAULTS section, below. For example, invoking `-defaults` on `zulu0`, `zulu0a`, or `zulu0b` will reset all card, managed area, and stream options for all these `zulu0` subdevices.

`-propt` — Displays the current values of all options in the `OWconfig` file specified by the `-file` option for the device specified by the `-dev` option. Displays the values of options as they will be in the `OWconfig` file after the call to `SUNWzulu_config` completes. The following is an example display:

```
--- OpenWindows Configuration for /dev/fbs/zulu0 ---

OWconfig File: machine

Card:
        Double(wide/high):      disable
        Stream to Port Mapping: direct (Stream A to Port A; B to B)
        Clearpixel Value:       255

Managed Area:
```

```
        Resolution:             SUNW_STD_1280x1024x76
        Samples Per Pixel:      max
        Multisample Mode:       forceon
        Jitter Table:           auto

Video Streams:
   Stream A:
        Offset (x,y):           (0, 0)
        Gamma Correction Value: 2.22
        Filter Type:            mitchell

   Stream B:
        Offset (x,y):           (0, 0)
        Gamma Correction Value: 2.22
        Filter Type:            mitchell

Framelock:
        Framelock/Stereo Port:  Output from Stream A
        Stream A Sync:          Free Run (no frame sync)
        Stream B Sync:          Free Run (no frame sync)
```

`-prconf` Displays the current XVR-4000 hardware configuration, including version numbers of each class of chip. The following is an example display:

```
--- Hardware Configuration for /dev/fbs/zulu0 ---
Type:   XVR-4000 Graphics Accelerator
Part:   501-5588

Memory:
        MAJC:           128MB
        Texture:        1GB total
        3DRAM64:        10.0M samples

Versions:
        Fcode 1.19    MCode 1.4       MAJC 2.1
         FBC3 3.0    Master 1.0   Convolve 0.0
        Sched 1.0       I/O 1.0       FPGA 0.0

Power Level:
        Monitor Power:  On
        Board Power:    On

Video Streams:
   Stream A:
        Current resolution setting: SUNW_STD_1280x1024x76
        Flags:  Allocated Default Primary
        Samples per pixel: 6
```

```
                    Port:   13W3a
                    Monitor/EDID data (13W3)
                             Monitor Manufacturer: SUN
                             Monitor Name:  GDM-5410
                             EDID:  Version 1, Revision 2

               Stream B:
                    Current resolution setting: SUNW_STD_1280x1024x76
                    Flags:  Allocated
                    Samples per pixel: 2
                    Port:   13W3b
                    Monitor/EDID data (13W3)
                             Monitor Manufacturer: SUN
                             EDID:  Version 1, Revision 3
```

`-help` Displays a list of the `SUNWzulu_config` command line options, along with a brief explanation of each.

`-res \?` Displays list of defined video mode names supported by the XVR-4000 Graphics Accelerator and the display device.

Card Options

`-doublewide enable | disable` This option makes it easy for you to combine both streams into one side-by-side virtual display. When enabled with `-outputs direct`, stream a is to the left of stream b. Both streams will use the same video mode defined with the `-res` option. If you specify `disable`, only stream a will be enabled. Enabling `-doublewide` disables `-doublehigh`.

`-doublehigh enable | disable` This option makes it easy for you to combine both streams into one virtual display with one display device above the other. When enabled with `-outputs direct`, stream a is above stream b. Both streams will use the same video mode defined with the `-res` option. If you specify `disable`, only stream a will be enabled. Enabling `-doublehigh` disables `-doublewide`.

`-outputs direct | swapped | streamA | streamB` Controls the internal routing of video streams to output ports (that is, backplane 13W3 connectors). The choices are:

		`direct` Stream a to output port a, stream b to output port b `swapped` Stream a to output port b, stream b to output port a `streamA` Stream a to both output ports `streamB` Stream b to both output ports The default is `direct`. `swapped` can be used to reverse the connectors when `-doublewide` or `-doublehigh` is enabled. The `streamA` and `streamB` arguments are incompatible with stream-specific device names (for example, `zulu0a` or `zulu0b`). When the `-res` option selects an S-video (NTSC or PAL composite) video mode, the svideo output port is automatically selected, sometimes overriding `-outputs` selection.
	`-master a | b`	This option controls which stream drives the `FIELD` and `FRAME_OUT` pins on the device's stereo/sync connector. This pin can drive stereo shutter glasses, and allow another device to framelock to this device's output. The default is `a`. Independent of this option, the `-slave external` option allows a stream to sync to another card by means of this connector's `FIELD_IN` pin.
	`-clearpixel 0 | 255`	Selects the overlay transparent color. This is the pixel value (color index) used by the transparent overlay visual to display the underlay (RGB) pixel contents. The default is `255` (all bits 1), but some applications require `0`. All other color indices display a colormap color.
Managed Area Options	`-res` *video_mode*	The *video_mode* argument specifies resolution and timing information for the display (for example,

SUNW_STD_1280x1024x76). The naming convention for the video mode specifier is:

*origin_type_width*x*height*x*rate*

The elements of the specifier are described as follows:

origin This can be one of:

- SUNW — Sun-derived resolution
- VESA — Video Electronics Standards Association-derived resolution
- *other* — other source

type This can be one or more of:

- STD — normal resolution, usable by most display devices
- DIG — resolution tuned only for LCD flat panels
- INT — interlaced
- STEREO — stereo

width screen width in pixels

height screen height in pixels

rate vertical frequency of the screen refresh (in hertz, that is, video frames per second)

Note that some video modes supported by the XVR-4000 might not be supported by the display device. The list of video modes supported by the device and the display device can be obtained by running SUNWzulu_config with the -res \? option.

-multisample available | disable | forceon

The -multisample option controls whether a multisample buffer is allocated

by the window system and used by OpenGL applications. The suboptions are:

`disable` No multisample rendering is possible. Only one sample per pixel is allocated, despite the `-samples` option value. Furthermore, display filtering is disabled.

`available` Multisample is possible but is selected on a per-application basis. (Each process may choose whether to multisample at the density allocated when the window system started, or not to multisample at all. Intermediate densities are not possible.)

`forceon` Sun OpenGL will use multisample rendering for all applications. There may be a minor performance penalty for rendering at higher sample densities.

Multisample allocation occurs when the window system starts up. This is the only allocation mode supported on the Sun XVR-4000 Graphics Accelerator.

`-samples` *samples-per-pixel*

Specifies the number of samples per pixel to allocate when multisample is not `disable`. Allowable choices are 1 to 16 or `max`, but a very high sample density can be allocated only at low resolution. Setting sample density to 1 is not equivalent to disabling multisampling; samples will still be subject to filtering and jitter. Sample resolutions (without frame rates) and their maximum sample densities follow.

Resolution	Maximum Density	Maximum Balanced Density
width by height	(single stream)	(one stream + another)
1920 by 1200	4 samples	2 + 2 samples
1600 by 1200	5 samples	3 + 2 samples
1600 by 1024	6 samples	3 + 3 samples
1280 by 1024	8 samples	4 + 4 samples
1152 by 900	9 samples	5 + 4 samples
1024 by 800	11 samples	5 + 5 samples
800 by 600	15 samples	7 + 7 samples
768 by 575	15 samples	7 + 7 samples
640 by 480	16 samples	9 + 9 samples

The default is `max`, which means to use the maximum number of samples that can be supported with the amount of memory available, possibly dependent on the video timing (horizontal frequency).

For dual independent streams, if the first stream used by the window system (typically, the first in the `Xservers` file) chooses `max`, it takes most of the memory and video resources. The second stream can then use only a low sample density. If it also chooses `max`, X automatically finds the highest sample density remaining, such as 1 or 2 samples per pixel. To assign sample density more evenly, set each stream's density explicitly. `SUNWzulu_config` allows a combination of resolutions and sample densities only if they will coexist successfully. You might have to reduce one stream's sample density (or choose `max`) before you can increase the other stream's.

`-jitter regular | random | permuted | auto`

Indirectly determines the subpixel (X,Y) locations of the samples stored in the

sample buffer. (The sample density also affects the sample locations.) Choices are:

`regular` Samples are regularly-spaced both vertically and horizontally. The sample locations repeat every pixel or two in X and Y.

`random` Samples are pseudo-randomly (irregularly) spaced within the pixel. The sample locations repeat every 2 pixels in X and Y.

`permuted` Samples are pseudo-randomly spaced within the pixel, and also permuted (stirred) in hardware so that the sample locations repeat every 128 pixels in X and Y. At moderate to high sample density, this choice can improve visual quality. At low sample density, straight lines or edges can appear jagged.

`auto` Automatically selects the best jitter option for the current sample density. This is the default.

The same jitter selection must be used by OpenGL when rendering and by the display subsystem when refreshing the display from the sample buffer. The jitter value is changed immediately in hardware, but any multisamples already in the sample buffer were rendered using the prior jitter selection; that will look incorrect (for example, unstraight lines or edges) if the jitter selection is changed.

When a new OpenGL application starts up, it will render using the new jitter selection. (The window system need not be restarted.) The jitter value is also saved in the `OWconfig` file for the next time the window system starts.

Stream Options `-stream a | b`

Determines whether stream options will be set for stream `a` or stream `b`. The "Device Usage and Invocation Forms" section, above, describes the usage and the default. The `-stream` option is required only to set different stream options for the two video streams enabled using card option `-doublewide` or `-doublehigh`.

`-filter cylinder | gaussian | mitchell | catmull`
`-filter_file` *filter_filename*

There are two ways to configure filtering. The `-filter` option is the simpler. It selects from these predefined filters:

`cylinder` Poorest visual quality, most like a box filter.

`gaussian` Blurriest; suitable for users who want to forgo detail to avoid all visible sampling artifacts.

`mitchell` The best photo-realistic compromise between sharp detail and noticeable blurriness. This filter is the default.

`catmull` The Catmull-Rom filter produces images a little sharper than Mitchell, but are more likely to have visible sampling artifacts, widely known as "jaggies".

The `-filter_file` option allows a user to provide his own filter by producing a filter file and copying or linking it into the directory `/etc/openwin/server/etc/filters` or `/usr/openwin/server/etc/filters`. (Both directories are writable by super-user by default.) The *filter_filename* must not start with `/` or `../` nor contain the substring `/../`, but can contain subdirectory components.

`fbconfig` and X search the directories above in the order listed. If the *filter_filename* is present and valid, the file takes precedence over a predefined filter.

The format of the file is a sequence of floating-point radius and weight values, each value separated by whitespace. Radius values must be monotonically increasing from 0. Weight values must be between -1.0 and +1.0, inclusive. Though more values can be present in the file, only values through radius 2.0 are used. Whitespace and comment lines prefixed with a hash mark (#) are ignored.

Example files contain the (irregular) radius values for which the device uses weight values. The file reader interpolates between existing values if the required radius is not present.

A valid filter option is changed immediately in hardware and saved in the `OWconfig` file for the next time the window system starts. However, when `multisample` is disabled, no filtering occurs.

`-offset` *xoff-value yoff-value*
Offsets the display of the stream (specified by `-stream`) relative to the adjoining edge of the other stream when doublewide or doublehigh is enabled. This can be used to cause an overlap.

xoff-value Number of pixels offset in horizontal direction for the righthand stream when doublewide is enabled. Positive direction is to the right (create a gap); negative is to the left (overlap the streams). Default is 0, which means the two edges abut.

yoff-value Number of pixels offset in vertical direction for the bottom stream when doublehigh is enabled. Positive direction is down (create a gap); negative is up (overlap the streams). Default is 0, which means the two edges touch.

`-g` *gamma-correction-value*
This option changes the gamma correction value. By default the gamma correction value is 2.22. Any value less than zero is illegal. This option can be used while the window system is running. Changing the gamma correction value will affect all the windows being displayed using gamma-corrected visuals. The gamma correction value is also saved in the `OWconfig` file for the next time the window system starts.

`-slave` [`enable` | `disable`]
[`framelock` [`internal` | `external`] | `genlock` | `bothlock`]
This option allows you to enable a synchronization technique for the specified stream. Available techniques:

`framelock` [`internal` | `external`] This provides "asynchronous frame reset": multiple streams all start a frame at roughly the same time. This allows stereo shutters to view the same eye's image from all the synchronized display devices. Using `framelock` requires the incoming synchronization signal have the same frame rate as the stream's video format.

When using `framelock` (or `bothlock`), you can also specify the synchronization source:

`internal` Indicates that the sync source is the other stream of this device.

`external` Indicates the sync is taken from a source outside the device. Using `external` requires a Frame Lock Cable (part number 530-2754) to be

connected. If `-slave enable` is used without specifying a technique, `framelock external` is used.

`genlock` — This provides pixel-accurate horizontal synchronization, which is important in some video mixing situations. Use of `genlock` requires a genlock cable. Use of `bothlock` is recommended, when possible. Certain video formats are incompatible with `genlock`.

`bothlock` — This enables both `framelock` and `genlock` techniques, and requires both framelock (if `external`) and genlock cables.

`-genlock` [`defaults`] [`hphase` ±*hphs*][`vphase` ±*vphs*] [`sync` [`auto`|`tip`|`tri`|`slice`]] [`pol` [`auto`|`pos`|`neg`]]

When `-slave` is enabled and the genlock technique is selected, the selections chosen with the `-genlock` option determine genlock details. These details are used immediately by the hardware, and saved in the `OWconfig` file for future use. Note that they may no longer be desired after changing to a different video format.

`defaults` — Reset all genlock details to their defaults.

`hphase` ±*hphs* — The horizontal phase allows a pixel offset between the external video format and the stream's output. It may be specified as an absolute integer ranging from 0 to the total number of pixel clocks in a horizontal period (active video plus blanked pixels). Or, if the *hphs* starts with a + or -, the value will be added to the current horizontal phase and and saved, modulo the valid range. Small deltas can be used repeatedly until the desired effect is observed.

`vphase` ±*vphs* — The vertical phase allows a scanline offset between the external video format and the stream's output. It can be specified as an absolute integer ranging from 0 to the total number of scanlines in a frame (active video plus blanked scanlines). Or, if the *vphs* starts with a + or -, the value will be added to the current vertical phase and and saved, modulo the valid range. Small deltas can be used repeatedly until the desired effect is observed.

`sync` [`auto`|`tip` | `slice` | `tri`] — This option controls the details of input sync signal sampling, if necessary:

`auto` — Sample the genlock input pulses as most appropriate for the (Sun) video format. This is

the default, and should be used whenever the sync master is also a Sun video format.

`tip` Consider the sync to have occurred at the minimum signal value. This can be used with RS-170 (NTSC or PAL) or with TTL signals.

`slice` Consider the frame sync to have occurred halfway between the minimum and maximum value (sync tip and back porch "blank" levels). This can be used with RS-170 (NTSC or PAL) or with TTL signals.

`tri` Synchronize to a tri-level signal, used by HDTV analog formats.

`pol` [`auto`|`pos`|`neg`] When the sync master is not a Sun video format, it might be necessary to choose which edge of the genlock input sync pulse should be used for genlock.

`auto` Choose rising or falling edge for sync pulse, whichever is most appropriate for the video format. This is the default, and should be used whenever the sync master is also a Sun video format.

`pos` Synchronize with a rising edge of a sync pulse.

`neg` Synchronize with a falling edge of a sync pulse.

Defaults For a given invocation of `SUNWzulu_config`, if an option does not appear on the command line, the corresponding `OWconfig` option is not updated. It retains its previous value.

When the window system starts, if an option has never been specified through `SUNWzulu_config`, a default value is used. The option defaults are as follows:

```
Option Class    Option          Default

General         -dev            /dev/fb
General         -file           machine
                        (/etc/openwin/server/etc/OWconfig)

Card            -doublewide     disable
Card            -doublehigh     disable
Card            -master         a
Card            -outputs        direct
Card            -clearpixel     255

Managed Area    -res            SUNW_STD_1280x1024x76
```

```
Managed Area   -multisample   forceon
Managed Area   -samples       max
Managed Area   -jitter        auto

Stream         -offset        0,0
Stream         -filter        mitchell
Stream         -slave         disable/external/framelock
Stream         -genlock       hphase 0/vphase 0/auto/auto
Stream         -g             2.22
```

Examples

EXAMPLE 1 Switching Resolution of a Monitor

The following example switches to the resolution of 1280 by 1024 at 76 hertz:

```
% fbconfig -dev zulu0 -res SUNW_STD_1280x1024x76
```

EXAMPLE 2 Using Two Side-by-side Monitors with One Large X Screen

The following example enables use of two side-by-side monitors to display together a single large shared X window system "screen" (frame buffer managed area):

```
% fbconfig -dev zulu0 -doublewide enable
```

If the wrong monitor is on the left, they can be swapped in software:

```
% fbconfig -dev zulu0 -outputs swapped
```

A stream option selects a Gaussian (blurry) filter for video stream b:

```
% fbconfig -dev zulu0 -stream b -filter gaussian
```

For the two examples above, the factory-installed `/usr/dt/config/Xservers` file is sufficient (if `/dev/fb` is a link to the Sun XVR-4000 Graphics Accelerator device). If an `/etc/dt/config/Xservers` file exists, for Example 1 or 2, the file would refer to device `zulu0` (not `zulu0a` or `zulu0b`):

```
:0 Local local_uid@console root /usr/openwin/bin/Xsun -dev /dev/fbs/zulu0
```

EXAMPLE 3 Using Two Displays as Independent X Screens

The following example enables use of two displays, each with their own X window system managed frame buffer area and resolution. The larger resolution is not multisampled or filtered, so the smaller resolution will have more samples available to it.

```
% fbconfig -dev zulu0a -res SUNW_STD_1920x1200x75 -multisample disable
% fbconfig -dev zulu0b -res SUNW_STD_1280x1024x76 -samples max
```

In this example, and assuming the display device for stream b is to the right of that for stream a, the `/etc/dt/config/Xservers` file might contain (as one long line):

EXAMPLE 3 Using Two Displays as Independent X Screens *(Continued)*

```
:0 Local local_uid@console root /usr/openwin/bin/Xsun -nobanner
       -dev /dev/fbs/zulu0a -dev /dev/fbs/zulu0b right
```

Files

`/dev/fb`	default device file
`/usr/lib/fbconfig/SUNWzulu_config`	device configuration program
`/etc/openwin/server/etc/filters/`	root file system directory for filter files
`/usr/openwin/server/etc/filters/`	`/usr` file system directory for filter files

An administrator might also have to edit `/etc/dt/config/Xservers`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWzuluc

See Also `fbconfig(1M)`, `attributes(5)`

See the `dtlogin(1)` man page in the CDE man page collection. Also useful is the `Xsun(1)` man page in the OpenWindows man page collection.

Name svcadm – manipulate service instances

Synopsis

```
/usr/sbin/svcadm [-v] enable [-rst] {FMRI | pattern}...

/usr/sbin/svcadm [-v] disable [-st] {FMRI | pattern}...

/usr/sbin/svcadm [-v] restart {FMRI | pattern}...

/usr/sbin/svcadm [-v] refresh {FMRI | pattern}...

/usr/sbin/svcadm [-v] clear {FMRI | pattern}...

/usr/sbin/svcadm [-v] mark [-It] instance_state
     {FMRI | pattern}...

/usr/sbin/svcadm [-v] milestone [-d] milestone_FMRI
```

Description `svcadm` issues requests for actions on services executing within the service management facility (see `smf(5)`). Actions for a service are carried out by its assigned service restarter agent. The default service restarter is `svc.startd` (see `svc.startd(1M)`).

Options The following options are supported:

`-v` Print actions verbosely to standard output.

Subcommands

Common Operations The subcommands listed below are used during the typical administration of a service instance.

For subcommands taking one or more operands, if the operand specifies a service (instead of a service instance), and that service has only a single instance, `svcadm` operates on that instance. If an abbreviated *FMRI* (a fault management resource identifier) or pattern matches more than one service, a warning message is displayed and that operand is ignored. See `smf(5)`.

In the case that the service has more than one instance, `svcadm` return a non-zero exit status.

`enable [-rst]` {*FMRI* | *pattern*}... Enables the service instances specified by the operands. For each service instance, the assigned restarter will try to bring it to the online state. This action requires permission to modify the "general" property group of the service instance (see `smf_security(5)`).

If the `-r` option is specified, `svcadm` enables each service instance and recursively enables its dependencies.

If the `-s` option is specified, `svcadm` enables each service instance and then waits for each service instance to enter the `online` or `degraded` state. `svcadm`

will return early if it determines that the service cannot reach these states without administrator intervention.

If the `-t` option is specified, `svcadm` temporarily enables each service instance. Temporary enable only lasts until reboot. This action requires permission to modify the "restarter_actions" property group of the service instance (see `smf_security(5)`). By default, `enable` is persistent across reboot.

`disable` [`-st`] {*FMRI* | *pattern*}. . .

Disables the service instance specified by the operands. For each service instance, the assigned restarter will try to bring it to the disabled state. This action requires permission to modify the "general" property group of the service instance (see `smf_security(5)`).

If the `-s` option is specified, `svcadm` disables each service instance and then waits for each service instance to enter the disabled state. `svcadm` will return early if it determines that the service cannot reach this state without administrator intervention.

If the `-t` option is specified, `svcadm` temporarily disables each service instance. Temporary disable only lasts until reboot. This action requires permission to modify the "restarter_actions" property group of the service instance (see `smf_security(5)`). By default, `disable` is persistent across reboot.

`restart` {*FMRI* | *pattern*}. . .

Requests that the service instances specified by the operands be restarted. This action requires permission to modify the "restarter_actions" property group of the service instance (see `smf_security(5)`).

This subcommand can restart only those services that are in the online or degraded states, as those states are defined in `smf(5)`.

`refresh` {*FMRI* | *pattern*}. . .

For each service instance specified by the operands, requests that the assigned restarter update the service's running configuration snapshot with the values from the current configuration. Some of these values take effect immediately (for example, dependency changes). Other values do not take effect until the next service `restart`. See the restarter and service documentation for more information.

If the service is managed by svc.startd(1M), the `refresh` method will be invoked if it exists to request the service reread its own configuration. For other restarters, see the restarter documentation.

This action requires permission to modify the "restarter_actions" property group of the service instance (see smf_security(5)).

`clear` {*FMRI* | *pattern*}...

For each service instance specified by the operands, if the instance is in the `maintenance` state, signal to the assigned restarter that the service has been repaired. If the instance is in the `degraded` state, request that the assigned restarter take the service to the `online` state. This action requires permission to modify the "restarter_actions" property group of the service instance (see smf_security(5)).

Exceptional Operations

The following subcommands are used for service development and temporary administrative manipulation.

`mark [-It]` *instance_state* {*FMRI* | *pattern*}...

If *instance_state* is "maintenance", then for each service specified by the operands, `svcadm` requests that the assigned restarter place the service in the `maintenance` state. See svc.startd(1M) and inetd(1M) for a detailed description of the actions taken for each restarter.

If *instance_state* is "degraded", then for services specified by the operands in the online state, `svcadm` requests that the restarters assigned to the services move them into the `degraded` state.

If the `-I` option is specified, the request is flagged as immediate.

The `-t` option is only valid for maintenance requests. When this option is specified, the request is flagged as temporary, and its effect will only last until the next reboot.

`milestone [-d]` *milestone_FMRI*

If *milestone_FMRI* is the keyword "none", all services other than the master restarter,

`svc:/system/svc/restarter:default`, will be temporarily disabled.

If *milestone_FMRI* is the keyword "all", temporary enable and disable requests for all services will be nullified.

If *milestone_FMRI* is one of the following:

```
svc:/milestone/single-user:default
svc:/milestone/multi-user:default
svc:/milestone/multi-user-server:default
```

then temporary enable and disable requests for the indicated service and all services it depends on (directly or indirectly) will be nullified. All other services will be temporarily disabled.

Changing the system's current milestone with the "milestone" subcommand will not change the current run level of the system. To change the system's run level, invoke `/sbin/init` directly.

This action requires permission to modify the "options_ovr" property group of the `svc:/system/svc/restarter:default` service instance (see `smf_security(5)`).

The `-d` option immediately changes the milestone to the requested milestone, as above. Additionally, it makes the specified milestone the default boot milestone, which persists across reboot. The default milestone is defined by the `options/milestone` property on the master restarter, `svc:/system/svc/restarter:default`. If this property is absent, "all" is the default. This action requires permission to modify the "options" property group of the `svc:/system/svc/restarter:default` service instance (see `smf_security(5)`).

Operands The following operands are supported:

FMRI An *FMRI* that specifies one or more instances. *FMRI*s can be abbreviated by specifying the instance name, or the trailing portion of the service name. For example, given the *FMRI*:

```
svc:/network/smtp:sendmail
```

All the following are valid abbreviations:

```
sendmail
:sendmail
smtp
smtp:sendmail
network/smtp
```

While the following are invalid:

```
mail
network
network/smt
```

If the *FMRI* specifies a service, then the command applies to all instances of that service. Abbreviated forms of *FMRI*s are unstable, and should not be used in scripts or other permanent tools.

pattern A pattern that is matched against the *FMRI*s of service instances according to the "globbing" rules described by `fnmatch(5)`. If the pattern does not begin with "svc:", then "svc:/" is prepended.

If an abbreviated *FMRI* or pattern matches more than one service, a warning message is displayed and that operand is ignored.

Examples

EXAMPLE 1 Restarting a Service Instance

The following command restarts the NFS server. The full *FMRI* for the default service instance is: `svc:/network/nfs/server:default`

However, you can abbreviate the full *FMRI* as follows:

```
# svcadm restart nfs/server
```

EXAMPLE 2 Disabling the Standard HTTP Server

The following command disables the standard HTTP server, using an abbreviated *FMRI*:

```
$ svcadm disable http
```

EXAMPLE 3 Enabling an Instance and Its Dependent Instances

The following command enables the `foo:bar` instance, and all instances on which it depends:

```
$ svcadm enable -r foo:bar
```

EXAMPLE 4 Synchronously enabling an instance

The following command enables the `foo:bar` instance. The command will not return until the instance comes online or `svcadm` determines it is not possible for the service to come online.

```
$ svcadm enable -s foo:bar
```

EXAMPLE 5 Restricting and Restoring the Running Services

The following command restricts the running services to single user mode:

```
# svcadm milestone milestone/single-user
```

The following command restores the running services:

```
# svcadm milestone all
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` A fatal error occurred. One or more error messages are displayed on standard error.

`2` Invalid command line options were specified.

`3` `svcadm` determined that a service instance that it was waiting for could not reach the desired state without administrator intervention due to a problem with the service instance itself.

`4` `svcadm` determined that a service instance that it was waiting for could not reach the desired state without administrator intervention due to a problem with the service's dependencies.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	See below.

The interactive output is Uncommitted. The invocation and non-interactive output are Committed.

See Also svcprop(1), svcs(1), inetd(1M), init(1M), svccfg(1M), svc.startd(1M), libscf(3LIB), contract(4), attributes(5), smf(5), smf_security(5)

Notes The amount of time svcadm will spend waiting for services and their dependencies to change state is implicitly limited by their method timeouts. For example, a service using the default restarter whose start method hangs will be transitioned to the maintenance state when its timeout expires. svcadm will then consider it impossible for this service to come online without administrator intervention.

Attempts to synchronously enable a service which depends (directly or indirectly) on a file may fail with an exit status indicating that dependencies are unsatisfied if the caller does not have the privileges necessary to search the directory containing the file. This limitation may be removed in a future Solaris release.

Name svccfg – import, export, and modify service configurations

Synopsis

```
/usr/sbin/svccfg [-v] [-s FMRI]

/usr/sbin/svccfg [-v] [-s FMRI] subcommand [args]...

/usr/sbin/svccfg [-v] [-s FMRI] -f command-file
```

Description The `svccfg` command manipulates data in the service configuration repository. `svccfg` can be invoked interactively, with an individual subcommand, or by specifying a command file that contains a series of subcommands.

Changes made to an existing service in the repository typically do not take effect for that service until the next time the service instance is refreshed. See the `refresh` subcommand on the `svcadm(1M)` man page for more details.

Options The following options are supported:

`-f` *command-file*
Reads and executes `svccfg` subcommands from *command-file*.

`-s` *FMRI*
Selects the entity indicated by *FMRI* (a fault management resource identifier) before executing any subcommands. See `smf(5)`.

`-v`
Verbose.

Subcommands Subcommands are divided into the categories specified in the subsections that follow.

All subcommands that accept *FMRI*s also accept abbreviated or globbed patterns. Instances and services can be abbreviated by specifying the instance name, or the trailing portion of the service name. For example, given the *FMRI*:

```
svc:/network/smtp:sendmail
```

All the following are valid abbreviations:

```
sendmail
:sendmail
smtp
smtp:sendmail
network/smtp
```

While the following are invalid:

```
mail
network
network/smt
```

Abbreviated forms of *FMRI*s are unstable, and should not be used in scripts or other permanent tools. If a pattern matches more than one instance or service, an error message is printed and no action is taken.

General Subcommands

`end`
`exit`
`quit`
Exits immediately.

`repository` *repfile*
Uses *repfile* as a repository. By default, `svccfg` uses the system repository.

Use repository only with files from the identical version of Solaris, including patches, that you are currently running. Do not use this subcommand with the system repository, `/etc/svc/repository.db`.

`set` [`-v`|`-V`]
Sets optional behavior. If no options are specified, set displays the options currently in effect.

`-v`
Turns on verbose mode.

`-V`
Turns off verbose mode.

Service Profile Subcommands

`apply` *file*
If *file* is a service profile, then service instances specified within the file are enabled or disabled according to it. See `smf(5)` for a description of service profiles. This command requires privileges to modify the "general/enabled" property of the service instances. See `smf_security(5)` for the privileges required to modify properties. If *file* is not a service profile, the subcommand fails.

`extract` [> *file*]
Prints a service profile which represents the enabled status of the service instances in the repository to standard output. The output may be redirected to a file.

Service Manifest Subcommands

`archive`
Dumps a full XML service description for all services, instances, and their persistent properties in the repository. This does not include transient properties such as service state, and is suitable for a relocatable repository backup.

`export` *service_FMRI* [>*file*]
The service description for the specified service and its instances is written to standard output or redirected to the given file. Dependencies with a boolean "external" property set to true are omitted in the belief that they were created on behalf of another service.

Without the `-a` option, property groups containing protected information (identified by the presence of the `read_authorization` property—see `smf_security(5)`) will be exported without their property values. When the `-a` option is specified, all values will be archived. An error results if there are insufficient privileges to read these values.

Note that `export` requires a service FMRI. If you specify an instance (including an abbreviation, such as `apache2` or `sendmail`, that specifies an instance), the command fails.

`import` *file*
If *file* is a service manifest, then the services and instances it specifies are imported into the repository. According to the file, dependencies may be created in other services. See `smf(5)` for a description of service manifests. See `smf_security(5)` for the privileges required to create and modify service configurations.

For existing services and instances, properties which have not changed since the last `import` snapshot was taken are upgraded to those specified by the manifest. Conflicts (properties which have been changed both in the repository and the manifest) are reported on the standard error stream. `svccfg` will never upgrade the "general/enabled" and "general/restarter" properties, since they represent administrator preference.

`inventory` *file*
If *file* is determined to be a service manifest, then the FMRIs of the services and instances the *file* describes are printed. For each service, the FMRIs of its instances are displayed before the FMRI of the service.

`validate` *file*
file is processed similarly to `import`, but no changes are made to the repository. If any errors are detected, `svccfg`(1M) exits with a nonzero exit status.

Entity Selection, Modification, and Navigation Subcommands

An "entity" refers to a scope, service, or service instance.

`add` *name*
A new entity with the given name is created as a child of the current selection. See `smf_security(5)` for the privileges required to create entities.

`delete` [`-f`] {*name* | *fmri*}
The named child of the current selection or the entity specified by *fmri* is deleted. Attempts to delete service instances in the "online" or "degraded" state will fail unless the `-f` flag is specified. If a service or service instance has a "dependents" property group of type "framework", then for each of its properties with type "astring" or "fmri", if the property has a single value which names a service or service instance then the dependency property group in the indicated service or service instance with the same name as the property will be deleted. See `smf_security(5)` for the privileges required to delete service configurations.

`list` [*pattern*]
The child entities of the current selection whose names match the glob pattern *pattern* are displayed (see `fnmatch(5)`). ':properties' is also listed for property-bearing entities, namely services and service instances.

`select` {*name* | *fmri*}
If the argument names a child of the current selection, it becomes the current selection. Otherwise, the argument is interpreted as an FMRI and the entity that the argument specifies becomes the current selection.

`unselect`
The parent of the current selection becomes the current selection.

Property Inspection and Modification Subcommands

`addpg` *name type* [*flags*]
Adds a property group with the given *name* and type to the current selection. *flags* is a string of characters which designates the flags with which to create the property group. 'P' represents SCF_PG_FLAG_NONPERSISTENT (see `scf_service_add_pg(3SCF)`). See `smf_security(5)` for the privileges required to create property groups.

`addpropvalue` *pg/name* [*type:*] *value*
Adds the given value to a property. If *type* is given and the property exists, then if *type* does not agree with the property's *type*, the subcommand fails. The values may be enclosed in double-quotes. String values containing double-quotes or backslashes must be enclosed by double-quotes and the contained double-quotes and backslashes must be quoted by backslashes. Nonexistent properties are created, in which case the *type* specifier must be present. See `scf_value_create(3SCF)` for a list of available property types. See `smf_security(5)` for the privileges required to modify properties.

`delpg` *name*
Deletes the property group *name* of the current selection. See `smf_security(5)` for the privileges required to delete property groups.

`delprop` *pg*[*/name*]
Deletes the named property group or property of the current selection. See `smf_security(5)` for the privileges required to delete properties.

`delpropvalue` *pg/name globpattern*
Deletes all values matching the given *glob* pattern in the named property. Succeeds even if no values match. See `smf_security(5)` for the privileges required to modify properties.

`editprop`
Comments of commands to reproduce the property groups and properties of the current selection are placed in a temporary file and the program named by the EDITOR environment variable is invoked to edit it. Upon completion, the commands in the temporary file are executed. The default editor is `vi(1)`. See `smf_security(5)` for the privileges required to create, modify, or delete properties.

`listpg` [*pattern*]
Displays the names, types, and flags of property groups of the current selection. If an argument is given, it is taken as a glob pattern and only property groups with names which match the argument are listed.

`listprop` [*pattern*]
Lists property groups and properties of the current selection. For property groups, names, types, and flags are listed. For properties, names (prepended by the property group name and a slash (/)), types, and values are listed. See `scf_value_create(3SCF)` for a list of available property types. If an argument is supplied it is taken as a glob pattern and only property groups and properties with names which match the argument are listed.

`setenv` [`-i` | `-s`] [`-m` *method_name*] *envvar value*
Sets a method environment variable for a service or instance by changing the "environment" property in the *method_name* property group, if that property group has type "method". If *method_name* is not specified and the `-i` option is used, the "method_context" property group is used, if an instance is currently selected. If the `-s` option is used and a service is currently selected, its "method_context" property group is used. If the `-s` option is used and an instance is currently selected, the "method_context" property group of its parent is used. If neither the `-i` option nor the `-s` option is used, the "start" property group is searched for in the currently selected entity and, if an instance is currently selected, its parent is also searched. If the "inetd_start" property group is not located, it is searched for in a similiar manner.

Once the property is located, all values which begin with *envvar* followed by a "=" are removed, and the value "*envvar=value*" is added. See `smf_security(5)` for the privileges required to modify properties.

`setprop` *pg/name* = [*type:*] *value*
setprop *pg/name* = [*type:*] ([*values* ...])
Sets the *name* property of the *pg* property group of the current selection to the given values of type *type*. See `scf_value_create(3SCF)` for a list of available property types. If the property already exists and the *type* disagrees with the existing *type* on the property, the subcommand fails. Values may be enclosed in double-quotes. String values which contain double-quotes or backslashes must be enclosed by double-quotes and the contained double-quotes and backslashes must be quoted by backslashes. If the named property does not exist, it is created, as long as the type is specified. See `smf_security(5)` for the privileges required to create or modify properties.

`unsetenv` [`-i` | `-s`] [`-m` *method_name*] *envvar value*
Removes a method environment variable for a service or instance by changing the "environment" property in the *method_name* property group, if that property group has type "method". If *method_name* is not specified and the `-i` option is used, the "method_context" property group is used, if an instance is currently selected. If the `-s` option is used and a service is currently selected, its "method_context" property group is used. If the `-s` option is used and an instance is currently selected, the "method_context" property group of its parent is used. If neither the `-i` option nor the `-s` option is used, the "start" property group is searched for in the currently selected entity and, if an instance is currently selected, its parent is also searched. If the "inetd_start" property group is not located, it is searched for in a similiar manner.

Once the property is located, all values which begin with *envvar* followed by "=" are removed. See `smf_security(5)` for the privileges required to modify properties.

Snapshot Navigation and Selection Subcommands

`listsnap`
Displays snapshots available for the currently selected instance.

`revert` [*snapshot*]
Reverts the properties of the currently selected instance and its service to those recorded in the named snapshot. If no argument is given, use the currently selected snapshot and deselect it on success. The changed property values can be made active via the `refresh` subcommand of `svcadm(1M)`. See `smf_security(5)` for the privileges required to change properties.

`selectsnap` [*name*]
Changes the current snapshot to the one named by *name*. If no *name* is specified, deselect the currently selected snapshot. Snapshots are read-only.

Examples

EXAMPLE 1 Importing a Service Description

The following example imports a service description for the `seismic` service in the XML manifest specified on the command line.

```
# svccfg import /var/svc/manifest/site/seismic.xml
```

Note that the manifest must follow the format specified in `service_bundle(4)`.

EXAMPLE 2 Exporting a Service Description

To export a service description on the local system:

```
$ svccfg export dumpadm >/tmp/dump.xml
```

EXAMPLE 3 Deleting a Service Instance

To delete a service instance:

```
$ svccfg delete network/inetd-upgrade:default
```

EXAMPLE 4 Checking Properties in an Alternate Repository

To examine the state of a service's properties after loading an alternate repository, use the sequence of commands shown below. One might use such commands, for example, to determine whether a service was enabled in a particular repository backup.

```
# svccfg
svc:> repository /etc/svc/repository-boot
svc:> select telnet:default
svc:/network/telnet:default> listprop general/enabled
general/enabled  boolean  false
svc:/network/telnet:default> exit
```

EXAMPLE 5 Enabling Debugging

To modify `LD_PRELOAD` for a start method and enable the use of `libumem(3LIB)` with debugging features active:

```
$ svccfg -s system/service setenv LD_PRELOAD libumem.so
$ svccfg -s system/service setenv UMEM_DEBUG default
```

Environmental Variables

EDITOR
The command to run when the `editprop` subcommand is used. The default editor is `vi(1)`.

Exit Status

The following exit values are returned:

`0`
Successful execution.

`1`
One or more subcommands resulted in failure. Error messages are written to the standard error stream.

`2`
Invalid command line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	See below.

The interactive output is Uncommitted. The invocation and non-interactive output are Committed.

See Also

`svcprop(1)`, `svcs(1)`, `svcadm(1M)`, `svc.configd(1M)`, `libscf(3LIB)`, `libumem(3LIB)`, `scf_service_add_pg(3SCF)`, `scf_value_create(3SCF)`, `contract(4)`, `service_bundle(4)`, `attributes(5)`, `fnmatch(5)`, `smf(5)`, `smf_method(5)`, `smf_security(5)`

Name svc.configd – Service Management Facility repository daemon

Synopsis

```
/lib/svc/bin/svc.configd
```

Description `svc.configd` is the repository daemon for the Service Management Facility. `svc.configd` is invoked automatically during system startup, and restarted if any failures occur. `svc.configd` should never be invoked directly.

Interaction with `svc.configd` is by way of `libscf(3LIB)` and the command line tools:`svcs(1)`, `svcprop(1)`, `svcadm(1M)`, and `svccfg(1M)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsr

See Also `svcs(1)`, `svcprop(1)`, `svcadm(1M)`, `svccfg(1M)`, `libscf(3LIB)`, `attributes(5)`

Name svc.startd – Service Management Facility master restarter

Synopsis

```
/lib/svc/bin/svc.startd

svc:/system/svc/restarter:default
```

Description `svc.startd` is the master restarter daemon for Service Management Facility (SMF) and the default restarter for all services. `svc.startd` starts, stops, and restarts services based on administrative requests, system failures, or application failures.

`svc.startd` maintains service state, as well as being responsible for managing faults in accordance with the dependencies of each service.

`svc.startd` is invoked automatically during system startup. It is restarted if any failures occur. `svc.startd` should never be invoked directly.

See `smf_restarter(5)` for information on configuration and behavior common to all restarters.

`svcs(1)` reports status for all services managed by the Service Configuration Facility. `svcadm(1M)` allows manipulation of service instances with respect to the service's restarter.

Environment Variables Environment variables with the "SMF_" prefix are reserved and may be overwritten.

`svc.startd` supplies the "SMF_" environment variables specified in `smf_method(5)` to the method. PATH is set to "`/usr/sbin:/usr/bin`" by default. By default, all other environment variables supplied to `svc.startd` are those inherited from `init(1M)`.

Duplicate entries are reduced to a single entry. The value used is undefined. Environment entries that are not prefixed with "<*name*>=" are ignored.

Restarter Options `svc.startd` is not configured by command line options. Instead, configuration is read from the service configuration repository. You can use `svccfg(1M)` to set all options and properties.

The following configuration variables in the `options` property group are available to developers and administrators:

`boot_messages`
: An *astring* (as defined in `scf_value_is_type`; see `scf_value_create(3SCF)`) that describes the default level of messages to print to the console during boot. The supported message options include `quiet` and `verbose`. The `quiet` option prints minimal messages to console during boot. The `verbose` option prints a single message per service started to indicate success or failure. You can use the `boot -m` option to override the `boot_messages` setting at boot time. See `kernel(1M)`.

`logging`
: Control the level of global service logging for `svc.startd`. An *astring* (as defined in `scf_value_is_type`; see `scf_value_create(3SCF)`) that describes the default level of messages to log to `syslog` (see `syslog(3C)` and `svc.startd`'s global logfile,

/var/svc/log/svc.startd.log. The supported message options include quiet, verbose, and debug. The quiet option sends error messages requiring administrative intervention to the console, syslog and svc.startd's global logfile. The verbose option sends error messages requiring administrative intervention to the console, syslog and svc.startd's global logfile, and information about errors which do not require administrative intervention to svc.startd's global logfile. A single message per service started is also sent to the console. The debug option sends svc.startd debug messages to svc.startd's global logfile, error messages requiring administrative intervention to the console, syslog and svc.startd's global logfile, and a single message per service started to the console.

`milestone`
An FMRI which determines the milestone used as the default boot level. Acceptable options include only the major milestones:

```
svc:/milestone/single-user:default
svc:/milestone/multi-user:default
svc:/milestone/multi-user-server:default
```

or the special values all or none. all represents an idealized milestone that depends on every service. none is a special milestone where no services are running apart from the master svc:/system/svc/restarter:default. By default, svc.startd uses all, a synthetic milestone that depends on every service. If this property is specified, it overrides any initdefault setting in inittab(4).

`system/reconfigure`
Indicates that a reconfiguration reboot has been requested. Services with actions that must key off of a reconfiguration reboot may check that this property exists and is set to 1 to confirm a reconfiguration boot has been requested.

This property is managed by svc.startd and should not be modified by the administrator.

Configuration errors, such as disabling svc.startd are logged by syslog, but ignored.

SERVICE STATES

Services managed by svc.startd can appear in any of the states described in smf(5). The state definitions are unmodified by this restarter.

SERVICE REPORTING

In addition to any logging done by the managed service, svc.startd provides a common set of service reporting and logging mechanisms.

Reporting properties svc.startd updates a common set of properties on all services it manages. These properties are a common interface that can be used to take action based on service instance health. The svcs(1) command can be used to easily display these properties.

`restarter/state`
`restarter/next_state`
The current and next (if currently in transition) state for an instance.

`restarter/auxiliary_state`
A caption detailing additional information about the current instance state. The auxiliary state available for services managed by `svc.startd` is:

maintenance

```
fault_threshold_reached
stop_method_failed
administrative_request
```

`restarter/state_timestamp`
The time when the current state was reached.

`restarter/contract`
The primary process contract ID, if any, that under which the service instance is executing.

Logs

By default, `svc.startd` provides logging of significant restarter actions for the service as well as method standard output and standard error file descriptors to `/var/svc/log/`*service*`:`*instance*`.log`. The level of logging to system global locations like `/var/svc/log/svc.startd.log` and `syslog` is controlled by the `options/logging` property.

SERVICE DEFINITION

When developing or configuring a service managed by `svc.startd`, a common set of properties are used to affect the interaction between the service instance and the restarter.

Methods

The general form of methods for the fork/exec model provided by `svc.startd` are presented in `smf_method(5)`. The following methods are supported as required or optional by services managed by `svc.startd`.

`refresh` Reload any appropriate configuration parameters from the repository or `config` file, without interrupting service. This is often implemented using `SIGHUP` for system daemons. If the service is unable to recognize configuration changes without a restart, no refresh method is provided.

This method is optional.

`start` Start the service. Return success only after the application is available to consumers. Fail if a conflicting instance is already running, or if the service is unable to start.

This method is required.

`stop` Stop the service. In some cases, the stop method can be invoked when some or all of the service has already been stopped. Only return an error if the service is not entirely stopped on method return.

This method is required.

If the service does not need to take any action in a required method, it must specify the `:true` token for that method.

`svc.startd` honors any method context specified for the service or any specific method. The method expansion tokens described in `smf_method(5)` are available for use in all methods invoked by `svc.startd`.

Properties

An overview of the general properties is available in `smf(5)`. The specific way in which these general properties interacts with `svc.startd` follows:

`general/enabled`
If enabled is set to true, the restarter attempts to start the service once all its dependencies are satisfied. If set to false, the service remains in the disabled state, not running.

`general/restarter`
If this FMRI property is empty or set to `svc:/system/svc/restarter:default`, the service is managed by `svc.startd`. Otherwise, the restarter specified is responsible (once it is available) for managing the service.

`general/single_instance`
If `single_instance` is set to true, `svc.startd` only allows one instance of this service to transition to online or degraded at any time.

Additionally, `svc.startd` managed services can define the optional properties listed below in the `startd` property group.

`startd/duration`
The `duration` property defines the service's model. It can be set to `transient`, `child` also known as "`wait`" model services, or `contract` (the default).

`startd/ignore_error`
The `ignore_error` property, if set, specifies a comma-separated list of ignored events. Legitimate string values in that list are `core` and `signal`. The default is to restart on all errors.

`startd/need_session`
The `need_session` property, if set to true, indicates that the instance should be launched in its own session. The default is not to do so.

`startd/utmpx_prefix`
The `utmpx_prefix` string property defines that the instance requires a valid `utmpx` entry prior to start method execution. The default is not to create a `utmpx` entry.

SERVICE FAILURE

`svc.startd` assumes that a method has failed if it returns a non-zero exit code or if fails to complete before the timeout specified expires. If `$SMF_EXIT_ERR_CONFIG` or `$SMF_EXIT_ERR_FATAL` is returned, `svc.startd` immediately places the service in the

maintenance state. For all other failures, `svc.startd` places the service in the offline state. If a service is offline and its dependencies are satisfied, `svc.startd` tries again to start the service (see `smf(5)`).

If a contract or transient service does not return from its start method before its defined timeout elapses, `svc.startd` sends a `SIGKILL` to the method, and returns the service to the offline state.

If three failures happen in a row, or if the service is restarting more than once a second, `svc.startd` places the service in the maintenance state.

The conditions of service failure are defined by a combination of the service model (defined by the `startd/duration` property) and the value of the `startd/ignore_error` property.

A contract model service fails if any of the following conditions occur:

- all processes in the service exit
- any processes in the service produce a core dump
- a process outside the service sends a service process a fatal signal (for example, an administrator terminates a service process with the `pkill` command)

The last two conditions may be ignored by the service by specifying core and/or signal in `startd/ignore_error`.

Defining a service as transient means that `svc.startd` does not track processes for that service. Thus, the potential faults described for contract model services are not considered failures for transient services. A transient service only enters the maintenance state if one of the method failure conditions occurs.

"`Wait`" model services are restarted whenever the child process associated with the service exits. A child process that exits is not considered an error for "`wait`" model services, and repeated failures do not lead to a transition to maintenance state.

LEGACY SERVICES

`svc.startd` continues to provide support for services invoked during the startup run level transitions. Each `/etc/rc?.d` directory is processed after all managed services which constitute the equivalent run level milestone have transitioned to the online state. Standard `init` scripts placed in the `/etc/rc?.d` directories are run in the order of their sequence numbers.

The milestone to run-level mapping is:

`milestone/single-user`	Single-user (`S`)
`milestone/multi-user`	Multi-user (`2`)
`milestone/multi-user-server`	Multi-user with network services (`3`)

Additionally, `svc.startd` gives these legacy services visibility in SMF by inserting an instance per script into the repository. These legacy instances are visible using standard SMF interfaces such as `svcs(1)`, always appear in the `LEGACY-RUN` state, cannot be modified, and can not be specified as dependencies of other services. The initial start time of the legacy service is captured as a convenience for the administrator.

Files

`/var/svc/log`	Directory where `svc.startd` stores log files.
`/etc/svc/volatile`	Directory where `svc.startd` stores log files in early stages of boot, before `/var` is mounted read-write.

Example **EXAMPLE 1** Turning on Verbose Logging

To turn on verbose logging, type the following:

```
# /usr/sbin/svccfg -s system/svc/restarter:default
svc:/system/svc/restarter:default> addpg options application
svc:/system/svc/restarter:default> setprop options/logging = \
astring: verbose
svc:/system/svc/restarter:default> exit
```

This request will take effect on the next restart of `svc.startd`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `svcs(1)`, `svcprop(1)`, `kernel(1M)`, `init(1M)`, `svcadm(1M)`, `svccfg(1M)`, `svc.configd(1M)`, `setsid(2)`, `syslog(3C)`, `libscf(3LIB)`, `scf_value_create(3SCF)`, `contract(4)`, `init.d(4)`, `process(4)`, `inittab(4)`, `attributes(5)`, `smf(5)`, `smf_method(5)`

Name swap – swap administrative interface

Synopsis

```
/usr/sbin/swap -a swapname [swaplow] [swaplen]

/usr/sbin/swap -d swapname [swaplow]

/usr/sbin/swap -l

/usr/sbin/swap -s
```

Description The `swap` utility provides a method of adding, deleting, and monitoring the system swap areas used by the memory manager.

Options The following options are supported:

`-a` *swapname* [*swaplow*] [*swaplen*]

Add the specified swap area. This option can only be used by the superuser or by one who has assumed the Primary Administrator role. *swapname* is the name of the swap area or regular file. For example, on system running a UFS root file system, specify a slice, such as `/dev/dsk/c0t0d0s1`, or a regular file for a swap area. On a system running a ZFS file system, specify a ZFS volume, such as `/dev/zvol/dsk/rpool/swap`, for a swap area. Using a regular file for swap is not supported on a ZFS file system. In addition, you cannot use the same ZFS volume for both the swap area and a dump device when the system is running a ZFS root file system.

swaplow is the offset in 512-byte blocks into the file where the swap area should begin. *swaplen* is the desired length of the swap area in 512-byte blocks. The value of *swaplen* can not be less than `16`. For example, if *n* blocks are specified, then (*n*-1) blocks would be the actual swap length. *swaplen* must be at least one page in length. The size of a page of memory can be determined by using the `pagesize` command. See `pagesize(1)`. Since the first page of a swap file is automatically skipped, and a swap file needs to be at least one page in length, the minimum size should be a multiple of 2 pagesize bytes. The size of a page of memory is machine-dependent.

swaplow + *swaplen* must be less than or equal to the size of the swap file. If *swaplen* is not specified, an area will be added starting at *swaplow* and extending to the end of the designated file. If neither *swaplow* nor *swaplen* are specified, the whole file will be used except for the first page. Swap areas are normally added automatically during system startup by the `/sbin/swapadd` script. This script adds all swap areas which have been specified in the `/etc/vfstab` file; for the syntax of these specifications, see `vfstab(4)`.

To use an NFS or local file system *swapname*, you should first create a file using `mkfile(1M)`. A local file system swap file can now be added to the running system by just running the `swap -a` command. For NFS mounted swap files, the server needs to export the file. Do this by performing the following steps:

1. Add the following line to `/etc/dfs/dfstab`:

   ```
   share -F nfs -o \
   rw=clientname,root=clientname path-to-swap-file
   ```

2. Run shareall(1M).
3. Have the client add the following line to /etc/vfstab:

   server:*path-to-swap-file* - *local-path-to-swap-file* nfs \
   - - - *local-path-to-swap-file* - - swap - - -

4. Have the client run mount:

   # mount *local-path-to-swap-file*

5. The client can then run swap -a to add the swap space:

   # swap -a *local-path-to-swap-file*

-d *swapname*
Delete the specified swap area. This option can only be used by the super-user. *swapname* is the name of the swap file: for example, /dev/dsk/c0t0d0s1 or a regular file. *swaplow* is the offset in 512-byte blocks into the swap area to be deleted. If *swaplow* is not specified, the area will be deleted starting at the second page. When the command completes, swap blocks can no longer be allocated from this area and all swap blocks previously in use in this swap area have been moved to other swap areas.

-l
List the status of all the swap areas. The output has five columns:

path
The path name for the swap area.

dev
The major/minor device number in decimal if it is a block special device; zeroes otherwise.

swaplo
The *swaplow* value for the area in 512-byte blocks.

blocks
The *swaplen* value for the area in 512-byte blocks.

free
The number of 512-byte blocks in this area that are not currently allocated.

The list does not include swap space in the form of physical memory because this space is not associated with a particular swap area.

If swap -l is run while *swapname* is in the process of being deleted (by swap-d), the string INDEL will appear in a sixth column of the swap stats.

-s
Print summary information about total swap space usage and availability:

allocated
The total amount of swap space in bytes currently allocated for use as backing store.

`reserved`
The total amount of swap space in bytes not currently allocated, but claimed by memory mappings for possible future use.

`used`
The total amount of swap space in bytes that is either allocated or reserved.

`available`
The total swap space in bytes that is currently available for future reservation and allocation.

These numbers include swap space from all configured swap areas as listed by the `-l` option, as well swap space in the form of physical memory.

Usage On the 32-bit operating system, only the first 2 Gbytes -1 are used for swap devices greater than or equal to 2 Gbytes in size. On the 64-bit operating system, a block device larger than 2 Gbytes can be fully utilized for swap up to 2^{63} –1 bytes.

Environment Variables See `environ(5)` for descriptions of the following environment variables that affect the execution of `swap`: `LC_CTYPE` and `LC_MESSAGE`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `pagesize(1)`, `mkfile(1M)`, `shareall(1M)`, `getpagesize(3C)`, `vfstab(4)`, `attributes(5)`, `largefile(5)`

Notes For information about setting up a swap area with ZFS, see the *ZFS Administration Guide*.

Warnings No check is done to determine if a swap area being added overlaps with an existing file system.

Name sync – update the super block

Synopsis

```
sync
```

Description `sync` executes the `sync` system primitive. If the system is to be stopped, `sync` must be called to ensure file system integrity. It will flush all previously unwritten system buffers out to disk, thus assuring that all file modifications up to that point will be saved. See `sync(2)` for details.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `sync(2)`, `attributes(5)`

Name syncinit – set serial line interface operating parameters

Synopsis

```
/usr/sbin/syncinit device
     [ [baud_rate] | [keyword=value,]... | [single-word option]]
```

Description The `syncinit` utility allows the user to modify some of the hardware operating modes common to synchronous serial lines. This can be useful in troubleshooting a link, or necessary to the operation of a communications package.

If run without options, `syncinit` reports the options as presently set on the port. If options are specified, the new settings are reported after they have been made.

Options Options to `syncinit` normally take the form of a keyword, followed by an equal sign and a value. The exception is that a baud rate may be specified as a decimal integer by itself. Keywords must begin with the value shown in the options table, but may contain additional letters up to the equal sign. For example, `loop=` and `loopback=` are equivalent.

The following options are supported:

Keyword	Value	Effect
`loop`	yes	Set the port to operate in `internal loopback` mode. The receiver is electrically disconnected from the DCE receive data input and tied to the outgoing transmit data line. Transmit data is available to the DCE. The Digital Phase-Locked Loop (DPLL) may not be used as a clock source in this mode. If no other clocking options have been specified, perform the equivalent of `txc=baud` and `rxc=baud`.
	no	Disable internal loopback mode. If no other clocking options have been specified, perform the equivalent of `txc=txc` and `rxc=rxc`.
`echo`	yes	Set the port to operate in `auto-echo` mode. The transmit data output is electrically disconnected from the transmitter and tied to the receive data input. Incoming receive data is still visible. Use of this mode in combination with local loopback mode has no value, and should be rejected by the device driver. The `auto-echo` mode is useful to make a system become the endpoint of a remote loopback test.
	no	Disable `auto-echo` mode.
`nrzi`	yes	Set the port to operate with `NRZI` data encoding.
	no	Set the port to operate with `NRZ` data encoding.
`txc`	`txc`	Transmit clock source will be the `TxC` signal (pin 15).
	`rxc`	Transmit clock source will be the `RxC` signal (pin 17).
	`baud`	Transmit clock source will be the internal `baud rate generator`.
	`pll`	Transmit clock source will be the output of the `DPLL` circuit.

`rxc`	`rxc`	Receive clock source will be the `RxC` signal (pin 17).
	`txc`	Receive clock source will be the `TxC` signal (pin 15).
	`baud`	Receive clock source will be the internal `baud rate generator`.
	`pll`	Receive clock source will be the output of the `DPLL` circuit.
`speed`	*integer*	Set the baud rate to *integer* bits per second.

There are also several single-word options that set one or more paramaters at a time:

Keyword	Equivalent to Options:
`external`	`txc=txc rxc=rxc loop=no`
`sender`	`txc=baud rxc=rxc loop=no`
`internal`	`txc=pll rxc=pll loop=no`
`stop`	`speed=0`

Examples

EXAMPLE 1 Using `syncinit`

The following command sets the first CPU port to loop internally, using internal clocking and operating at 38400 baud:

```
example# syncinit zsh0 38400 loop=yes
device: /dev/zsh  ppa: 0
speed=38400, loopback=yes, echo=no, nrzi=no, txc=baud, rxc=baud
```

The following command sets the same port's clocking, local loopback and baud rate settings to their default values:

```
example# syncinit zsh0 stop loop=no
device: /dev/zsh  ppa: 0
speed=0, loopback=no, echo=no, nrzi=no, txc=txc, rxc=rxc
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`syncloop(1M)`, `syncstat(1M)`, `Intro(2)`, `ioctl(2)`, `attributes(5)`, `zsh(7D)`

Diagnostics

device `missing minor device number`	The name *device* does not end in a decimal number that can be used as a minor device number.

`bad speed:` *arg*	The string *arg* that accompanied the `speed=` option could not be interpreted as a decimal integer.
`Bad arg:` *arg*	The string *arg* did not make sense as an option.
`ioctl failure code =` *errno*	An `ioctl(2)` system called failed. The meaning of the value of *errno* may be found in `Intro(2)`.

Warnings Do not use `syncinit` on an active serial link, unless needed to resolve an error condition. Do not use this command casually or without being aware of the consequences.

Name syncloop – synchronous serial loopback test program

Synopsis `/usr/sbin/syncloop [-cdlstv]` *device*

Description The `syncloop` command performs several loopback tests that are useful in exercising the various components of a serial communications link.

Before running a test, `syncloop` opens the designated port and configures it according to command line options and the specified test type. It announces the names of the devices being used to control the hardware channel, the channel number (ppa) corresponding to the *device* argument, and the parameters it has set for that channel. It then runs the loopback test in three phases.

The first phase is to listen on the port for any activity. If no activity is seen for at least four seconds, `syncloop` proceeds to the next phase. Otherwise, the user is informed that the line is active and that the test cannot proceed, and the program exits.

In the second phase, called the "first-packet" phase, `syncloop` attempts to send and receive one packet. The program will wait for up to four seconds for the returned packet. If no packets are seen after five attempts, the test fails with an excoriating message. If a packet is returned, the result is compared with the original. If the length and content do not match exactly, the test fails.

The final phase, known as the "multiple-packet" phase, attempts to send many packets through the loop. Because the program has verified the integrity of the link in the first-packet phase, the test will not fail after a particular number of timeouts. If a packet is not seen after four seconds, a message is displayed. Otherwise, a count of the number of packets received is updated on the display once per second. If it becomes obvious that the test is not receiving packets during this phase, the user may wish to stop the program manually. The number and size of the packets sent during this phase is determined by default values, or by command line options. Each returned packet is compared with its original for length and content. If a mismatch is detected, the test fails. The test completes when the required number of packets have been sent, regardless of errors.

After the multiple-packet phase has completed, the program displays a summary of the hardware event statistics for the channel that was tested. The display takes the following form:

```
CRC errors   Aborts   Overruns   Underruns   In<-Drops-> Out
        0         0          0           0   0             0
```

This is followed by an estimated line speed, which is an approximation of the bit rate of the line, based on the number of bytes sent and the actual time that it took to send them.

Options The options for `syncloop` are described in the following table:

Option	Parameter	Default	Description
`-c`	*packet_count*	100	Specifies the number of packets to be sent in the multiple-packet phase.
`-d`	*hex_data_byte*	*random*	Specifies that each packet will be filled with bytes with the value of *hex_data_byte*.
`-l`	*packet_length*	100	Specifies the length of each packet in bytes.
`-s`	*line_speed*	9600	Bit rate in bits per second.
`-v`			Sets verbose mode. If data errors occur, the expected and received data is displayed.
`-t`	*test_type*	*none*	A number, from 1 to 4, that specifies which test to perform. The values for *test_type* are as follows: 1: Internal loopback test. Port loopback is on. Transmit and receive clock sources are internal (baud rate generator). 2: External loopback test. Port loopback is off. Transmit and receive clock sources are internal. Requires a loopback plug suitable to the port under test. 3: External loopback test. Port loopback is off. Transmit and receive clock sources are external (modem). Requires that one of the local modem, the remote modem, or the remote system be set in a loopback configuration. 4: Test using predefined parameters. User defines hardware configuration and may select port parameters using the `syncinit(1M)` command.

All numeric options except `-d` are entered as decimal numbers (for example, `-s 19200`). If you do not provide the `-t` *test_type* option, `syncloop` prompts for it.

Examples

EXAMPLE 1 A sample display of using the `syncloop` command.

In the following command `syncloop` uses a packet length of 512 bytes over the first CPU port:

```
example# syncloop -l 512 zsh0
```

In response to the above command, `syncloop` prompts you for the test option you want.

The following command performs an internal loopback test on the first CPU port, using 5000 packets and a bit rate of 56Kbps:

```
example# syncloop -t 1 -s 56000 -c 5000 zsh0
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also syncinit(1M), syncstat(1M), attributes(5), zsh(7D)

Diagnostics

device `missing minor device number`	The name *device* does not end in a decimal number that can be used as a minor device number.
`invalid packet length:` *nnn*	The packet length was specified to be less than zero or greater than 4096.
`poll: nothing to read`	
`poll: nothing to read or write.`	The poll(2) system call indicates that there is no input pending and/or that output would be blocked if attempted.
`len` *xxx* `should be` *yyy*	The packet that was sent had a length of *yyy*, but was received with a length of *xxx*.
nnn `packets lost in outbound queueing`	
nnn `packets lost in inbound queueing`	A discrepancy has been found between the number of packets sent by `syncloop` and the number of packets the driver counted as transmitted, or between the number counted as received and the number read by the program.

Warnings To allow its tests to run properly, as well as prevent disturbance of normal operations, `syncloop` should only be run on a port that is not being used for any other purpose at that time.

Name syncstat – report driver statistics from a synchronous serial link

Synopsis

```
/usr/sbin/syncstat [-c] device [interval]
```

Description The `syncstat` command reports the event statistics maintained by a synchronous serial device driver. The report may be a single snapshot of the accumulated totals, or a series of samples showing incremental changes. Prior to these it prints the device name being used to query a particular device driver, along with a number indicating the channel number (ppa) under control of that driver.

Event statistics are maintained by a driver for each physical channel that it supports. They are initialized to zero at the time the driver module is loaded into the system, which may be either at boot time or when one of the driver's entry points is first called.

The *device* argument is the name of the serial device as it appears in the `/dev` directory. For example, `zsh0` specifies the first on-board serial device.

The following is a breakdown of `syncstat` output:

`speed`	The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.
`ipkts`	The total number of input packets.
`opkts`	The total number of output packets.
`undrun`	The number of transmitter underrun errors.
`ovrrun`	The number of receiver overrun errors.
`abort`	The number of aborted received frames.
`crc`	The number of received frames with CRC errors.
`isize`	The average size (in bytes) of input packets.
`osize`	The average size (in bytes) of output packets.

Options

`-c` Clear the accumulated statistics for the device specified. This may be useful when it is not desirable to unload a particular driver, or when the driver is not capable of being unloaded.

interval `syncstat` samples the statistics every *interval* seconds and reports incremental changes. The output reports line utilization for input and output in place of average packet sizes. These are the relationships between bytes transferred and the baud rate, expressed as percentages. The loop repeats indefinitely, with a column heading printed every twenty lines for convenience.

Examples

EXAMPLE 1 Sample output from the `syncstat` command:

```
example# syncstat zsh0


speed ipkts opkts undrun ovrrun abort crc isize osize
9600  15716 17121   0      0      1    3   98    89

example# syncstat -c zsh0

speed ipkts opkts undrun ovrrun abort crc isize osize
9600   0     0     0      0      0     0    0     0
```

In the following sample output a new line of output is generated every five seconds:

```
example# syncstat zsh0 5

ipkts opkts undrun ovrrun abort crc iutil outil
12    10      0     0      0     0   5%    4%
22    60      0     0      0     0   3%    90%
36    14      0     0      0     1   51%   2%
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`syncinit(1M)`, `syncloop(1M)`, `attributes(5)`, `zsh(7D)`

Diagnostics

`bad interval:` *arg*	The argument *arg* is expected to be an interval and could not be understood.
device `missing minor device number`	The name *device* does not end in a decimal number that can be used as a minor device number.
`baud rate not set`	The *interval* option is being used and the baud rate on the device is zero. This would cause a divide-by-zero error when computing the line utilization statistics.

Warnings

Underrun, overrun, frame-abort, and CRC errors have a variety of causes. Communication protocols are typically able to handle such errors and initiate recovery of the transmission in which the error occurred. Small numbers of such errors are not a significant problem for most protocols. However, because the overhead involved in recovering from a link error can be much greater than that of normal operation, high error rates can greatly degrade overall link throughput. High error rates are often caused by problems in the link hardware, such as

cables, connectors, interface electronics or telephone lines. They may also be related to excessive load on the link or the supporting system.

The percentages for input and output line utilization reported when using the *interval* option may occasionally be reported as slightly greater than 100% because of inexact sampling times and differences in the accuracy between the system clock and the modem clock. If the percentage of use greatly exceeds 100%, or never exceeds 50%, then the baud rate set for the device probably does not reflect the speed of the modem.

Name sysdef – output system definition

Synopsis

```
/usr/sbin/sysdef [-i] [-n namelist]

/usr/sbin/sysdef [-h] [-d] [-i] [-D]
```

Description The `sysdef` utility outputs the current system definition in tabular form. It lists all hardware devices, as well as pseudo devices, system devices, loadable modules, and the values of selected kernel tunable parameters.

It generates the output by analyzing the named bootable operating system file (*namelist*) and extracting the configuration information from it.

The default system *namelist* is `/dev/kmem`.

Options

`-i` Prints the configuration information from `/dev/kmem`. This is the default and only needs to be specified if the configuration information from both `/dev/kmem` and the system file specified with the "`-n` *namelist*" option is needed.

`-n` *namelist* Specifies a *namelist* other than the default (`/dev/kmem`). The *namelist* specified must be a valid bootable operating system.

`-h` Prints the identifier of the current host in hexadecimal. If `sysdef -h` is executed within a non-global zone and the zone emulates a host identifier, then the zone's host identifier is printed. This numeric value is not guaranteed to be unique.

`-d` The output includes the configuration of system peripherals formatted as a device tree.

`-D` For each system peripheral in the device tree, display the name of the device driver used to manage the peripheral.

Examples **EXAMPLE 1** Sample output format

The following example displays the format of the `sysdef -d` output:

```
example% sysdef -d
Node 'SUNW,Ultra-5_10', unit #-1
    Node 'packages', unit #-1 (no driver)
        Node 'terminal-emulator', unit #-1 (no driver)
        Node 'deblocker', unit #-1 (no driver)
        Node 'obp-tftp', unit #-1 (no driver)
        Node 'disk-label', unit #-1 (no driver)
        Node 'SUNW,builtin-drivers', unit #-1 (no driver)
        Node 'sun-keyboard', unit #-1 (no driver)
        Node 'ufs-file-system', unit #-1 (no driver)
    Node 'chosen', unit #-1 (no driver)
    Node 'openprom', unit #-1 (no driver)
```

EXAMPLE 1 Sample output format *(Continued)*

```
        Node 'client-services', unit #-1 (no driver)
    Node 'options', unit #0
    Node 'aliases', unit #-1 (no driver)
    Node 'memory', unit #-1 (no driver)
    Node 'virtual-memory', unit #-1 (no driver)
    Node 'pci', unit #0
        Node 'pci', unit #0
            Node 'ebus', unit #0
                Node 'auxio', unit #-1 (no driver)
                Node 'power', unit #0
                Node 'SUNW,pll', unit #-1 (no driver)
                Node 'se', unit #0 (no driver)
                Node 'su', unit #0
                Node 'su', unit #1
                Node 'ecpp', unit #-1 (no driver)
                Node 'fdthree', unit #0
                Node 'eeprom', unit #-1 (no driver)
                Node 'flashprom', unit #-1 (no driver)
                Node 'SUNW,CS4231', unit #0 (no driver)
            Node 'network', unit #0
            Node 'SUNW,m64B', unit #0
            Node 'ide', unit #0
                Node 'disk', unit #-1 (no driver)
                Node 'cdrom', unit #-1 (no driver)
                Node 'sd', unit #1
                Node 'dad', unit #1
        Node 'pci', unit #-1 (no driver)
    Node 'SUNW,UltraSPARC-IIi', unit #-1 (no driver)
    Node 'pseudo', unit #0
```

[output truncated]

Files `/dev/kmem` default operating system image

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs

See Also `hostid(1)`, `prtconf(1M)`, `nlist(3ELF)`, `attributes(5)`, `zones(5)`

Name syseventadm – sysevent event specification administration

Synopsis

```
syseventadm add [-R rootdir] [-v vendor] [-p publisher]
     [-c class] [-s subclass] [-u username] path [args]

syseventadm remove [-R rootdir] [-v vendor] [-p publisher]
     [-c class] [-s subclass] [-u username] [path [args]]

syseventadm list [-R rootdir] [-v vendor] [-p publisher]
     [-c class] [-s subclass] [-u username] [path [args]]

syseventadm restart
```

Description The `syseventadm` command is an administrative front-end to add, remove and list `sysevent` event handlers. You can also restart the sysevent daemon by use of the restart command. `syseventadm` can only be run by root.

The `syseventadm add` command adds a handler for a `sysevent` event specified by at least one of vendor, publisher or class. If *class* is specified, it may be qualified with a *sub-class*. Only the values specified for *vendor*, *publisher*, *class* and *sub-class* when adding the handler are matched against `sysevent` events to determine if the specification matches the event and the handler should be run. *path* is the full pathname of the command to be run in response to matching events, with optional arguments (*args*). If username is specified, the command is invoked as user *username*, otherwise as root.

The `syseventadm remove` command removes handlers for matching sysevent event specifications. Event specifications may be matched by specifying at least one of *vendor*, *publisher*, *class*, *username* or *path*. If *class* is specified, it may be qualified with a *sub-class*. Any of *vendor*, *publisher*, *class*, *sub-class*, *username*, *path* or *args* not specified match the corresponding fields of all events. Handlers for all matching specifications are removed.

The `syseventadm list` command lists the handlers for matching `sysevent` event specifications using the same match criteria as the `remove` command but without the requirement that at least one of *vendor*, *publisher*, *class*, *username* or *path* be specified. With no match criteria, all specifications are listed. The list command output format is: [vendor=*vendor*] [publisher=*publisher*] [class=*class*] [subclass=*subclass*] [username=*username*] *path* [*args*] where each of *class*, *sub-class*, *vendor*, *publisher* and *username* is listed only if part of the match criteria for the listed specification.

The `syseventadm restart` command informs the `syseventd` daemon to reread the `sysevent` registry after a change has been made by adding or removing one or more `sysevent` handler specifications.

Argument Macro Substitution The `sysevent` handling facility provides extensive macro capability for constructing the command line arguments to be executed in response to an event. Macro expansion applies only to the command line *args* specified for an event handler, with macros expanded with data from the event itself. Pre-defined macros are provided for the event *class*, *subclass*, *publisher* and *vendor* information. Macros not matching one of the pre-defined macro names cause the

attribute list attached to the event to be searched for an attribute of that name, with the value of the matching attribute substituted on the command line.

Macros are introduced by the `$` character, with the macro name being the following token separated by a SPACE or TAB character. If the macro name is embedded in text, it may be delineated by `${` and `}`. A `\` before the `$` causes macro expansion not to occur.

$*class*	The class string defining the event
$*publisher*	The publisher string defining the event
$*sequence*	The sequence number of the event.
$*subclass*	The subclass string defining the event
$*timestamp*	The timestamp of the event.
$*vendor*	The vendor string defining the event

Macro names other than those pre-defined are compared against the attribute list provided with the event. An attribute with name matching the macro name causes the value of the attribute to be substituted as ASCII text on the generated command line.

Use of a macro for which no attribute with that name is defined, or for which multiple attributes with that name are provided, cause an error and the command is not invoked.

Attributes with signed data types (`DATA_TYPE_INT16`, `DATA_TYPE_INT32` and `DATA_TYPE_INT64`) are expanded as decimal digits.

Attributes with unsigned data types (`DATA_TYPE_BYTE`, `DATA_TYPE_UINT16`, `DATA_TYPE_UINT32`, `DATA_TYPE_UINT64` and `DATA_TYPE_HTTIME`) are expanded as hexadecimal digits with a `0x` prefix.

Attributes with string data type (`DATA_TYPE_STRING`) are expanded with the string data. The data is not quoted. If if it desired that the quoted strings be generated on the command line, put quotes around the macro call in the arguments.

Array types are expanded with each element expanded as defined for that scalar type, with a space separating each element substitution.

Options The `add`, `list` and `remove` subcommands support the following options:

`-c` *class*	Specify the event class, *class*.
`-p` *publisher*	Specify the event publisher, *publisher*.
`-R` *rootdir*	Specify an alternate root path, *rootdir*.

Note – The root file system of any non-global zones must not be referenced with the `-R` option. Doing so might damage the global zone's file system,

might compromise the security of the global zone, and might damage the non-global zone's file system. See zones(5).

`-s` *subclass* Specify the event subclass, *subclass*.

`-u` *username* Specify the username (*username*) to invoke the command.

`-v` *vendor* Specify the vendor (*vendor*) that defines the event. Events defined by third-party software should specify the company's stock symbol as *vendor*. Sun-defined events use `SUNW`.

Operands The `add`, `list` and `remove` subcommands support the following options:

args Command arguments

path Full path of command to be run in response to event

Examples **EXAMPLE 1** Adding an Event Handler

The following example adds an event handler for an event defined by vendor `MYCO` ("My Company"), class `EC_ENV` and sub-class `ESC_ENV_TEMP`. The command to be run is `/opt/MYCOenv/bin/ec_env_temp`, with arguments being the class name, sub-class name and pathname derived from the event attributes. The `$` characters are preceded by a backslash to circumvent shell interpretation. There is no need to restart the service after the change since the registry is maintained on `$ALTROOT`.

```
# syseventadm add -R \$ALTROOT -v MYCO -c EC_ENV -s ESC_ENV_TEMP \
     /opt/MYCOenv/bin/ec_env_temp \$class \$subclass \$pathname
```

Note the caveat on the use of the `-R` option in the description of that option, above.

EXAMPLE 2 Removing an Event Handler

The following example removes the event handler added in Example 1.

```
# syseventadm remove -R \$ALTROOT -v MYCO -c EC_ENV -s ESC_ENV_TEMP \
     /opt/MYCOenv/bin/ec_env_temp \${class} \${subclass} \${pathname}
```

Note the caveat on the use of the `-R` option in the description of that option, above.

EXAMPLE 3 Listing Event Handlers

The following example lists all event handlers for events of class `EC_ENV`, subclass `ESC_ENV_TEMP`, as defined by vendor `MYCO`:

```
# syseventadm list -v MYCO -c EC_ENV -s ESC_ENV_TEMP \
     vendor=MYCO class=EC_ENV subclass=ESC_ENV_TEMP \
     /opt/MYCOenv/bin/ec_env_temp \${class} \${subclass} \${pathname}
```

EXAMPLE 4 Listing Event Handlers

The following example lists all event handlers defined by vendor `VRTS`.

```
# syseventadm list -v VRTS
```

EXAMPLE 5 Removing Event Handlers

The following example removes all event handlers defined by vendor `VRTS`, and restarts service.

```
# syseventadm remove -v VRTS
# syseventadm restart
```

EXAMPLE 6 Listing All Event Handlers Specified to Run a Command

The following example lists all event handlers specified to run the command `/opt/MYCOenv/bin/ec_env_temp`:

```
# syseventadm list /opt/MYCOenv/bin/ec_env_temp
```

EXAMPLE 7 Removing Event Handlers and Restarting Service

The following example removes all event handlers specified to run the command `/opt/MYCOenv/bin/ec_env_temp`, and restarts service:

```
# syseventadm remove /opt/MYCOenv/bin/ec_env_temp
# syseventadm restart
```

Exit Status

The following exit values are returned:

`0` Successful completion.

`1` No matching event specification found (remove or list commands only).

`2` Incorrect command usage.

`3` Permission denied.

`4` Command failed.

`5` Out of memory.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`syseventd(1M)`, `sysevent_post_event(3SYSEVENT)`, `attributes(5)`, `ddi_log_sysevent(9F)`

Notes To avoid upgrade problems, packages delivering a sysevent event handler should install the event handler by running `syseventadm` from the package's postinstall script. The event handler can then be removed by running `syseventadm` from the package's preremove script using the same arguments as when added.

Name syseventconfd – kernel system event command invocation daemon

Synopsis

```
/usr/lib/sysevent/syseventconfd [-r rootdir]
```

Description `syseventconfd` is the user-level daemon that invokes user-level commands in response to kernel system events received from `syseventd(1M)`.

Options The following options are supported:

`-r` *rootdir* Cause `syseventconfd` to use an alternate root path when creating its door. The root path must match the root path used to invoke `syseventd`.

Files

`/etc/sysevent/syseventconfd_event_service`
`syseventconfd` event service door file

`/usr/lib/sysevent/modules/sysevent_conf_mod.so`
`syseventd` loadable module (SLM) managing `sysevent.conf` files

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `svcs(1)`, `svcadm(1M)`, `syseventd(1M)`, `attributes(5)`, `smf(5)`

Notes The `syseventconfd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/sysevent:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name syseventd – kernel system event notification daemon

Synopsis `/usr/lib/sysevent/syseventd [-d` *debug_level*`] [-r` *rootdir*`]`

Description `syseventd` is a user-level daemon that accepts delivery of system event buffers from the kernel. Once an event buffer has been delivered to `syseventd`, it, in turn, attempts to propagate the event to all interested end event subscribers.

Event subscribers take the form of a `syseventd` loadable module (SLM). syseventd passes the event buffer to each of its subscribers and in return expects a notification as to the successful or unsuccessful delivery attempt.

Upon successful delivery of the event buffer to all interested event subscribers, `syseventd` frees the event buffer from the kernel event queue.

Options The following options are supported:

`-d` *debug_level* Enable debug mode. Messages are printed to the invoking user's terminal.

`-r` *rootdir* Cause `syseventd` to use an alternate root path when creating its door and lock files. Modules continue to be loaded from the standard module directories.

Exit Status The following exit values are returned:

`0` Successful completion.

`non-zero` An error occurred.

Files `/etc/sysevent/syseventd_daemon.lock`
daemon lock file

`/etc/sysevent/sysevent_door`
kernel to `syseventd` door file

`/usr/lib/sysevent/modules`
SLM directory repository

`/usr/platform/‘uname –i‘/lib/sysevent/modules`
SLM directory repository

`/usr/platform/‘uname –m‘/lib/sysevent/modules`
SLM directory repository

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also svcs(1), svcadm(1M), syseventconfd(1M), attributes(5), smf(5)

Notes The sysevent d service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/sysevent:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name sysidconfig – execute system configuration applications, or define set of system configuration applications

Synopsis

```
sysidconfig [-lv] [-a application] [-b basedir]
     [-r application]
```

Description Invoked without any options, the `sysidconfig` program executes a list of applications. An application on this list is referred to as a "system configuration application." Every application on this list will be passed one command-line argument, `-c`. This flag will cause the system configuration application to perform its configuration function. Without options, `sysidconfig` should only be invoked by startup scripts, which occurs during the initial installation and during a reconfigure reboot.

All applications on the list will be executed, if possible. All activity taken by the `sysidconfig` program is logged in the `sysidconfig` log file, `/var/log/sysidconfig.log`. If one or more of the applications on the list are either not present at execution time, are not executable, or execute but return a failure code upon completion, then that information will be logged as well. Successful completion of the program can be assumed if no error message is present in the log file. Programs are executed sequentially, with only one configuration application active at a time.

Executed with the `-l`, `-a`, or `-r` options, the `sysidconfig` program allows the super-user to list the defined configuration applications, and to add items to or remove items from that list. Running `sysidconfig` with options is the only way to view or manipulate the list. Only the super-user can execute the `sysidconfig` program with options.

The `-b` and `-v` options change the behavior of `sysidconfig`, and can be used with or without the list manipulation options discussed above. The `-b` *basedir* option is used to specify a reference `root` directory other than the default, `/`. The `-v` option duplicates the log file output on `stdout`.

By default, no SPARC based applications exist on this list. However, the x86 based systems are delivered with one application, `kdmconfig(1M)`, on the list. `kdmconfig` is not delivered on SPARC based systems.

This application is an extension of the `sysidtool(1M)` suite of programs. It is executed during initial installation and during a reconfigure reboot, before the window system has been started. Graphical User Interface (GUI) applications will not execute successfully if they are added to the list of configuration applications via `sysidconfig -a`.

This program is referenced, but not fully described, in the `sysidtool(1M)` manual page.

Options The valid options are:

`-a` *application* Add the named application to the list of defined applications. When next invoked without arguments, `sysidconfig` will run this newly added application after all previously defined applications. *application* must be a fully qualified path name that is not currently on the list of applications to execute.

`-b` *basedir* Specify an alternate base directory (`/` is defined as the default base directory if no other is specified). The specified directory is used as the `root` directory when adding, listing, removing, or executing configuration applications. The log file where information is recorded is in `/var/log`, relative to the specified *basedir*. In the log file, the *basedir* is not noted. This means, for example, that if the super-user on a diskless client's server executes:

```
sysidconfig -b /export/root/client -a /sbin/someapp
```

then the diskless client *client* would have `/sbin/`*someapp* executed upon reconfigure reboot. The diskless client's log file would note that `/sbin/`*someapp* was added, not `/export/root/client/sbin/`*someapp*.

Note – The root file system of any non-global zones must not be referenced with the `-b` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-l` List defined configuration applications. Applications will be executed one at a time, in the order shown in the list.

`-r` *application* Remove the named application from the list of defined applications. *application* must be a fully qualified path name and it must be on the existing list of applications to execute.

`-v` Verbose mode. This option echoes all information sent to the log file to `stdout`. Such information includes timestamp information about when the program was executed, the names of applications being executed, and results of those executions.

Return Values The `sysidconfig` program will return `0` if it completes successfully.

When executed with the `-r` or `-a` options, error conditions or warnings will be reported on `stderr`. If the requested action completes successfully, an exit code of `0` will be returned.

Errors `EPERM` The program was executed by a user other than the super-user.

`EINVAL` Option `-l`, `-a`, or `-r` was passed and the action could not be completed successfully.

Files `/var/log/sysidconfig.log` `sysidconfig` log file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmap

See Also sys-unconfig(1M), sysidtool(1M), attributes(5)

x86 Only kdmconfig(1M)

Diagnostics When run without options, a log of the `sysidconfig` program's activity can be found in `/var/log/sysidconfig.log`. This file contains a timestamp log of each program executed, its resulting `stderr` output, and its exit code. If an application in the list was not found or is not executable, that will also be noted.

Name sysidtool, sysidnet, sysidns, sysidsys, sysidroot, sysidpm, sysidnfs4, sysidkbd – system configuration

Synopsis

```
/usr/sbin/sysidnet

/usr/sbin/sysidns

/usr/sbin/sysidsys

/usr/sbin/sysidroot

/usr/sbin/sysidpm

/usr/sbin/sysidnfs4

/usr/sbin/sysidkbd
```

Description `sysidtool` is a suite of programs that configure a new system, or one that has been unconfigured with `sys-unconfig(1M)`. The `sysidtool` programs run automatically at system installation, or during the first boot after a machine has been successfully unconfigured.

These programs have no effect except at such times, and should never be run manually.

The `sysidtool` programs set up the appropriate information in the machine's configuration files, in the kernel, and on the machine's network interface. The following list shows the available commands and the information for which each of the commands lists.

`sysidnet`: network configuration
Machine's default locale. Machine's console type. Machine's host name. Machine's IP address.

`sysidns`: name service configuration
Name service choice: NIS, NIS+, DNS, LDAP, or none. Machine's IP subnet mask (if no NIS/NIS+ server can automatically be located on the machine's sub-network. Domain name for chosen name service. Hostname and IP address of name server(s). DNS search list (DNS name service only)

`sysidsys`: miscellaneous system configuration
Machine's IP subnet mask (if an NIS/NIS+ server was automatically located on the machine's sub-network). Machine's time zone. Date and time.

`sysidroot`: control superuser information
Machine's root password.

`sysidpm`: power management configuration
Auto-shutdown confirmation if the system is Energystar-V2 compliant, that is, a new system model shipped after October 1, 1995.

`sysidnfs4`: NFSv4 domain configuration
Domain name to be used by NFSv4 client(s) and server(s) to transmit user and group id's as strings of the general form "[*user*|*group*]@*domain*".

`sysidkbd`: keyboard layout configuration
: The corresponding keytable is loaded into the kernel according to the configured keyboard layout.

`sysidconfig`: host– or platform-specific configuration
: This command controls specification and execution of custom configuration applications that can be specified for a particular host or a particular platform. See `sysidconfig(1M)`.

The `sysidtool` programs attempt to obtain system configuration information from various name service databases, for example, NIS, or from the `sysidcfg(4)` file, and you are prompted to provide the information if it cannot be found. However, you can avoid one or more of the prompts by preconfiguring the appropriate configuration information in the name service databases or in the `sysidcfg(4)` file.

To preconfigure the information in the name service databases, you must use the name service commands or the Solstice AdminSuite tools. See *Solaris 10 Installation Guide: Basic Installations* for more details about how to preconfigure the system configuration information.

The machine's configuration information is set up in its `/etc` and `/var` files.

If a system has more than one network interface, you can use `sysidtool` to configure all interfaces on the system.

You cannot use the name service databases or the `sysidcfg(4)` file to suppress the Power Management configuration prompt. However, you can suppress it by creating either the `/autoshutdown` or `/noautoshutdown` file before installation reboot. Accordingly, the auto-shutdown feature is silently configured. The `/autoshutdown` or `/noautoshutdown` files are removed by `sysidpm` before it exits.

If you have non-global zones installed and the `nfs4_domain` keyword exists in the `sysidcfg` file, the first boot of a non-global zone sets the domain. Otherwise, the Solaris interactive installation program comes up and you are prompted to provide a domain name before the boot process completes. See `sysidcfg(4)` and *Solaris 10 Installation Guide: Custom JumpStart and Advanced Installations*.

`sysidkbd` prompts for the keyboards with `zero-bCountryCode` and then sets the keyboard layout string in the `/etc/default/kbd` file. If the `sysidkbd` gets the valid keyboard layout string, the string will be set into the entry “LAYOUT=” in the file.

Files

```
/etc/.UNCONFIGURED
/etc/nodename
/etc/hostname.??[0-9]
/etc/default/init
/etc/defaultdomain
```

`/etc/passwd` password file. See `passwd(4)`.

`/etc/shadow` shadow password file. See `shadow(4)`.

```
/etc/inet/hosts
/etc/inet/netmasks
/var/nis/NIS_COLD_START
/var/yp/aliases
/var/yp/binding/*/ypservers
/etc/.sysIDtool.state
```

`/etc/power.conf` Power Management configuration file. See `power.conf(4)`.

`/etc/.PM_RECONFIGURE` If this file is present during system reboot, the `sysidpm` program is run. This file is removed by `sysidpm`.

```
/etc/.NFS4inst_state.domain
/etc/default/kdb
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmap, SUNWpmu

See Also `powerd(1M)`, `sys-unconfig(1M)`, `sysidconfig(1M)`, `passwd(4)`, `power.conf(4)`, `shadow(4)`, `sysidcfg(4)`, `attributes(5)`

Solaris 10 Installation Guide: Custom JumpStart and Advanced Installations

Notes NIS+ might not be supported in future releases of the Solaris Operating system. Tools to aid the migration from NIS+ to LDAP are available in the current Solaris release. For more information, visit http://www.sun.com/directory/nisplus/transition.html.

Name syslogd – log system messages

Synopsis

```
/usr/sbin/syslogd [-d] [-f configfile] [-m markinterval]
     [-p path] [-t | -T]
```

Description `syslogd` reads and forwards system messages to the appropriate log files or users, depending upon the priority of a message and the system facility from which it originates. The configuration file `/etc/syslog.conf` (see `syslog.conf(4)`) controls where messages are forwarded. `syslogd` logs a mark (timestamp) message every *markinterval* minutes (default `20`) at priority `LOG_INFO` to the facility whose name is given as `mark` in the `syslog.conf` file.

A system message consists of a single line of text, which may be prefixed with a priority code number enclosed in angle-brackets (`< >`); priorities are defined in `<sys/syslog.h>`.

`syslogd` reads from the STREAMS log driver, `/dev/log`, and from any transport provider specified in `/etc/netconfig`, `/etc/net/transport/hosts`, and `/etc/net/transport/services`.

`syslogd` reads the configuration file when it starts up, and again whenever it receives a `HUP` signal (see `signal.h(3HEAD)`, at which time it also closes all files it has open, re-reads its configuration file, and then opens only the log files that are listed in that file. `syslogd` exits when it receives a `TERM` signal.

As it starts up, `syslogd` creates the file `/var/run/syslog.pid`, if possible, containing its process identifier (PID).

If message ID generation is enabled (see `log(7D)`), each message will be preceded by an identifier in the following format: `[ID` *msgid facility*`.`*priority*`]`. *msgid* is the message's numeric identifier described in `msgid(1M)`. *facility* and *priority* are described in `syslog.conf(4)`. `[ID 123456 kern.notice]` is an example of an identifier when message ID generation is enabled.

If the message originated in a loadable kernel module or driver, the kernel module's name (for example, `ufs`) will be displayed instead of `unix`. See `EXAMPLES` for sample output from `syslogd` with and without message ID generation enabled.

In an effort to reduce visual clutter, message IDs are not displayed when writing to the console; message IDs are only written to the log file. See Examples.

The `/etc/default/syslogd` file contains the following default parameter settings, which are in effect if neither the `-t` nor `-T` option is selected. See `FILES`.

The recommended way to allow or disallow message logging is through the use of the service management facility (`smf(5)`) property:

```
svc:/system/system-log/config/log_from_remote
```

This property specifies whether remote messages are logged. `log_from_remote=true` is equivalent to the `-t` command-line option and `false` is equivalent to the `-T` command-line option. The default value for `-log_from_remote` is `false`. See NOTES, below.

`LOG_FROM_REMOTE`
Specifies whether remote messages are logged. `LOG_FROM_REMOTE=NO` is equivalent to the `-t` command-line option. The default value for `LOG_FROM_REMOTE` is `YES`.

Options

The following options are supported:

`-d`
Turn on debugging. This option should only be used interactively in a root shell once the system is in multi-user mode. It should *not* be used in the system start-up scripts, as this will cause the system to hang at the point where `syslogd` is started.

`-f` *configfile*
Specify an alternate configuration file.

`-m` *markinterval*
Specify an interval, in minutes, between mark messages.

`-p` *path*
Specify an alternative log device name. The default is `/dev/log`.

`-T`
Enable the `syslogd` UDP port to turn on logging of remote messages. This is the default behavior. See Files.

`-t`
Disable the `syslogd` UDP port to turn off logging of remote messages. See Files.

Examples

EXAMPLE 1 `syslogd` Output Without Message ID Generation Enabled

The following example shows the output from `syslogd` when message ID generation *is not* enabled:

```
Sep 29 21:41:18 cathy unix: alloc /: file system full
```

EXAMPLE 2 `syslogd` Output with ID generation Enabled

The following example shows the output from `syslogd` when message ID generation *is* enabled. The message ID is displayed when writing to log file`/var/adm/messages`.

```
Sep 29 21:41:18 cathy ufs: [ID 845546 kern.notice]
                                    alloc /: file system full
```

EXAMPLE 3 `syslogd` Output with ID Generation Enabled

The following example shows the output from `syslogd` when message ID generation *is* enabled when writing to the console. Even though message ID is enabled, the message ID is not displayed at the console.

```
Sep 29 21:41:18 cathy ufs: alloc /: file system full
```

EXAMPLE 4 Enabling Acceptance of UDP Messages from Remote Systems

The following commands enable `syslogd` to accept entries from remote systems.

```
# svccfg -s svc:/system/system-log setprop config/log_from_remote = true
# svcadm restart svc:/system/system-log
```

Files

`/etc/syslog.conf`
Configuration file

`/var/run/syslog.pid`
Process ID

`/etc/default/syslogd`
Contains default settings. You can override some of the settings by command-line options.

`/dev/log`
STREAMS log driver

`/etc/netconfig`
Transport providers available on the system

`/etc/net/transport/hosts`
Network hosts for each transport

`/etc/net/transport/services`
Network services for each transport

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`logger(1)`, `svcs(1)`, `msgid(1M)`, `svcadm(1M)`, `svccfg(1M)`, `syslog(3C)`, `syslog.conf(4)`, `attributes(5)`, `signal.h(3HEAD)`, `smf(5)`, `log(7D)`

Notes

The `mark` message is a system time stamp, and so it is only defined for the system on which `syslogd` is running. It can not be forwarded to other systems.

When `syslogd` receives a `HUP` signal, it attempts to complete outputting pending messages, and close all log files to which it is currently logging messages. If, for some reason, one (or more) of these files does not close within a generous grace period, `syslogd` discards the pending messages, forcibly closes these files, and starts reconfiguration. If this shutdown procedure is disturbed by an unexpected error and `syslogd` cannot complete reconfiguration, `syslogd` sends a mail message to the superuser on the current system stating that it has shut down, and exits.

Care should be taken to ensure that each window displaying messages forwarded by `syslogd` (especially console windows) is run in the system default locale (which is `syslogd`'s locale). If

this advice is not followed, it is possible for a `syslog` message to alter the terminal settings for that window, possibly even allowing remote execution of arbitrary commands from that window.

The `syslogd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/system-log:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

When `syslogd` is started by means of `svcadm(1M)`, if a value is specified for `LOG_FROM_REMOTE` in the `/etc/defaults/syslogd` file, the SMF property `svc:/system/system-log/config/log_from_remote` is set to correspond to the `LOG_FROM_REMOTE` value and the `/etc/default/syslogd` file is modified to replace the `LOG_FROM_REMOTE` specification with the following comment:

```
# LOG_FROM_REMOTE is now set using svccfg(1m), see syslogd(1m).
```

If neither `LOG_FROM_REMOTE` nor `svc:/system/system-log/config/log_from_remote` are defined, the default is to log remote messages.

On installation, the initial value of `svc:/system/system-log/config/log_from_remote` is `false`.

Name sys-unconfig – undo a system's configuration

Synopsis `/usr/sbin/sys-unconfig`

Description The `sys-unconfig` command is used to restore a system's configuration to an "as-manufactured" state, ready to be reconfigured again. The system's configuration consists of hostname, Network Information Service (NIS) domain name, timezone, IP address, IP subnet mask, and root password. This operation is the inverse of those performed by the `sysidnet(1M)`, `sysidns(1M)`, and `sysidsys(1M)` programs run at boot. See `sysidtool(1M)`.

`sys-unconfig` does the following:

- Saves current `/etc/inet/hosts` file information in `/etc/inet/hosts.saved`.
- If the current `/etc/vfstab` file contains NFS mount entries, saves the `/etc/vfstab` file to `/etc/vfstab.orig`.
- Restores the default `/etc/inet/hosts` file.
- Removes the default hostname in `/etc/hostname.`*interface* files for all interfaces configured when this command is run. To determine which interfaces are configured, run the command '`ifconfig -a`'. The `/etc/hostname.`*interface* files corresponding to all of the interfaces listed in the resulting output, with the exception of the loopback interface (lo0), will be removed.
- Removes the default domainname in `/etc/defaultdomain`.
- Restores the timezone to PST8PDT in `/etc/TIMEZONE`.
- Disables the Network Information Service (NIS) and Network Information Service Plus (NIS+) if either NIS or NIS+ was configured.
- Removes the file `/etc/inet/netmasks`.
- Removes the file `/etc/defaultrouter`.
- Removes the password set for root in `/etc/shadow`.
- Removes the file `/etc/.rootkey`.
- Executes all system configuration applications. These applications are defined by prior executions of a `sysidconfig -a` *application*. (See `sysidconfig(1M)`). When `sys-unconfig` is run, all system configuration applications are passed one argument, `-u`.
- Removes the file `/etc/resolv.conf`.
- Removes the file `/etc/sysidcfg`.
- Disables LDAP by removing `/var/ldap/ldap_client_cache`, `/var/ldap/ldap_client_file`, `/var/ldap/ldap_client_cred`, and `/var/ldap/cachemgr.log`.
- Regenerates keys for `sshd(1M)`.

When `sys-unconfig` is finished, it performs a system shutdown. `sys-unconfig` is a potentially dangerous utility and can only be run by the super user.

Files

`/etc/default/init`	process control initialization
`/etc/defaultdomain`	determines host's domain name
`/etc/defaultrouter`	specifies an IPv4 host's default router
`/etc/hostname.`*interface*	identifies symbolic host name associated with network interface *interface*
`/etc/inet/hosts`	host name database
`/etc/inet/netmasks`	network mask database
`/etc/nodename`	local source for system name
`/etc/.rootkey`	super-user's secret key
`/etc/shadow`	shadow password file
`/etc/sysidcfg`	system identification configuration file for diskless booting
`/etc/vfstab`	virtual file system table
`/var/nis/NIS_COLD_START`	list of NIS+ servers that serve a host's default domain
`/var/yp/binding/*/ypservers`	identifies NIS servers to which the client is allowed to bind

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWadmap

See Also `init(1M)`, `kdmconfig(1M)`, `sshd(1M)`, `sysidconfig(1M)`, `sysidtool(1M)`, `hosts(4)`, `netmasks(4)`, `shadow(4)`, `sysidcfg(4)`, `attributes(5)`

Notes `sys-unconfig` is not available on diskless clients.

Name tapes – creates /dev entries for tape drives attached to the system

Synopsis `/usr/sbin/tapes [-r root_dir]`

Description devfsadm(1M) is now the preferred command for `/dev` and `/devices` and should be used instead of `tapes`.

`tapes` creates symbolic links in the `/dev/rmt` directory to the actual tape device special files under the `/devices` directory tree. `tapes` searches the kernel device tree to see what tape devices are attached to the system. For each equipped tape drive, the following steps are performed:

1. The `/dev/rmt` directory is searched for a `/dev/rmt/`*n* entry that is a symbolic link to the `/devices` special node of the current tape drive. If one is found, this determines the logical controller number of the tape drive.
2. The rest of the special devices associated with the drive are checked, and incorrect symbolic links are removed and necessary ones added.
3. If none are found, a new logical controller number is assigned (the lowest-unused number), and new symbolic links are created for all the special devices associated with the drive.

`tapes` does not remove links to non-existent devices; these must be removed by hand.

`tapes` is run each time a reconfiguration-boot is performed, or when add_drv(1M) is executed.

Notice to Driver Writers tapes(1M) considers all devices with the node type `DDI_NT_TAPE` to be tape devices; these devices must have their minor name created with a specific format. The minor name encodes operational modes for the tape device and consists of an ASCII string of the form [`l`,`m`,`h`,`c`,`u`][`b`][`n`].

The first character set is used to specify the tape density of the device, and are named low (`l`), medium (`m`), high (`h`), compressed (`c`), and ultra (`u`). These specifiers only express a relative density; it is up to the driver to assign specific meanings as needed. For example, 9 track tape devices interpret these as actual bits-per-inch densities, where `l` means 800 BPI, `m` means 1600 BPI , and `h` means 6250 BPI, whereas 4mm DAT tapes defines `l` as standard format, and `m`, `h`, `c` and `u` as compressed format. Drivers may choose to implement any or all of these format types.

During normal tape operation (non-BSD behavior), once an EOF mark has been reached, subsequent reads from the tape device return an error. An explicit IOCTL must be issued to space over the EOF mark before the next file can be read. `b` instructs the device to observe BSD behavior, where reading at EOF will cause the tape device to automatically space over the EOF mark and begin reading from the next file.

`n` or no-rewind-on-close instructs the driver to not rewind to the beginning of tape when the device is closed. Normal behavior for tape devices is to reposition to BOT when closing. See mtio(7I).

The minor number for tape devices should be created by encoding the device's instance number using the tape macro `MTMINOR` and ORing in the proper combination of density, BSD behavior, and no-rewind flags. See `mtio(7I)`.

To prevent `tapes` from attempting to automatically generate links for a device, drivers must specify a private node type and refrain from using the node type string `DDI_NT_TAPE` when calling`ddi_create_minor_node(9F)`.

Options The following options are supported:

`-r` *root_dir* Causes `tapes` to presume that the `/dev/rmt` directory tree is found under *root_dir*, not directly under `/`.

Errors If `tapes` finds entries of a particular logical controller linked to different physical controllers, it prints an error message and exits without making any changes to the `/dev` directory, since it cannot determine which of the two alternative logical to physical mappings is correct. The links should be manually corrected or removed before another reconfiguration boot is performed.

Examples **EXAMPLE 1** Creating Tape Device Nodes From Within the Driver's `attach()` Function

This example demonstrates creating tape device nodes from within the `xktape` driver's `attach(9E)` function.

```
#include <sys/mtio.h>
struct tape_minor_info {
    char *minor_name;
    int   minor_mode;
};
/*
 * create all combinations of logical tapes
*/
static struct tape_minor_info example_tape[] = {
   {"",    0},                    /* default tape */
   {"l",   MT_DENSITY1},
   {"lb",  MT_DENSITY1 | MT_BSD},
   {"lbn", MT_DENSITY1 | MT_BSD | MT_NOREWIND},
   {"m",   MT_DENSITY2},
   {"mb",  MT_DENSITY2 | MT_BSD},
   {"mbn", MT_DENSITY2 | MT_BSD | MT_NOREWIND},
   {"h",   MT_DENSITY3},
   {"hb",  MT_DENSITY3 | MT_BSD},
   {"hbn", MT_DENSITY3 | MT_BSD | MT_NOREWIND},
   {"c",   MT_DENSITY4},
   {"cb",  MT_DENSITY4 | MT_BSD},
   {"cbn", MT_DENSITY4| MT_BSD | MT_NOREWIND},
   {NULL,  0},
};
```

EXAMPLE 1 Creating Tape Device Nodes From Within the Driver's `attach()` Function *(Continued)*

```
int
xktapeattach(dev_info_t *dip, ddi_attach_cmd_t cmd)
{
   int instance;
   struct tape_minor_info *mdp;
      /* other stuff in attach... */
   instance = ddi_get_instance(dip);

   for (mdp = example_tape; mdp->minor_name != NULL; mdp++) {
            ddi_create_minor_node(dip, mdp->minor_name, S_IFCHR,
                 (MTMINOR(instance) | mdp->minor_mode), DDI_NT_TAPE, 0);
 }
```

Installing the `xktape` driver on a Sun Fire 4800, with the driver controlling a SCSI tape (target 4 attached to an `isp(7D)` SCSI HBA) and performing a reconfiguration-boot creates the following special files in `/devices`.

```
# ls -l /devices/ssm@0,0/pci@18,700000/pci@1/SUNW,isptwo@4
crw-rw-rw-   1 root sys   33,136 Aug 29 00:02  xktape@4,0:
crw-rw-rw-   1 root sys   33,200 Aug 29 00:02  xktape@4,0:b
crw-rw-rw-   1 root sys   33,204 Aug 29 00:02  xktape@4,0:bn
crw-rw-rw-   1 root sys   33,152 Aug 29 00:02  xktape@4,0:c
crw-rw-rw-   1 root sys   33,216 Aug 29 00:02  xktape@4,0:cb
crw-rw-rw-   1 root sys   33,220 Aug 29 00:02  xktape@4,0:cbn
crw-rw-rw-   1 root sys   33,156 Aug 29 00:02  xktape@4,0:cn
crw-rw-rw-   1 root sys   33,144 Aug 29 00:02  xktape@4,0:h
crw-rw-rw-   1 root sys   33,208 Aug 29 00:02  xktape@4,0:hb
crw-rw-rw-   1 root sys   33,212 Aug 29 00:02  xktape@4,0:hbn
crw-rw-rw-   1 root sys   33,148 Aug 29 00:02  xktape@4,0:hn
crw-rw-rw-   1 root sys   33,128 Aug 29 00:02  xktape@4,0:l
crw-rw-rw-   1 root sys   33,192 Aug 29 00:02  xktape@4,0:lb
crw-rw-rw-   1 root sys   33,196 Aug 29 00:02  xktape@4,0:lbn
crw-rw-rw-   1 root sys   33,132 Aug 29 00:02  xktape@4,0:ln
crw-rw-rw-   1 root sys   33,136 Aug 29 00:02  xktape@4,0:m
crw-rw-rw-   1 root sys   33,200 Aug 29 00:02  xktape@4,0:mb
crw-rw-rw-   1 root sys   33,204 Aug 29 00:02  xktape@4,0:mbn
crw-rw-rw-   1 root sys   33,140 Aug 29 00:02  xktape@4,0:mn
crw-rw-rw-   1 root sys   33,140 Aug 29 00:02  xktape@4,0:n
```

`/dev/rmt` will contain the logical tape devices (symbolic links to tape devices in `/devices`).

```
# ls -l /dev/rmt
/dev/rmt/0    -> ../../devices/[....]/xktape@4,0:
/dev/rmt/0b   -> ../../devices/[....]/xktape@4,0:b
/dev/rmt/0bn  -> ../../devices/[....]/xktape@4,0:bn
```

EXAMPLE 1 Creating Tape Device Nodes From Within the Driver's `attach()` Function *(Continued)*

```
/dev/rmt/0c   -> ../../devices/[....]/xktape@4,0:c
/dev/rmt/0cb  -> ../../devices/[....]/xktape@4,0:cb
/dev/rmt/0cbn -> ../../devices/[....]/xktape@4,0:cbn
/dev/rmt/0cn  -> ../../devices/[....]/xktape@4,0:cn
/dev/rmt/0h   -> ../../devices/[....]/xktape@4,0:h
/dev/rmt/0hb  -> ../../devices/[....]/xktape@4,0:hb
/dev/rmt/0hbn -> ../../devices/[....]/xktape@4,0:hbn
/dev/rmt/0hn  -> ../../devices/[....]/xktape@4,0:hn
/dev/rmt/0l   -> ../../devices/[....]/xktape@4,0:l
/dev/rmt/0lb  -> ../../devices/[....]/xktape@4,0:lb
/dev/rmt/0lbn -> ../../devices/[....]/xktape@4,0:lbn
/dev/rmt/0ln  -> ../../devices/[....]/xktape@4,0:ln
/dev/rmt/0m   -> ../../devices/[....]/xktape@4,0:m
/dev/rmt/0mb  -> ../../devices/[....]/xktape@4,0:mb
/dev/rmt/0mbn -> ../../devices/[....]/xktape@4,0:mbn
/dev/rmt/0mn  -> ../../devices/[....]/xktape@4,0:mn
/dev/rmt/0n   -> ../../devices/[....]/xktape@4,0:n
```

Files

`/dev/rmt/*` logical tape devices

`/devices/*` tape device nodes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `add_drv(1M)`, `devfsadm(1M)`, `attributes(5)`, `isp(7D)`, `devfs(7FS)`, `mtio(7I)`, `attach(9E)`, `ddi_create_minor_node(9F)`

Writing Device Drivers

Bugs `tapes` silently ignores malformed minor device names.

Name taskstat – prints ASET tasks status

Synopsis

```
/usr/aset/util/taskstat  [-d aset_dir]
```

Description `taskstat` is located in the `/usr/aset/util` directory. `/usr/aset` is the default operating directory of the Automated Security Enhancement Tool (ASET). An alternative working directory can be specified by the administrators through the `aset -d` command or the `ASETDIR` environment variable. See `aset`(1M). Because `aset` dispatches its tasks to run in the background, when it returns, these tasks may or may not have completed. `taskstat` prints the status of the tasks, listing those that are completed and those that are still executing.

The ASET reports, which are located in the `/usr/aset/reports` directory (see the `-d` option), are not complete until all the tasks finish executing.

Options

`-d` *aset_dir* Specify the working directory for ASET. By default, this directory is `/usr/aset`. With this option, the `reports` directory will be located under *aset_dir*.

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWast

See Also `aset`(1M), `attributes`(5)

System Administration Guide: Basic Administration

Name tcxconfig – configure default linearity of 24-bit TrueColor Visual for OpenWindows

Synopsis

```
/usr/sbin/tcxconfig [linear | nonlinear]
```

Description The `tcxconfig` script changes the default linearity of a 24-bit TrueColor Visual for OpenWindows on a system with an S24 frame buffer. When the S24 graphics driver for OpenWindows is installed, the default 24-bit TrueColor Visual is nonlinear. You can run `tcxconfig` with an argument that specifies the setting you want.

OpenWindows should not be running when you execute the `tcxconfig` script with an option. Start OpenWindows after `tcxconfig` has set the linearity you desire.

Options If you specify no option, `tcxconfig` displays the current default setting.

You must become superuser before you can execute `tcxconfig` with one of the following options.

`linear` Set linear visual to be the default 24-bit TrueColor Visual. This means colors will be gamma-corrected.

`nonlinear` Set nonlinear visual to be the default 24-bit TrueColor Visual.

Exit Status The following exit values are returned:

`0` success

`1` an error has occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtcxow

See Also `attributes(5)`

REFERENCE

System Administration Commands - Part 3

Name th_define – create fault injection test harness error specifications

Synopsis

```
th_define [-n name -i instance| -P path] [-a acc_types]
     [-r reg_number] [-l offset [length]]
     [-c count [failcount]] [-o operator [operand]]
     [-f acc_chk] [-w max_wait_period [report_interval]]

or

th_define [-n name -i instance| -P path]
     [-a log [acc_types] [-r reg_number] [-l offset [length]]]
     [-c count [failcount]] [-s collect_time] [-p policy]
     [-x flags] [-C comment_string]
     [-e fixup_script [args]]

or

th_define [-h]
```

Description The `th_define` utility provides an interface to the `bus_ops` fault injection `bofi` device driver for defining error injection specifications (referred to as errdefs). An errdef corresponds to a specification of how to corrupt a device driver's accesses to its hardware. The command line arguments determine the precise nature of the fault to be injected. If the supplied arguments define a consistent errdef, the `th_define` process will store the errdef with the `bofi` driver and suspend itself until the criteria given by the errdef become satisfied (in practice, this will occur when the access counts go to zero).

You use the `th_manage(1M)` command with the `start` option to activate the resulting errdef. The effect of `th_manage` with the `start` option is that the `bofi` driver acts upon the errdef by matching the number of hardware accesses—specified in *count*, that are of the type specified in *acc_types*, made by instance number *instance*—of the driver whose name is *name*, (or by the driver instance specified by *path*) to the register set (or DMA handle) specified by *reg_number*, that lie within the range *offset* to *offset* + *length* from the beginning of the register set or DMA handle. It then applies *operator* and *operand* to the next *failcount* matching accesses.

If *acc_types* includes `log`, `th_define` runs in automatic test script generation mode, and a set of test scripts (written in the Korn shell) is created and placed in a sub-directory of the current directory with the name <*driver*>`.test.`<*id*> (for example, `glm.test.978177106`). A separate, executable script is generated for each access handle that matches the logging criteria. The log of accesses is placed at the top of each script as a record of the session. If the current directory is not writable, file output is written to standard output. The base name of each test file is the driver name, and the extension is a number that discriminates between different access handles. A control script (with the same name as the created test directory) is generated that will run all the test scripts sequentially.

Executing the scripts will install, and then activate, the resulting error definitions. Error definitions are activated sequentially and the driver instance under test is taken offline and brought back online before each test (refer to the `-e` option for more information). By default, logging applies to all `PIO` accesses, all interrupts, and all DMA accesses to and from areas

mapped for both reading and writing. You can constrain logging by specifying additional *acc_types*, *reg_number*, *offset* and *length*. Logging will continue for *count* matching accesses, with an optional time limit of *collect_time* seconds.

Either the `-n` or `-P` option must be provided. The other options are optional. If an option (other than `-a`) is specified multiple times, only the final value for the option is used. If an option is not specified, its associated value is set to an appropriate default, which will provide maximal error coverage as described below.

Options The following options are available:

`-n` *name*
: Specify the name of the driver to test. (String)

`-i` *instance*
: Test only the specified driver instance (-1 matches all instances of driver). (Numeric)

`-P` *path*
: Specify the full device path of the driver to test. (String)

`-r` *reg_number*
: Test only the given register set or DMA handle (-1 matches all register sets and DMA handles). (Numeric)

`-a` *acc_types*
: Only the specified access types will be matched. Valid values for the *acc_types* argument are `log`, `pio`, `pio_r`, `pio_w`, `dma`, `dma_r`, `dma_w` and `intr`. Multiple access types, separated by spaces, can be specified. The default is to match all hardware accesses.

  If *acc_types* is set to `log`, logging will match all `PIO` accesses, interrupts and DMA accesses to and from areas mapped for both reading and writing. `log` can be combined with other *acc_types*, in which case the matching condition for logging will be restricted to the specified addional *acc_types*. Note that `dma_r` will match only DMA handles mapped for reading only; `dma_w` will match only DMA handles mapped for writing only; `dma` will match only DMA handles mapped for both reading and writing.

`-l` *offset* [*length*]
: Constrain the range of qualifying accesses. The *offset* and *length* arguments indicate that any access of the type specified with the `-a` option, to the register set or DMA handle specified with the `-r` option, lie at least *offset* bytes into the register set or DMA handle and at most *offset* + *length*

bytes into it. The default for *offset* is 0. The default for *length* is the maximum value that can be placed in an `offset_t` C data type (see `types.h`). Negative values are converted into unsigned quantities. Thus, `th_define -l 0 -1` is maximal.

`-c` *count* [*failcount*]

Wait for *count* number of matching accesses, then apply an operator and operand (see the `-o` option) to the next *failcount* number of matching accesses. If the access type (see the `-a` option) includes logging, the number of logged accesses is given by *count* + *failcount* - 1. The -1 is required because the last access coincides with the first faulting access.

Note that access logging may be combined with error injection if *failcount* and *operator* are nonzero and if the access type includes logging and any of the other access types (`pio`, `dma` and `intr`) See the description of access types in the definition of the `-a` option, above.

When the *count* and *failcount* fields reach zero, the status of the errdef is reported to standard output. When all active errdefs created by the `th_define` process complete, the process exits. If *acc_types* includes `log`, *count* determines how many accesses to log. If *count* is not specified, a default value is used. If *failcount* is set in this mode, it will simply increase the number of accesses logged by a further *failcount* - 1.

`-o` *operator* [*operand*]

For qualifying PIO read and write accesses, the value read from or written to the hardware is corrupted according to the value of *operator*:

`EQ` *operand* is returned to the driver.

`OR` *operand* is bitwise ORed with the real value.

`AND` *operand* is bitwise ANDed with the real value.

`XOR` *operand* is bitwise XORed with the real value.

For PIO write accesses, the following operator is allowed:

`NO` Simply ignore the driver's attempt to write to the hardware.

Note that a driver performs PIO via the `ddi_get`*X*`()`, `ddi_put`*X*`()`, `ddi_rep_get`*X*`()` and `ddi_rep_put`*X*`()` routines (where *X* is 8, 16, 32 or 64). Accesses made using `ddi_get`*X*`()` and `ddi_put`*X*`()` are treated as a single access, whereas an access made using the `ddi_rep_*`(9F) routines are broken down into their respective number of accesses, as given by the *repcount* parameter to these DDI calls. If the access is performed via a DMA handle, *operator* and *value* are applied to every access that comprises the DMA request. If interference with interrupts has been requested then the operator may take any of the following values:

`DELAY` After *count* accesses (see the `-c` option), delay delivery of the next *failcount* number of interrupts for *operand* number of microseconds.

`LOSE` After *count* number of interrupts, fail to deliver the next *failcount* number of real interrupts to the driver.

`EXTRA` After *count* number of interrupts, start delivering *operand* number of extra interrupts for the next *failcount* number of real interrupts.

The default value for *operand* and *operator* is to corrupt the data access by flipping each bit (XOR with -1).

`-f` *acc_chk*

If the *acc_chk* parameter is set to 1 or `pio`, then the driver's calls to `ddi_check_acc_handle(9F)` return `DDI_FAILURE` when the access count goes to 1. If the *acc_chk* parameter is set to 2 or `dma`, then the driver's calls to `ddi_check_dma_handle(9F)` return `DDI_FAILURE` when the access count goes to 1.

`-w` *max_wait_period* [*report_interval*]

Constrain the period for which an error definition will remain active. The option applies only to non-logging errdefs. If an error definition remains active for *max_wait_period* seconds, the test will be aborted. If *report_interval* is set to a nonzero value, the current status of the error definition is reported to standard output every *report_interval* seconds. The default value is zero. The status of the errdef is reported in parsable format (eight fields, each separated by a colon (`:`) character, the last of which is a string enclosed by double quotes and the remaining seven fields are integers):

ft:*mt*:*ac*:*fc*:*chk*:*ec*:*s*:*"message"* which are defined as follows:

ft	The UTC time when the fault was injected.
mt	The UTC time when the driver reported the fault.
ac	The number of remaining non-faulting accesses.
fc	The number of remaining faulting accesses.
chk	The value of the *acc_chk* field of the errdef.
ec	The number of fault reports issued by the driver against this errdef (*mt* holds the time of the initial report).
s	The severity level reported by the driver.
"message"	Textual reason why the driver has reported a fault.

`-h`

Display the command usage string.

`-s` *collect_time*

If *acc_types* is given with the `-a` option and includes `log`, the errdef will log accesses for *collect_time* seconds (the default is to log until the log becomes full). Note that, if the errdef

specification matches multiple driver handles, multiple logging errdefs are registered with the `bofi` driver and logging terminates when all logs become full or when *collect_time* expires or when the associated errdefs are cleared. The current state of the log can be checked with the `th_manage(1M)` command, using the `broadcast` parameter. A log can be terminated by running `th_manage(1M)` with the `clear_errdefs` option or by sending a `SIGALRM` signal to the `th_define` process. See `alarm(2)` for the semantics of `SIGALRM`.

`-p` *policy*

Applicable when the *acc_types* option includes `log`. The parameter modifies the policy used for converting from logged accesses to errdefs. All policies are inclusive:

- Use `rare` to bias error definitions toward rare accesses (default).
- Use `operator` to produce a separate error definition for each operator type (default).
- Use `common` to bias error definitions toward common accesses.
- Use `median` to bias error definitions toward median accesses.
- Use `maximal` to produce multiple error definitions for duplicate accesses.
- Use `unbiased` to create unbiased error definitions.
- Use `onebyte`, `twobyte`, `fourbyte`, or `eightbyte` to select errdefs corresponding to 1, 2, 4 or 8 byte accesses (if chosen, the `-xr` option is enforced in order to ensure that `ddi_rep_*()` calls are decomposed into *multiple single accesses*).
- Use `multibyte` to create error definitions for multibyte accesses performed using `ddi_rep_get*()` and `ddi_rep_put*()`.

Policies can be combined by adding together these options. See the NOTES section for further information.

-x *flags*

Applicable when the *acc_types* option includes `log`. The *flags* parameter modifies the way in which the `bofi` driver logs accesses. It is specified as a string containing any combination of the following letters:

`w` Continuous logging (that is, the log will wrap when full).

`t` Timestamp each log entry (access times are in seconds).

`r` Log repeated I/O as individual accesses (for example, a `ddi_rep_get16(9F)` call which has a repcount of *N* is logged *N* times with each transaction logged as size 2 bytes. Without this option, the default logging behavior is to log this access once only, with a transaction size of twice the *repcount*).

-C *comment_string*

Applicable when the *acc_types* option includes `log`. It provides a comment string to be placed in any generated test scripts. The string must be enclosed in double quotes.

-e *fixup_script* [*args*]

Applicable when the *acc_types* option includes `log`. The output of a logging errdefs is to generate a test script for each driver access handle. Use this option to embed a command in the resulting script before the errors are injected. The generated test scripts will take an instance offline and bring it back online before injecting errors in order to bring the instance into a known fault–free state. The executable *fixup_script* will be called twice with the set of optional *args*— once just before the instance is taken offline and again after the instance has been brought online. The following variables are passed into the environment of the called executable:

`DRIVER_PATH` Identifies the device path of the instance.

`DRIVER_INSTANCE` Identifies the instance number of the device.

`DRIVER_UNCONFIGURE`	Has the value 1 when the instance is about to be taken offline.
`DRIVER_CONFIGURE`	Has the value 1 when the instance has just been brought online.

Typically, the executable ensures that the device under test is in a suitable state to be taken offline (unconfigured) or in a suitable state for error injection (for example configured, error free and servicing a workload). A minimal script for a network driver could be:

```
#!/bin/ksh

driver=xyznetdriver
ifnum=$driver$DRIVER_INSTANCE

if [[ $DRIVER_CONFIGURE = 1 ]]; then
    ifconfig $ifnum plumb
    ifconfig $ifnum ...
    ifworkload start $ifnum
elif [[ $DRIVER_UNCONFIGURE = 1 ]]; then
    ifworkload stop $ifnum
    ifconfig $ifnum down
    ifconfig $ifnum unplumb
fi
exit $?
```

The `-e` option must be the last option on the command line.

If the `-a log` option is selected but the `-e` option is not given, a default script is used. This script repeatedly attempts to detach and then re-attach the device instance under test.

Examples

Examples of Error Definitions

```
th_define -n foo -i 1 -a log
```

Logs all accesses to all handles used by instance 1 of the `foo` driver while running the default workload (attaching and detaching the instance). Then generates a set of test scripts to inject appropriate errdefs while running that default workload.

```
th_define -n foo -i 1 -a log pio
```

Logs PIO accesses to each PIO handle used by instance 1 of the `foo` driver while running the default workload (attaching and detaching the instance). Then generates a set of test scripts to inject appropriate errdefs while running that default workload.

```
th_define -n foo -i 1 -p onebyte median -e fixup arg -now
```

Logs all accesses to all handles used by instance 1 of the `foo` driver while running the workload defined in the fixup script `fixup` with arguments `arg` and `-now`. Then generates a set of test scripts to inject appropriate errdefs while running that workload. The resulting error definitions are requested to focus upon single byte accesses to locations that are accessed a *median* number of times with respect to frequency of access to I/O addresses.

```
th_define -n se -l 0x20 1 -a pio_r -o OR 0x4 -c 10 1000
```

Simulates a stuck serial chip command by forcing 1000 consecutive read accesses made by any instance of the `se` driver to its command status register, thereby returning status busy.

```
th_define -n foo -i 3 -r 1 -a pio_r -c 0 1 -f 1 -o OR 0x100
```

Causes 0x100 to be ORed into the next physical I/O read access from any register in register set 1 of instance 3 of the `foo` driver. Subsequent calls in the driver to `ddi_check_acc_handle()` return `DDI_FAILURE`.

```
th_define -n foo -i 3 -r 1 -a pio_r -c 0 1 -o OR 0x0
```

Causes 0x0 to be ORed into the next physical I/O read access from any register in register set 1 of instance 3 of the `foo` driver. This is of course a no-op.

```
th_define -n foo -i 3 -r 1 -l 0x8100 1 -a pio_r -c 0 10 -o EQ 0x70003
```

Causes the next ten next physical I/O reads from the register at offset 0x8100 in register set 1 of instance 3 of the `foo` driver to return 0x70003.

```
th_define -n foo -i 3 -r 1 -l 0x8100 1 -a pio_w -c 100 3 -o AND
0xfffffffffffefff
```

The next 100 physical I/O writes to the register at offset 0x8100 in register set 1 of instance 3 of the `foo` driver take place as normal. However, on each of the three subsequent accesses, the 0x1000 bit will be cleared.

```
th_define -n foo -i 3 -r 1 -l 0x8100 0x10 -a pio_r -c 0 1 -f 1 -o XOR 7
```

Causes the bottom three bits to have their values toggled for the next physical I/O read access to registers with offsets in the range 0x8100 to 0x8110 in register set 1 of instance 3 of the `foo` driver. Subsequent calls in the driver to `ddi_check_acc_handle()` return `DDI_FAILURE`.

```
th_define -n foo -i 3 -a pio_w -c 0 1 -o NO 0
```

Prevents the next physical I/O write access to any register in any register set of instance 3 of the `foo` driver from going out on the bus.

```
th_define -n foo -i 3 -l 0 8192 -a dma_r -c 0 1 -o OR 7
```

Causes 0x7 to be ORed into each `long long` in the first 8192 bytes of the next DMA read, using any DMA handle for instance 3 of the `foo` driver.

```
th_define -n foo -i 3 -r 2 -l 0 8 -a dma_r -c 0 1 -o OR 0x7070707070707070
```

Causes 0x70 to be ORed into each byte of the first `long long` of the next DMA read, using the DMA handle with sequential allocation number 2 for instance 3 of the `foo` driver.

```
th_define -n foo -i 3 -l 256 256 -a dma_w -c 0 1 -f 2 -o OR 7
```

Causes 0x7 to be ORed into each `long long` in the range from offset 256 to offset 512 of the next DMA write, using any DMA handle for instance 3 of the `foo` driver. Subsequent calls in the driver to `ddi_check_dma_handle()` return `DDI_FAILURE`.

```
th_define -n foo -i 3 -r 0 -l 0 8 -a dma_w -c 100 3 -o AND 0xfffffffffffefff
```

The next 100 DMA writes using the DMA handle with sequential allocation number 0 for instance 3 of the `foo` driver take place as normal. However, on each of the three subsequent accesses, the 0x1000 bit will be cleared in the first `long long` of the transfer.

```
th_define -n foo -i 3 -a intr -c 0 6 -o LOSE 0
```

Causes the next six interrupts for instance 3 of the `foo` driver to be lost.

```
th_define -n foo -i 3 -a intr -c 30 1 -o EXTRA 10
```

When the thirty-first subsequent interrupt for instance 3 of the `foo` driver occurs, a further ten interrupts are also generated.

```
th_define -n foo -i 3 -a intr -c 0 1 -o DELAY 1024
```

Causes the next interrupt for instance 3 of the `foo` driver to be delayed by 1024 microseconds.

Notes The policy option in the `th_define -p` syntax determines how a set of logged accesses will be converted into the set of error definitions. Each logged access will be matched against the chosen policies to determine whether an error definition should be created based on the access.

Any number of policy options can be combined to modify the generated error definitions.

Bytewise Policies These select particular I/O transfer sizes. Specifing a byte policy will exclude other byte policies that have not been chosen. If none of the byte type policies is selected, all transfer sizes are treated equally. Otherwise, only those specified transfer sizes will be selected.

`onebyte`	Create errdefs for one byte accesses (`ddi_get8()`)
`twobyte`	Create errdefs for two byte accesses (`ddi_get16()`)
`fourbyte`	Create errdefs for four byte accesses (`ddi_get32()`)
`eightbyte`	Create errdefs for eight byte accesses (`ddi_get64()`)
`multibyte`	Create errdefs for repeated byte accesses (`ddi_rep_get*()`)

Frequency of Access Policies

The frequency of access to a location is determined according to the access type, location and transfer size (for example, a two-byte read access to address A is considered distinct from a four-byte read access to address A). The algorithm is to count the number of accesses (of a given type and size) to a given location, and find the locations that were most and least accessed (let *maxa* and *mina* be the number of times these locations were accessed, and *mean* the total number of accesses divided by total number of locations that were accessed). Then a rare access is a location that was accessed less than

(mean - mina) / 3 + mina

times. Similarly for the definition of common accesses:

maxa - (maxa - mean) / 3

A location whose access patterns lies within these cutoffs is regarded as a location that is accessed with median frequency.

`rare`	Create errdefs for locations that are rarely accessed.
`common`	Create errdefs for locations that are commonly accessed.
`median`	Create errdefs for locations that are accessed a median frequency.

Policies for Minimizing errdefs

If a transaction is duplicated, either a single or multiple errdefs will be written to the test scripts, depending upon the following two policies:

`maximal`	Create multiple errdefs for locations that are repeatedly accessed.
`unbiased`	Create a single errdef for locations that are repeatedly accessed.
`operators`	For each location, a default operator and operand is typically applied. For maximal test coverage, this default may be modified using the `operators` policy so that a separate errdef is created for each of the possible corruption operators.

See Also `kill(1)`, `th_manage(1M)`, `alarm(2)`, `ddi_check_acc_handle(9F)`, `ddi_check_dma_handle(9F)`

Name th_manage – manage the fault injection test harness

Synopsis

```
th_manage name instance command

th_manage path command
```

Description `th_manage` applies the action specified by *command* to the instance specified by *instance* of the driver specified by *name* (or the driver instance specified by *path*). The driver instance must be running fault injection specifications (errdefs) defined by `th_define(1M)`.

`th_manage` supports several commands that operate on the driver instance specified by *name* and *instance* (or *path*). The commands are:

`broadcast`	Awaken all `th_define` processes, causing them to display their current status and exit if the errdef is now defunct (that is, if *count*, *failcount* , and *acc_chk* are all zero).
`clear_acc_chk`	Awaken all `th_define` processes. If *count* and *failcount* are already zero, then set *acc_chk* to zero, so that `th_define` exits once it has displayed its status.
`clear_errdefs`	Awaken all `th_define processes`. *count*, *failcount* and *acc_chk* are all set to zero so that all `th_define` commands exit once they have displayed their status.
`clear_errors`	Awaken all `th_define` processes. If *count* is already zero, set *failcount* and *acc_chk* to zero, so that `th_define` exits once it has displayed its status.
`get_handles`	List all the access handles.
`start`	Begin or resume execution of all errdefs.
`stop`	Suspend all errdefs for this *name* and *instance* (or *path).*

Examples **EXAMPLE 1** Useful Commands

To begin the tests, enter:

```
# th_manage foo 0 start
```

To check the status of the errdefs, enter:

```
# th_manage foo 0 broadcast
```

This causes each `th_define` process to print out its current status.

If the driver has reported a fatal error, you can take the driver offline using `libdevice`, clear the error condition by entering:

```
# th_manage foo 0 clear_acc_chk
```

EXAMPLE 1 Useful Commands *(Continued)*

or

```
# th_manage foo 0 clear_errors
```

and bring the driver online again using `libdevice`.

To terminate testing, enter:

```
# th_manage foo 0 clear_errdefs
```

See Also `th_define(1M)`

Name tic – terminfo compiler

Synopsis `tic [-v [n]] [-c] file`

Description The command `tic` translates a `terminfo` file from the source format into the compiled format. The results are placed in the directory `/usr/share/lib/terminfo`. The compiled format is necessary for use with the library routines in `curses(3CURSES)`.

If the environment variable `TERMINFO` is set, the compiled results are placed there instead of `/usr/share/lib/terminfo`.

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes. Terminal names exceeding 14 characters will be truncated to 14 characters and a warning message will be printed.

Options The following options are supported:

`-c` Specifies to check only `file` for errors. Errors in `use=` links are not detected.

`-v`[*n*] Specify that (verbose) output be written to standard error trace information showing `tic`'s progress. The optional integer *n* is a number from 1 to 10, indicating the desired level of detail of information. If *n* is omitted, the default level is 1. If *n* is specified and greater than 1, the level of detail is increased.

Operands `file` Contains one or more `terminfo` terminal descriptions in source format [see `terminfo(4)`]. Each description in the file describes the capabilities of a particular terminal. When a `use=`*entry-name* field is discovered in a terminal entry currently being compiled, `tic` reads in the binary from `/usr/share/lib/terminfo` to complete the entry. (Entries created from `file` will be used first. If the environment variable `TERMINFO` is set, that directory is searched instead of `/usr/share/lib/terminfo`.) `tic` duplicates the capabilities in *entry-name* for the current entry, with the exception of those capabilities that are explicitly defined in the current entry.

Files `/usr/share/lib/terminfo/?/*` Compiled terminal description database

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `captoinfo(1M)`, `infocmp(1M)`, `curses(3CURSES)`, `terminfo(4)`, `attributes(5)`

Notes When an entry, for example, `entry_name_1`, contains a `use=`*entry_name_2* field, any canceled capabilities in *entry_name_2* must also appear in `entry_name_1` before `use=` for these capabilities to be canceled in `entry_name_1`.

Name tnchkdb – check file syntax of trusted network databases

Synopsis

```
/usr/sbin/tnchkdb  [-h path] [-t path] [-z path]
```

Description `tnchkdb` checks the syntax of the `tnrhtp`, `tnrhdb`, and `tnzonecfg` databases. By default, the *path* for each file is:

- `/etc/security/tsol/tnrhtp`
- `/etc/security/tsol/tnrhdb`
- `/etc/security/tsol/tnzonecfg`

You can specify an alternate path for any or all of the files by specifying that path on the command line by using the `-h` (`tnrhdb`), `-t` (`tnrhtp`) and `-z` (`tnzonecfg`) options. The options are useful when testing a set of modified files before installing the files as new system databases.

All three database files are checked for integrity. `tnchkdb` returns an exit status of `0` if all of the files are syntactically and, to the extent possible, semantically correct. If one or more files have errors, then an exit status of `1` is returned. If there are command line problems, such as an unreadable file, an exit status of `2` is returned. Errors are written to standard error.

To avoid cascading errors, when there are errors in `tnrhtp`, the template names in `tnrhdb` are not validated.

`tnchkdb` can be run at any label, but the standard `/etc/security/tsol` files are visible only in the global zone.

Options

`-h` [*path*] Check *path* for proper `tnrhdb` syntax. If *path* is not specified, then check `/etc/security/tsol/tnrhdb`.

`-t` [*path*] Check *path* for proper `tnrhtp` syntax. If *path* is not specified, then check `/etc/security/tsol/tnrhtp`.

`-z` [*path*] Check *path* for proper `tnzonecfg` syntax. If *path* is not specified, then check `/etc/security/tsol/tnzonecfg`.

Examples **EXAMPLE 1** Sample Error Message

The `tnchkdb` command checks for CIPSO errors. In this example, the `admin_low` template has an incorrect value of `ADMIN_HIGH` for its default label.

```
# tnchkdb
checking /etc/security/tsol/tnrhtp ...
tnchkdb: def_label classification 7fff is invalid for cipso labels:
line 14 entry admin_low
tnchkdb: def_label compartments 241-256 must be zero for cipso labels:
line 14 entry admin_low
checking /etc/security/tsol/tnrhdb ...
checking /etc/security/tsol/tnzonecfg ...
```

Files

`/etc/security/tsol/tnrhdb`	Trusted network remote-host database
`/etc/security/tsol/tnrhtp`	Trusted network remote-host templates
`/etc/security/tsol/tnzonecfg`	Trusted zone configuration database

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	See below.

The command line is Committed. The output is Uncommitted.

See Also `tnd(1M)`, `tnctl(1M)`, `attributes(5)`

"How to Check the Syntax of Trusted Network Databases" in *Oracle Solaris Trusted Extensions Administrator's Procedures*

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

It is possible to have inconsistent but valid configurations of `tnrhtp` and `tnrhdb` when LDAP is used to supply missing templates.

Name tnctl – configure Trusted Extensions network parameters

Synopsis

```
/usr/sbin/tnctl [-dfv] [-h host [/prefix] [:template]]
     [-m zone:mlp:shared-mlp][-t template [:key=val [;key=val]]]
     [-HTz] file]
```

Description `tnctl` provides an interface to manipulate trusted network parameters in the Solaris kernel.

As part of Solaris Trusted Extensions initialization, `tnctl` is run in the global zone by an `smf(5)` script during system boot. The `tnctl` command is not intended to be used during normal system administration. Instead, if a local trusted networking database file is modified without using the Solaris Management Console, the administrator first issues `tnchkdb(1M)` to check the syntax, and then refreshes the kernel copy with this command:

```
# svcadm restart svc:/network/tnctl
```

See `WARNINGS` about the risks of changing remote host and template information on a running system.

Options

`-d`
Delete matching entries from the kernel. The default is to add new entries.

When deleting MLPs, the MLP range must match exactly. MLPs are specified in the form:

port`[-`*port*`]/`*protocol*

Where *port* can be a number in the range 1 to 65535. or any known service (see `services(4)`), and protocol can be a number in the range 1 to 255, or any known protocol (see `protocols(4)`).

`-f`
Flush all kernel entries before loading the entries that are specified on the command line. The flush does not take place unless at least one entry parsed successfully.

`-v`
Turn on verbose mode.

`-h` *host*[/*prefix*][:*template*]
Update the kernel remote-host cache on the local host for the specified *host* or, if a template name is given, change the kernel's cache to use the specified *template*. If *prefix* is not specified, then an implied prefix length is determined according to the rules used for interpreting the `tnrhdb`. If `-d` is specified, then a template name cannot be specified.

`-m` *zone*:*mlp*:*shared-mlp*
Modify the kernel's multilevel port (MLP) configuration cache for the specified *zone*. *zone* specifies the zone to be updated. *mlp* and *shared-mlp* specify the MLPs for the zone-specific and shared IP addresses. The *shared-mlp* field is effective in the global zone only.

`-t` *template*[*key=val*[;*key=val*]]
Update the kernel template cache for *template* or, if a list of *key=val* pairs is given, change the kernel's cache to use the specified entry. If `-d` is specified, then *key=val* pairs cannot be specified.

`-T` *file*
Load all template entries in *file* into the kernel cache.

`-H` *file*
Load all remote host entries in *file* into the kernel cache.

`-z` *file*
Load just the global zone's MLPs from *file* into the kernel cache. To reload MLPs for a non-global zone, reboot the zone:

```
# zoneadm -z non-global zone reboot
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	Uncommitted

Files

`/etc/security/tsol/tnrhdb`	Trusted network remote-host database
`/etc/security/tsol/tnrhtp`	Trusted network remote-host templates
`/etc/security/tsol/tnzonecfg`	Trusted zone configuration database
`/etc/nsswitch.conf`	Configuration file for the name service switch

See Also

`svcs(1)`, `svcadm(1M)`, `tninfo(1M)`, `tnd(1M)`, `tnchkdb(1M)`, `zoneadm(1M)`, `nsswitch.conf(4)`, `protocols(4)`, `services(4)`, `attributes(5)`, `smf(5)`

"How to Synchronize the Kernel Cache With Trusted Network Databases" in *Oracle Solaris Trusted Extensions Administrator's Procedures*

Warnings

Changing a template while the network is up can change the security view of an undetermined number of hosts.

Notes

The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

The `tnctl` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/tnctl
```

The service's status can be queried by using svcs(1). Administrative actions on this service, such as refreshing the kernel cache, can be performed using svcadm(1M), as in:

```
svcadm refresh svc:/network/tnctl
```

Name tnd – trusted network daemon

Synopsis `/usr/sbin/tnd [-p` *poll-interval*`]`

Description The `tnd` (trusted network daemon) initializes the kernel with trusted network databases and also reloads the databases on demand from an LDAP server and local files. `tnd` follows the order specified in the `nsswitch.conf(4)` file when loading configuration databases. `tnd` is started at the beginning of the boot process.

`tnd` loads two databases into the kernel: the remote host database, `tnrhdb` and the remote-host template database, `tnrhtp`. These databases and their effect on the trusted network are described in their respective man pages. When the associated LDAP database or local databases are changed, `tnd` also updates the local kernel cache at the predetermined interval.

If a local trusted networking database file is modified, the administrator should run `tnchkdb(1M)` to check the syntax, and should also run `svcadm refresh svc:/network/tnd` to initiate an immediate database scan by `tnd`.

`tnd` is intended to be started from an `smf(5)` script and to run in the global zone. The following signals cause specific `svcadm` actions:

`SIGHUP` Causes `svcadm refresh svc:/network/tnd` to be run.

Initiates a rescan of the local and LDAP `tnrhdb` and `tnrhtp` databases. `tnd` updates the kernel database with any changes found.

`SIGTERM` Causes `svcadm disable svc:/network/tnd` to be run.

Terminates the `tnd` daemon. No changes are made to the kernel database.

Running `tnd` in debug mode is determined by the value of the following service management facility (SMF) property:

```
tnd/debug_level = 0
```

A value of `0`, as above, prevents debug information from being collected; `1` turns on debugging. The default value is `0`. Debug output is sent to the `/var/tsol/tndlog` log file.

Options `-p` *poll-interval* Set poll interval to *poll-interval* seconds. The default *poll-interval* is 1800 seconds (30 minutes).

Examples **EXAMPLE 1** Changing the Poll Interval

The following command changes the polling interval to one hour, and puts this interval in the SMF repository. At the next boot, the `tnd` poll interval will be one hour.

```
# svccfg -s network/tnd setprop tnd/poll_interval=3600
```

The following command changes the polling interval, but does not update the repository. At the next boot, the `tnd` poll interval remains the default, 30 minutes.

EXAMPLE 1 Changing the Poll Interval *(Continued)*

```
# tnd -p 3600
```

Files

`/etc/security/tsol/tnrhdb`	Trusted network remote-host database
`/etc/security/tsol/tnrhtp`	Trusted network remote-host templates
`/etc/security/tsol/tnzonecfg`	Trusted zone configuration database
`/etc/nsswitch.conf`	Configuration file for the name service switch

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	See below.

The command invocation is Committed. The service is Private.

See Also `svcs(1)`, `svcadm(1M)`, `tninfo(1M)`, `tnctl(1M)`, `tnchkdb(1M)`, `nsswitch.conf(4)`, `attributes(5)`, `smf(5)`

"How to Synchronize the Kernel Cache With Trusted Network Databases" in *Oracle Solaris Trusted Extensions Administrator's Procedures*

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

The `tnd` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/tnd
```

The service's status can be queried by using `svcs(1)`. Administrative actions on this service, such as requests to restart the daemon, can be performed using `svcadm(1M)`, as in:

```
svcadm restart svc:/network/tnd
```

Name tninfo – print kernel-level network information and statistics

Synopsis `/usr/sbin/tninfo [-h` *hostname*`] [-m` *zone-name*`] [-t` *template*`]`

Description `tninfo` provides an interface to retrieve and display kernel-level network information and statistics.

Options

`-h` *hostname*	Display the security structure for the specified host in the remote-host cache. The output should reflect what is specified in the `tnrhdb` database.
`-m` *zone-name*	Display the MLP configuration associated with the specified zone. The output should reflect what is specified in the `tnzonecfg` database.
`-t` *template*	Display the structure associated with the specified *template*. The output should reflect what is specified in the `tnrhtp` database.

Examples

EXAMPLE 1 Displaying Remote Host Structures Cached in the Kernel

This example shows the remote host structures cached in the kernel. The output reflects the definition in the `tnrhdb` database.

```
# tninfo -h machine1
   IP address= 192.168.8.61
   Template = cipso
```

EXAMPLE 2 Displaying Multilevel Ports for the Global Zone

This example shows the kernel-cached MLPs for the global zone. The output reflects the definition in the `tnzonecfg` database, plus any dynamically allocated MLPs. `private` indicates zone-specific MLPs.

```
# tninfo -m global
private:23/tcp;111/tcp;111/udp;515/tcp;2049/tcp;6000-6003/tcp;
        32812/tcp;36698/ip;38634/tcp;64365/ip
shared: 6000-6003/tcp
```

EXAMPLE 3 Displaying the `cipso` Template Definition

This example shows the kernel-cached `cipso` template definition. The output reflects the definition in the `tnrhtp` database.

```
# tninfo -t cipso
====================================
   Remote Host Template Table Entries:

   ___________________________
   template: cipso
   host_type: CIPSO
   doi: 1
   min_sl: ADMIN_LOW
   hex: ADMIN_LOW
   max_sl: ADMIN_HIGH
```

EXAMPLE 3 Displaying the `cipso` Template Definition *(Continued)*

```
    hex: ADMIN_HIGH
```

Files

`/etc/security/tsol/tnrhdb`	Trusted network remote-host database
`/etc/security/tsol/tnrhtp`	Trusted network remote-host templates
`/etc/security/tsol/tnzonecfg`	Trusted zone configuration database

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	See below.

The command line is Committed. The output is Uncommitted.

See Also `tnd(1M)`, `tnctl(1M)`, `attributes(5)`

"How to Synchronize the Kernel Cache With Trusted Network Databases" in *Oracle Solaris Trusted Extensions Administrator's Procedures*

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

Name traceroute – print the route packets take to network host

Synopsis

```
traceroute [-adFIlnSvx] [-A addr_family] [-c traffic_class]
     [-f first_hop] [-g gateway [-g gateway...] | -r]
     [-i iface] [-L flow_label] [-m max_hop]
     [-P pause_sec] [-p port] [-Q max_timeout]
     [-q nqueries] [-s src_addr] [-t tos] [-w wait_time] host
     [packetlen]
```

Description The Internet is a large and complex aggregation of network hardware, connected by gateways. Tracking the route a packet follows can be difficult. The utility `traceroute` traces the route that an IP packet follows to another internet host.

The `traceroute` utility utilizes the both the IPv4 and IPv6 protocols. Use the `-A` option to override the default behavior. `traceroute` uses the IPv4 protocol *ttl* (time to live) field or the IPv6 field *hop limit*. It attempts to elicit an ICMP or ICMP6 `TIME_EXCEEDED` response from each *gateway* along the path, and a `PORT_UNREACHABLE`(or `ECHO_REPLY` if `-I` is used) response from the destination host. It starts by sending probes with a *ttl* or *hop limit* of `1` and increases by `1` until it either gets to the host, or it hits the maximum *max_hop*. The default maximum *max_hop* is 30 hops, but this can be set by the `-m` option.

Three probes are sent at each *ttl* (*hop limit*) setting, and a line is printed showing the *ttl* (*hop limit*), the hostname and the address of the gateway, and the *rtt* (round trip time) of each probe. The number of probes may be specifically set using the `-q` option. If the probe answers come from different gateways, the hostname and the address of each responding system will be printed. If there is no response within a 5 second timeout interval, an asterisk (`*`) is displayed for that probe. The `-w` option may be used to set the timeout interval. Other possible annotations that may appear after the time are:

`!`
: the *ttl* (*hop limit*) value in the received packet is `<= 1`.

`!H`
: host unreachable.

`!X`
: communication administratively prohibited.

`<!N>`
: ICMP (ICMP6) unreachable code N.

The following annotations appear only for IPv4:

`!F`
: fragmentation needed. This should never occur. If this is seen, the associated gateway is broken.

`!N`
: network unreachable.

`!P`
protocol unreachable.

`!S`
source route failed. It is likely that the gateway does not support source routing.

`!T`
unreachable for the specified tos (type-of-service).

`!U`
source host isolated or precedence problem.

The following annotations appear only for IPv6:

`!A`
host unreachable for a reason other than lack of an entry in the routing table.

`!B`
packet too big.

`!E`
destination is not a neighbor.

`!R`
unrecognized next header.

If almost all the probes result in some kind of unreachable code, then `traceroute` gives up and exits.

The destination *host* is not supposed to process the UDP probe packets, so the destination *port* default is set to an unlikely value. However, if some application on the destination is using that value, the value of *port* can be changed with the `-p` option.

The only mandatory parameter is the destination *host* name or IP number. The default probe datagram length is 40 bytes (60 bytes for IPv6), but this may be increased by specifying a packet length (in bytes) after the destination *host* name.

All integer arguments to `traceroute` can be specified in either decimal or hexadecimal notation. For example, *packetlen* can be specified either as `256` or `0x100`.

Options `-A` *addr_family*
Specify the address family of the target host. *addr_family* can be either `inet` or `inet6`. Address family determines which protocol to use. For an argument of `inet`, IPv4 is used. For `inet6`, IPv6 is used.

By default, if the name of a host is provided, not the literal IP address, and a valid IPv6 address exists in the name service database, `traceroute` will use this address. Otherwise, if the name service database contains an IPv4 address, it will try the IPv4 address.

Specify the address family `inet` or `inet6` to override the default behavior. If the argument specified is `inet`, `traceroute` will use the IPv4 address associated with the hostname. If none exists, `traceroute` will state that the host is unknown and exit. It will not try to determine if an IPv6 address exists in the name service database.

If the specified argument is `inet6`, `traceroute` will use the IPv6 address that is associated with the hostname. If none exists, `traceroute` will state that the host is unknown and exit.

`-a`
Probe all of the addresses of a multi-homed destination. The output looks like `traceroute` has been run once for each IP address of the destination. If this option is used together with `-A`, `traceroute` probes only the addresses that are of the specified address family. While probing one of the addresses of the destination, user can skip to the next address by sending a `SIGINT`, or exit `traceroute` by sending a `SIGQUIT` signal. See `signal(3C)`

`-c` *traffic_class*
Specify the traffic class of probe packets. The value must be an integer in the range from 0 to 255. Gateways along the path may route the probe packet differently depending upon the value of *traffic_class* set in the probe packet. This option is valid only on IPv6.

`-d`
Set the `SO_DEBUG` socket option.

`-F`
Set the "don't fragment" bit. This option is valid only on IPv4. When specified from within a shared-IP zone, this option has no effect as the "don't fragment" bit is always set in this case.

`-f` *first_hop*
Set the starting *ttl* (*hop limit*) value to *first_hop*, to override the default value `1`. `traceroute` skips processing for those intermediate gateways which are less than *first_hop* hops away.

`-g` *gateway*
Specify a loose source route *gateway*. The user can specify more than one *gateway* by using `-g` for each gateway. The maximum number of gateways is 8 for IPv4 and 127 for IPv6. Note that some factors such as the link MTU can further limit the number of gateways for IPv6. This option cannot be used with the `-r` option.

Only users with the `{PRIV_NET_RAWACCESS}` privilege can specify a loose source route with this option.

`-I`
Use ICMP (ICMP6) `ECHO` instead of UDP datagrams.

`-i` *iface*
For IPv4, this option specifies a network interface to obtain the source IP address. This is normally only useful on a multi-homed host. The `-s` option is also another way to do this.

For IPv6, it specifies the network interface on which probe packets are transmitted. The argument can be either an interface index, for example, `1`, `2`, or an interface name, for example, `eri0`, `hme0`.

`-L` *flow_label*
Specify the flow label of probe packets. The value must be an integer in the range from 0 to 1048575. This option is valid only on IPv6.

`-l`
Print the value of the *ttl* (*hop limit*) field in each packet received.

`-m` *max_hop*
Set the maximum *ttl* (*hop limit*) used in outgoing probe packets. The default is 30 hops, which is the same default used for TCP connections.

`-n`
Print hop addresses numerically rather than symbolically and numerically. This saves a nameserver address-to-name lookup for each gateway found on the path.

`-P` *pause_sec*
Specify a delay, in seconds, to pause between probe packets. This may be necessary if the final destination does not accept undeliverable packets in bursts. By default, `traceroute` sends the next probe as soon as it has received a reply. Note that *pause_sec* is a real number.

`-p` *port*
Set the base UDP *port* number used in probes.The default is 33434. `traceroute` hopes that nothing is listening on UDP *ports* `(base+(nhops-1)*nqueries)` to `(base+(nhops*nqueries)-1)` at the destination host, so that an ICMP (ICMP6) `PORT_UNREACHABLE` message will be returned to terminate the route tracing. If something is listening on a *port* in the default range, this option can be used to select an unused *port* range. `nhops` is defined as the number of hops between the source and the destination.

`-Q` *max_timeout*
Stop probing this hop after *max_timeout* consecutive timeouts are detected. The default value is 5. Useful in combination with the `-q` option if you have specified a large *nqueries* probe count.

`-q` *nqueries*
Set the desired number of probe queries. The default is 3.

`-r`
Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to send probes to a local host through an interface that has been dropped by the router deamon. See `in.routed(1M)`. You cannot use this option if the `-g` option is used.

`-S`
Display a summary of how many probes were not answered for each hop.

`-s` *src_addr*
Use the following address, which usually is given as a literal IP address, not a hostname, as the source address in outgoing probe packets. On multi-homed hosts, those with more than one IP address, this option can be used to force the source address to be something other than the IP address `traceroute` picks by default. If the IP address is not one of this machine's interface addresses, an error is returned and nothing is sent. For IPv4, when used together with the `-i` option, the given IP address should be configured on the specified interface. Otherwise, an error will be returned. In the case of IPv6, the interface name and the source address do not have to match.

`-t` *tos*
Set the *tos*(type-of-service) in probe packets to the specified value. The default is zero. The value must be an integer in the range from 0 to 255. Gateways along the path may route the probe packet differently depending upon the *tos* value set in the probe packet. This option is valid only on IPv4.

`-v`
Verbose output. For each hop, the size and the destination of the response packets is displayed. Also ICMP (ICMP6) packets received other than `TIME_EXCEEDED` and `UNREACHABLE` are listed as well.

`-w` *waittime*
Set the time, in seconds, to wait for a response to a probe. The default is 5 seconds.

`-x`
Prevent `traceroute` from calculating checksums. Checksums are usually required for the last hop when using ICMP `ECHO` probes. This option is valid only on IPv4. See the `-I` option.

When specified from within a shared-IP zone, this option has no effect as the checksum is always calculated by the operating system in this case.

Operands

The following operands are supported:

host
The network host.

Examples

EXAMPLE 1 Sample Output From the `traceroute` Utility

Some sample output from the `traceroute` utility might be:

```
istanbul% traceroute london
traceroute: Warning: london has multiple addresses;  \
  using 4::114:a00:20ff:ab3d:83ed
traceroute: Warning: Multiple interfaces found; \
  using 4::56:a00:20ff:fe93:8dde @ eri0:2
traceroute to london (4::114:a00:20ff:ab3d:83ed), 30 hops max, \
  60 byte packets
1  frbldg7c-86 (4::56:a00:20ff:fe1f:65a1)  1.786 ms  1.544 ms  1.719 ms
```

EXAMPLE 1 Sample Output From the `traceroute` Utility *(Continued)*

```
2  frbldg7b-77 (4::255:0:0:c0a8:517)  2.587 ms 3.001 ms  2.988 ms
3  london (4::114:a00:20ff:ab3d:83ed)  3.122 ms  2.744 ms  3.356 ms
```

The target host, london, has both IPv4 and IPv6 addresses in the name service database. According to the default behavior, `traceroute` uses IPv6 address of the destination host.

EXAMPLE 2 Using the `traceroute` Utility For a Host Which has Only IPv4 Addresses

In the following examples, `traceroute` is tracking the route to host sanfrancisco, which has only IPv4 addresses in the name service database. Therefore traceroute uses only IPv4 addresses. The following shows the 7-hop path that a packet would follow from the host istanbul to the host sanfrancisco.

```
istanbul% traceroute sanfrancisco
traceroute: Warning: Multiple interfaces found; using 172.31.86.247 @eri0
traceroute to sanfrancisco (172.29.64.39), 30 hops max, 40 byte packets
1  frbldg7c-86 (172.31.86.1)  1.516 ms  1.283 ms  1.362 ms
2  bldg1a-001 (172.31.1.211)  2.277 ms  1.773 ms  2.186 ms
3  bldg4-bldg1 (172.30.4.42)  1.978 ms  1.986 ms  13.996 ms
4  bldg6-bldg4 (172.30.4.49)  2.655 ms  3.042 ms  2.344 ms
5  ferbldg11a-001 (172.29.1.236)  2.636 ms  3.432 ms  3.830 ms
6  frbldg12b-153 (172.29.153.72)  3.452 ms  3.146 ms  2.962 ms
7  sanfrancisco (172.29.64.39)  3.430 ms  3.312 ms  3.451 ms
```

EXAMPLE 3 Using the `traceroute` Utility With Source Routing

The following example shows the path of a packet that goes from istanbul to sanfrancisco through the hosts cairo and paris, as specified by the `-g` option. The `-I` option makes `traceroute` send ICMP `ECHO` probes to the host sanfrancisco. The `-i` options sets the source address to the IP address configured on the interface `qe0`.

```
istanbul% traceroute -g cairo -g paris -i qe0 -q 1 -I sanfrancisco
traceroute to sanfrancisco (172.29.64.39), 30 hops max, 56 byte packets
1  frbldg7c-86 (172.31.86.1)  2.012 ms
2  flrbldg7u (172.31.17.131)  4.960 ms
3  cairo (192.168.163.175) 4.894 ms
4  flrbldg7u (172.31.17.131)  3.475 ms
5  frbldg7c-017 (172.31.17.83)  4.126 ms
6  paris (172.31.86.31) 4.086 ms
7  frbldg7b-82 (172.31.82.1)  6.454 ms
8  bldg1a-001 (172.31.1.211)  6.541 ms
9  bldg6-bldg4 (172.30.4.49)  6.518 ms
10  ferbldg11a-001 (172.29.1.236)  9.108 ms
11  frbldg12b-153 (172.29.153.72)  9.634 ms
12  sanfrancisco (172.29.64.39)  14.631 ms
```

Exit Status The following exit values are returned:

`0`
Successful operation.

`>0`
An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `netstat(1M)`, `signal(3C)`, `ping(1M)`, `attributes(5)`, `privileges(5)`, `zones(5)`

Warnings This utility is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use `traceroute`(1M) during normal operations or from automated scripts.

Name trapstat – report trap statistics

Synopsis

```
/usr/sbin/trapstat [-t | -T | -e entry]
     [-C processor_set_id | -c cpulist] [-P] [-a]
     [-r rate] [ [interval [count]] | command | [args]]

/usr/sbin/trapstat -l
```

Description The `trapstat` utility gathers and displays run-time trap statistics on UltraSPARC-based systems. The default output is a table of trap types and CPU IDs, with each row of the table denoting a trap type and each column of the table denoting a CPU. If standard output is a terminal, the table contains as many columns of data as can fit within the terminal width; if standard output is not a terminal, the table contains at most six columns of data. By default, data is gathered and and displayed for all CPUs; if the data cannot fit in a single table, it is printed across multiple tables. The set of CPUs for which data is gathered and displayed can be optionally specified with the `-c` or `-C` option.

Unless the `-r` option or the `-a` option is specified, the value displayed in each entry of the table corresponds to the number of traps per second. If the `-r` option is specified, the value corresponds to the number of traps over the interval implied by the specified sampling rate; if the `-a` option is specified, the value corresponds to the accumulated number of traps since the invocation of `trapstat`.

By default, `trapstat` displays data once per second, and runs indefinitely; both of these behaviors can be optionally controlled with the *interval* and *count* parameters, respectively. The *interval* is specified in seconds; the *count* indicates the number of intervals to be executed before exiting. Alternatively, `command` can be specified, in which case `trapstat` executes the provided command and continues to run until the command exits. A positive integer is assumed to be an *interval*; if the desired *command* cannot be distinguished from an integer, the full path of *command* must be specified.

UltraSPARC I (obsolete), II, and III handle translation lookaside buffer (TLB) misses by trapping to the operating system. TLB miss traps can be a significant component of overall system performance for some workloads; the `-t` option provides in-depth information on these traps. When run with this option, `trapstat` displays both the rate of TLB miss traps and the percentage of time spent processing those traps. Additionally, TLB misses that hit in the translation storage buffer (TSB) are differentiated from TLB misses that further miss in the TSB. (The TSB is a software structure used as a translation entry cache to allow the TLB to be quickly filled; it is discussed in detail in the *UltraSPARC II User's Manual*.) The TLB and TSB miss information is further broken down into user- and kernel-mode misses.

Workloads with working sets that exceed the TLB reach may spend a significant amount of time missing in the TLB. To accommodate such workloads, the operating system supports multiple page sizes: larger page sizes increase the effective TLB reach and thereby reduce the number of TLB misses. To provide insight into the relationship between page size and TLB miss rate, `trapstat` optionally provides in-depth TLB miss information broken down by page size using the `-T` option. The information provided by the `-T` option is a superset of that provided by the `-t` option; only one of `-t` and `-T` can be specified.

Options The following options are supported:

`-a` Displays the number of traps as accumulating, monotonically increasing values instead of per-second or per-interval rates.

`-c` *cpulist* Enables `trapstat` only on the CPUs specified by *cpulist*.

cpulist can be a single processor ID (for example, `4`), a range of processor IDs (for example, `4-6`), or a comma separated list of processor IDs or processor ID ranges (for example, `4,5,6` or `4,6-8`).

`-C` *processor_set_id* Enables `trapstat` only on the CPUs in the processor set specified by *processor_set_id*.

`trapstat` modifies its output to always reflect the CPUs in the specified processor set. If a CPU is added to the set, `trapstat` modifies its output to include the added CPU; if a CPU is removed from the set, `trapstat` modifies its output to exclude the removed CPU. At most one processor set can be specified.

`-e` *entrylist* Enables `trapstat` only for the trap table entry or entries specified by *entrylist*. A trap table entry can be specified by trap number or by trap name (for example, the level–10 trap can be specified as `74`, `0x4A`, `0x4a`, or `level-10`).

entrylist can be a single trap table entry or a comma separated list of trap table entries. If the specified trap table entry is not valid, `trapstat` prints a table of all valid trap table entries and values. A list of valid trap table entries is also found in *The SPARC Architecture Manual, Version 9* and the *Sun Microelectronics UltraSPARC II User's Manual*. If the parsable option (`-P`) is specified in addition to the `-e` option, the format of the data is as follows:

Field	Contents
1	Timestamp (nanoseconds since start)
2	CPU ID
3	Trap number (in hexadecimal)
4	Trap name
5	Trap rate per interval

Each field is separated with whitespace. If the format is modified, it will be modified by adding potentially new fields beginning with field 6; exant fields will remain unchanged.

`-l` Lists trap table entries. By default, a table is displayed containing all valid trap numbers, their names and a brief description. The trap name is used in both the default output and in the *entrylist* parameter for the `-e` argument. If the parsable option (`-P`) is specified in addition to the `-l` option, the format of the data is as follows:

Field	Contents
1	Trap number in hexadecimal
2	Trap number in decimal
3	Trap name
Remaining	Trap description

`-P` Generates parsable output. When run without other data gathering modifying options (that is, `-e`, `-t` or `-T`), `trapstat`'s the parsable output has the following format:

Field	Contents
1	Timestamp (nanoseconds since start)
2	CPU ID
3	Trap number (in hexadecimal)
4	Trap name
5	Trap rate per interval

Each field is separated with whitespace. If the format is modified, it will be modified by adding potentially new fields beginning with field 6; extant fields will remain unchanged.

`-r` *rate* Explicitly sets the sampling rate to be *rate* samples per second. If this option is specified, `trapstat`'s output changes from a traps-per-second to traps-per-sampling-interval.

`-t` Enables TLB statistics.

A table is displayed with four principal columns of data: *itlb-miss*, *itsb-miss*, *dtlb-miss*, and *dtsb-miss*. The columns contain both the rate of the corresponding event and the percentage of CPU time spent processing the event. The percentage of CPU time is given only in terms of a single CPU. The rows of the table correspond to CPUs,

with each CPU consuming two rows: one row for user-mode events (denoted with `u`) and one row for kernel-mode events (denoted with `k`). For each row, the percentage of CPU time is totalled and displayed in the rightmost column. The CPUs are delineated with a solid line. If the parsable option (`-P`) is specified in addition to the `-t` option, the format of the data is as follows:

Field	Contents
1	Timestamp (nanoseconds since start)
2	CPU ID
3	Mode (`k` denotes kernel, `u` denotes user)
4	I-TLB misses
5	Percentage of time in I-TLB miss handler
6	I-TSB misses
7	Percentage of time in I-TSB miss handler
8	D-TLB misses
9	Percentage of time in D-TLB miss handler
10	D-TSB misses
11	Percentage of time in D-TSB miss handler

Each field is separated with whitespace. If the format is modified, it will be modified by adding potentially new fields beginning with field 12; extant fields will remain unchanged.

`-T` Enables TLB statistics, with page size information. As with the `-t` option, a table is displayed with four principal columns of data: *itlb-miss*, *itsb-miss*, *dtlb-miss*, and *dtsb-miss*. The columns contain both the absolute number of the corresponding event, and the percentage of CPU time spent processing the event. The percentage of CPU time is given only in terms of a single CPU. The rows of the table correspond to CPUs, with each CPU consuming two sets of rows: one set for user-level events (denoted with `u`) and one set for kernel-level events (denoted with `k`). Each set, in turn, contains as many rows as

there are page sizes supported (see `getpagesizes(3C)`). For each row, the percentage of CPU time is totalled and displayed in the right-most column. The two sets are delineated with a dashed line; CPUs are delineated with a solid line. If the parsable option (`-P`) is specified in addition to the `-T` option, the format of the data is as follows:

Field	Contents
1	Timestamp (nanoseconds since start)
2	CPU ID
3	Mode `k` denotes kernel, `u` denotes user)
4	Page size, in decimal
5	I-TLB misses
6	Percentage of time in I-TLB miss handler
7	I-TSB misses
8	Percentage of time in I-TSB miss handler
9	D-TLB misses
10	Percentage of time in D-TLB miss handler
11	D-TSB misses
12	Percentage of time in D-TSB miss handler

Each field is separated with whitespace. If the format is modified, it will be modified by adding potentially new fields beginning with field 13; extant fields will remain unchanged.

Examples

EXAMPLE 1 Using `trapstat` Without Options

When run without options, `trapstat` displays a table of trap types and CPUs. At most six columns can fit in the default terminal width; if (as in this example) there are more than six CPUs, multiple tables are displayed:

```
example# trapstat
vct  name                |     cpu0     cpu1     cpu4     cpu5     cpu8     cpu9
-----------------------+-------------------------------------------------------
 24 cleanwin             |     6446     4837     6368     2153     2623     1321
 41 level-1              |      100        0        0        0        1        0
```

EXAMPLE 1 Using `trapstat` Without Options *(Continued)*

```
 44 level-4             |        0        1        1        1        0        0
 45 level-5             |        0        0        0        0        0        0
 47 level-7             |        0        0        0        0        9        0
 49 level-9             |      100      100      100      100      100      100
 4a level-10            |      100        0        0        0        0        0
 4d level-13            |        6       10        7       16       13       11
 4e level-14            |      100        0        0        0        1        0
 60 int-vec             |     2607     2740     2642     2922     2920     3033
 64 itlb-miss           |     3129     2475     3167     1037     1200      569
 68 dtlb-miss           |   121061    86162   109838    37386    45639    20269
 6c dtlb-prot           |      997      847     1061      379      406      184
 84 spill-user-32       |     2809     2133     2739   200806   332776   454504
 88 spill-user-64       |    45819   207856    93487   228529    68373    77590
 8c spill-user-32-cln   |      784      561      767      274      353      215
 90 spill-user-64-cln   |        9       37       17       39       12       13
 98 spill-kern-64       |    62913    50145    63869    21916    28431    11738
 a4 spill-asuser-32     |     1327      947     1288      460      572      335
 a8 spill-asuser-64     |       26       48       18       54       10       14
 ac spill-asuser-32-cln |     4580     3599     4555     1538     1978      857
 b0 spill-asuser-64-cln |       26        0        0        2        0        0
 c4 fill-user-32        |     2862     2161     2798   191746   318115   435850
 c8 fill-user-64        |    45813   197781    89179   217668    63905    74281
 cc fill-user-32-cln    |     3802     2833     3733    10153    16419    19475
 d0 fill-user-64-cln    |      329    10105     4873    10603     4235     3649
 d8 fill-kern-64        |    62519    49943    63611    21824    28328    11693
108 syscall-32          |     2285     1634     2278      737      957      383
126 self-xcall          |      100        0        0        0        0        0

vct  name               |    cpu12    cpu13    cpu14    cpu15
------------------------+-----------------------------------
 24 cleanwin            |     5435     4232     6302     6104
 41 level-1             |        0        0        0        0
 44 level-4             |        2        0        0        1
 45 level-5             |        0        0        0        0
 47 level-7             |        0        0        0        0
 49 level-9             |      100      100      100      100
 4a level-10            |        0        0        0        0
 4d level-13            |       15       11       22       11
 4e level-14            |        0        0        0        0
 60 int-vec             |     2813     2833     2738     2714
 64 itlb-miss           |     2636     1925     3133     3029
 68 dtlb-miss           |    90528    70639   107786   103425
 6c dtlb-prot           |      819      675      988      954
 84 spill-user-32       |   175768    39933     2811     2742
 88 spill-user-64       |        0   241348    96907   118298
 8c spill-user-32-cln   |      681      513      753      730
```

EXAMPLE 1 Using `trapstat` Without Options *(Continued)*

```
 90 spill-user-64-cln   |        0       42       16       20
 98 spill-kern-64       |    52158    40914    62305    60141
 a4 spill-asuser-32     |     1113      856     1251     1208
 a8 spill-asuser-64     |        0       64       16       24
 ac spill-asuser-32-cln |     3816     2942     4515     4381
 b0 spill-asuser-64-cln |        0        0        0        0
 c4 fill-user-32        |   170744    38444     2876     2784
 c8 fill-user-64        |        0   230381    92941   111694
 cc fill-user-32-cln    |     8550     3790     3612     3553
 d0 fill-user-64-cln    |        0    10726     4495     5845
 d8 fill-kern-64        |    51968    40760    62053    59922
108 syscall-32          |     1839     1495     2144     2083
126 self-xcall          |        0        0        0        0
```

EXAMPLE 2 Using `trapset` with CPU Filtering

The `-c` option can be used to limit the CPUs on which `trapstat` is enabled. This example limits `CPU` 1 and CPUs 12 through 15.

```
example# trapstat -c 1,12-15
```

```
vct  name                |     cpu1    cpu12    cpu13    cpu14    cpu15
-------------------------+-----------------------------------------------
 24 cleanwin             |     6923     3072     2500     3518     2261
 44 level-4              |        3        0        0        1        1
 49 level-9              |      100      100      100      100      100
 4d level-13             |       23        8       14       19       14
 60 int-vec              |     2559     2699     2752     2688     2792
 64 itlb-miss            |     3296     1548     1174     1698     1087
 68 dtlb-miss            |   114788    54313    43040    58336    38057
 6c dtlb-prot            |     1046      549      417      545      370
 84 spill-user-32        |    66551    29480   301588    26522   213032
 88 spill-user-64        |        0   318652   111239   299829   221716
 8c spill-user-32-cln    |      856      347      331      416      293
 90 spill-user-64-cln    |        0       55       21       59       39
 98 spill-kern-64        |    66464    31803    24758    34004    22277
 a4 spill-asuser-32      |     1423      569      560      698      483
 a8 spill-asuser-64      |        0       74       32       98       46
 ac spill-asuser-32-cln  |     4875     2250     1728     2384     1584
 b0 spill-asuser-64-cln  |        0        2        0        1        0
 c4 fill-user-32         |    64193    28418   287516    27055   202093
 c8 fill-user-64         |        0   305016   106692   288542   210654
 cc fill-user-32-cln     |     6733     3520    15185     2396    12035
 d0 fill-user-64-cln     |        0    13226     3506    12933    11032
 d8 fill-kern-64         |    66220    31680    24674    33892    22196
108 syscall-32           |     2446      967      817     1196      755
```

EXAMPLE 3 Using `trapstat` with TLB Statistics

The `-t` option displays in-depth TLB statistics, including the amount of time spent performing TLB miss processing. The following example shows that the machine is spending 14.1 percent of its time just handling D-TLB misses:

```
example# trapstat -t
cpu m| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
-----+-------------------------------+-------------------------------+----
  0 u|      2571  0.3         0  0.0 |     10802  1.3         0  0.0 | 1.6
  0 k|         0  0.0         0  0.0 |    106420 13.4       184  0.1 |13.6
-----+-------------------------------+-------------------------------+----
  1 u|      3069  0.3         0  0.0 |     10983  1.2       100  0.0 | 1.6
  1 k|        27  0.0         0  0.0 |    106974 12.6        19  0.0 |12.7
-----+-------------------------------+-------------------------------+----
  2 u|      3033  0.3         0  0.0 |     11045  1.2       105  0.0 | 1.6
  2 k|        43  0.0         0  0.0 |    107842 12.7       108  0.0 |12.8
-----+-------------------------------+-------------------------------+----
  3 u|      2924  0.3         0  0.0 |     10380  1.2       121  0.0 | 1.6
  3 k|        54  0.0         0  0.0 |    102682 12.2        16  0.0 |12.2
-----+-------------------------------+-------------------------------+----
  4 u|      3064  0.3         0  0.0 |     10832  1.2       120  0.0 | 1.6
  4 k|        31  0.0         0  0.0 |    107977 13.0       236  0.1 |13.1
=====+===============================+===============================+====
 ttl |     14816  0.3         0  0.0 |    585937 14.1      1009  0.0 |14.5
```

EXAMPLE 4 Using `trapstat` with TLB Statistics and Page Size Information

By specifying the `-T` option, `trapstat` shows TLB misses broken down by page size. In this example, CPU 0 is spending 7.9 percent of its time handling user-mode TLB misses on 8K pages, and another 2.3 percent of its time handling user-mode TLB misses on 64K pages.

```
example# trapstat -T -c 0
cpu m size| itlb-miss %tim itsb-miss %tim | dtlb-miss %tim dtsb-miss %tim |%tim
----------+-------------------------------+-------------------------------+----
  0 u   8k|      1300  0.1        15  0.0 |    104897  7.9        90  0.0 | 8.0
  0 u  64k|         0  0.0         0  0.0 |     29935  2.3         7  0.0 | 2.3
  0 u 512k|         0  0.0         0  0.0 |      3569  0.2         2  0.0 | 0.2
  0 u   4m|         0  0.0         0  0.0 |       233  0.0         2  0.0 | 0.0
- - - - - + - - - - - - - - - - - - - - - + - - - - - - - - - - - - - - - + - -
  0 k   8k|        13  0.0         0  0.0 |     71733  6.5       110  0.0 | 6.5
  0 k  64k|         0  0.0         0  0.0 |         0  0.0         0  0.0 | 0.0
  0 k 512k|         0  0.0         0  0.0 |         0  0.0       206  0.1 | 0.1
  0 k   4m|         0  0.0         0  0.0 |         0  0.0         0  0.0 | 0.0
==========+===============================+===============================+====
      ttl |      1313  0.1        15  0.0 |    210367 17.1       417  0.2 |17.5
```

EXAMPLE 5 Using `trapstat` with Entry Filtering

By specifying the `-e` option, `trapstat` displays statistics for only specific trap types. Using this option minimizes the probe effect when seeking specific data. This example yields statistics for only the *dtlb-prot* and *syscall-32* traps on CPUs 12 through 15:

```
example# trapstat -e dtlb-prot,syscall-32 -c 12-15
vct  name                |    cpu12    cpu13    cpu14    cpu15
-------------------------+-----------------------------------
 6c dtlb-prot            |      817      754     1018      560
108 syscall-32           |     1426     1647     2186     1142

vct  name                |    cpu12    cpu13    cpu14    cpu15
-------------------------+-----------------------------------
 6c dtlb-prot            |     1085      996      800      707
108 syscall-32           |     2578     2167     1638     1452
```

EXAMPLE 6 Using `trapstat` with a Higher Sampling Rate

The following example uses the `-r` option to specify a sampling rate of 1000 samples per second, and filter only for the level-10 trap. Additionally, specifying the `-P` option yields parsable output.

Notice the timestamp difference between the level-10 events: 9,998,000 nanoseconds and 10,007,000 nanoseconds. These level-10 events correspond to the system clock, which by default ticks at 100 hertz (that is, every 10,000,000 nanoseconds).

```
example# trapstat -e level-10 -P -r 1000
1070400 0 4a level-10 0
2048600 0 4a level-10 0
3030400 0 4a level-10 1
4035800 0 4a level-10 0
5027200 0 4a level-10 0
6027200 0 4a level-10 0
7027400 0 4a level-10 0
8028200 0 4a level-10 0
9026400 0 4a level-10 0
10029600 0 4a level-10 0
11028600 0 4a level-10 0
12024000 0 4a level-10 0
13028400 0 4a level-10 1
14031200 0 4a level-10 0
15027200 0 4a level-10 0
16027600 0 4a level-10 0
17025000 0 4a level-10 0
18026000 0 4a level-10 0
19027800 0 4a level-10 0
20025600 0 4a level-10 0
21025200 0 4a level-10 0
22025000 0 4a level-10 0
```

EXAMPLE 6 Using `trapstat` with a Higher Sampling Rate *(Continued)*

```
23035400 0 4a level-10 1
24027400 0 4a level-10 0
25026000 0 4a level-10 0
26027000 0 4a level-10 0
```

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	
Human Readable Output	Unstable
Parsable Output	Evolving

See Also

`lockstat(1M)`, `pmap(1)`, `psrset(1M)`, `psrinfo(1M)`, `pbind(1M)`, `ppgsz(1)`, `getpagesizes(3C)`

Sun Microelectronics UltraSPARC II User's Manual, January 1997, STP1031,

The SPARC Architecture Manual, Version 9, 1994, Prentice-Hall.

Notes

When enabled, `trapstat` induces a varying probe effect, depending on the type of information collected. While the precise probe effect depends upon the specifics of the hardware, the following table can be used as a rough guide:

Option	Approximate probe effect
default	3-5% per trap
`-e`	3-5% per specified trap
`-t`, `-T`	40-45% per TLB miss trap hitting in the TSB, 25-30% per TLB miss trap missing in the TSB

These probe effects are *per trap* not for the system as a whole. For example, running `trapstat` with the default options on a system that spends 7% of total time handling traps induces a performance degradation of less than one half of one percent; running `trapstat` with the `-t` or `-T` option on a system spending 5% of total time processing TLB misses induce a performance degradation of no more than 2.5%.

When run with the `-t` or `-T` option, `trapstat` accounts for its probe effect when calculating the *%tim* fields. This assures that the *%tim* fields are a reasonably accurate indicator of the time a given workload is spending handling TLB misses — regardless of the perturbing presence of `trapstat`.

While the *%tim* fields include the explicit cost of executing the TLB miss handler, they do *not* include the implicit costs of TLB miss traps (for example, pipeline effects, cache pollution, etc). These implicit costs become more significant as the trap rate grows; if high *%tim* values are reported (greater than 50%), you can accurately infer that much of the balance of time is being spent on the implicit costs of the TLB miss traps.

Due to the potential system wide degradation induced, only the super-user can run `trapstat`.

Due to the limitation of the underlying statistics gathering methodology, only one instance of `trapstat` can run at a time.

Name ttyadm – format and output port monitor-specific information

Synopsis

```
/usr/sbin/ttyadm [-b] [-c] [-h] [-I] [-r count] [-i msg]
     [-m modules] [-p prompt] [-t timeout] [-S y | n]
     [-T termtype] -d device -l ttylabel -s service

/usr/sbin/ttyadm -V
```

Description The `ttyadm` command is an administrative command that formats `ttymon`(1M)-specific information and writes it to standard output. The Service Access Facility (SAF) requires each port monitor to provide such a command. Note that the port monitor administrative file is updated by the Service Access Controller's administrative commands, `sacadm`(1M) and `pmadm`(1M). `ttyadm` provides a means of presenting formatted port monitor-specific (`ttymon`-specific) data to these commands.

Options The following options are supported:

`-b` Set the "bi-directional port" flag. When this flag is set, the line can be used in both directions. `ttymon` will allow users to connect to the service associated with the port, but if the port is free, `uucico`(1M), `cu`(1C), or `ct`(1C) can use it for dialing out.

`-c` Set the connect-on-carrier flag for the port. If the `-c` flag is set, `ttymon` will invoke the port's associated service immediately when a connect indication is received (that is, no prompt is printed and no baud-rate searching is done).

`-d` *device* *device* is the full pathname of the device file for the TTY port.

`-h` Set the hangup flag for the port. If the `-h` flag is not set, `ttymon` will force a hangup on the line by setting the speed to `0` before setting the speed to the default or specified value.

`-i` *message* Specify the inactive (disabled) response message. This message will be sent to the TTY port if the port is disabled or the `ttymon` monitoring the port is disabled.

`-I` Initialize only. If the `-I` option is used, `ttymon` will invoke the service only once. This can be used to configure a particular device without actually monitoring it, as with software carrier.

`-l` *ttylabel* Specify which *ttylabel* in the `/etc/ttydefs` file to use as the starting point when searching for the proper baud rate.

`-m` *modules* Specify a list of pushable STREAMS modules. The modules will be pushed in the order in which they are specified before the service is invoked. *modules* must be a comma-separated list of modules, with no white space included. Any modules currently on the stream will be popped before these modules are pushed.

`-p` *prompt* Specify the prompt message, for example, "`login:`".

`-r` *count* — When the `-r` option is invoked, `ttymon` will wait until it receives data from the port before it displays a prompt. If *count* is `0`, `ttymon` will wait until it receives any character. If *count* is greater than `0`, `ttymon` will wait until *count* newlines have been received.

`-s` *service* — *service* is the full pathname of the service to be invoked when a connection request is received. If arguments are required, the command and its arguments must be enclosed in double quotes (" ").

`-S y|n` — Set the software carrier value. `y` will turn software carrier on. `n` will turn software carrier off.

`-t` *timeout* — Specify that `ttymon` should close a port if the open on the port succeeds, and no input data is received in *timeout* seconds.

`-T` *termtype* — Set the terminal type. The `TERM` environment variable will be set to *termtype*.

`-V` — Display the version number of the current `/usr/lib/saf/ttymon` command.

Output If successful, `ttyadm` will generate the requested information, write it to standard output, and exit with a status of `0`. If `ttyadm` is invoked with an invalid number of arguments or invalid arguments, or if an incomplete option is specified, an error message will be written to standard error and `ttymon` will exit with a non-zero status.

Files `/etc/ttydefs`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `ct(1C)`, `cu(1C)`, `pmadm(1M)`, `sacadm(1M)`, `ttymon(1M)`, `uucico(1M)`, `attributes(5)`

System Administration Guide: Basic Administration

Name ttymon – port monitor for terminal ports

Synopsis

```
/usr/lib/saf/ttymon

/usr/lib/saf/ttymon  -g [-d device] [-h] [-t timeout]
 [-l ttylabel] [-p prompt] [-m modules] [-T termtype]
```

Description `ttymon` is a STREAMS-based TTY port monitor. Its function is to monitor ports, to set terminal modes, baud rates, and line disciplines for the ports, and to connect users or applications to services associated with the ports. Normally, `ttymon` is configured to run under the Service Access Controller, `sac(1M)`, as part of the Service Access Facility (SAF). It is configured using the `sacadm(1M)` command. Each instance of `ttymon` can monitor multiple ports. The ports monitored by an instance of `ttymon` are specified in the port monitor's administrative file. The administrative file is configured using the `pmadm(1M)` and `ttyadm(1M)` commands. When an instance of `ttymon` is invoked by the `sac` command, it starts to monitor its ports. For each port, `ttymon` first initializes the line disciplines, if they are specified, and the speed and terminal settings. For ports with entries in `/etc/logindevperm`, device owner, group and permissions are set. (See `logindevperm(4)`.) The values used for initialization are taken from the appropriate entry in the TTY settings file. This file is maintained by the `sttydefs(1M)` command. Default line disciplines on ports are usually set up by the `autopush(1M)` command of the Autopush Facility.

`ttymon` then writes the prompt and waits for user input. If the user indicates that the speed is inappropriate by pressing the BREAK key, `ttymon` tries the next speed and writes the prompt again. When valid input is received, `ttymon` interprets the per-service configuration file for the port, if one exists, creates a `utmpx` entry if required (see `utmpx(4)`), establishes the service environment, and then invokes the service associated with the port. Valid input consists of a string of at least one non-newline character, terminated by a carriage return. After the service terminates, `ttymon` cleans up the `utmpx` entry, if one exists, and returns the port to its initial state.

If *autobaud* is enabled for a port, `ttymon` will try to determine the baud rate on the port automatically. Users must enter a carriage return before `ttymon` can recognize the baud rate and print the prompt. Currently, the baud rates that can be determined by *autobaud* are `110`, `1200`, `2400`, `4800`, and `9600`.

If a port is configured as a bidirectional port, `ttymon` will allow users to connect to a service, and, if the port is free, will allow `uucico(1M)`, `cu(1C)`, or `ct(1C)` to use it for dialing out. If a port is bidirectional, `ttymon` will wait to read a character before it prints a prompt.

If the *connect-on-carrier* flag is set for a port, `ttymon` will immediately invoke the port's associated service when a connection request is received. The prompt message will not be sent.

If a port is disabled, `ttymon` will not start any service on that port. If a disabled message is specified, `ttymon` will send out the disabled message when a connection request is received. If `ttymon` is disabled, all ports under that instance of `ttymon` will also be disabled.

Service Invocation

The service `ttymon` invokes for a port is specified in the `ttymon` administrative file. `ttymon` will scan the character string giving the service to be invoked for this port, looking for a `%d` or a `%%` two-character sequence. If `%d` is found, `ttymon` will modify the service command to be executed by replacing those two characters by the full path name of this port (the device name). If `%%` is found, they will be replaced by a single `%`. When the service is invoked, file descriptor `0`, `1`, and `2` are opened to the port device for reading and writing. The service is invoked with the user ID, group ID and current home directory set to that of the user name under which the service was registered with `ttymon`. Two environment variables, `HOME` and `TTYPROMPT`, are added to the service's environment by `ttymon`. `HOME` is set to the home directory of the user name under which the service is invoked. `TTYPROMPT` is set to the prompt string configured for the service on the port. This is provided so that a service invoked by `ttymon` has a means of determining if a prompt was actually issued by `ttymon` and, if so, what that prompt actually was.

See `ttyadm(1M)` for options that can be set for ports monitored by `ttymon` under the Service Access Controller.

System Console Invocation

The invocation of ttymon on the system console is managed under `smf(5)` by the service `svc:/system/console-login`. It provides a number of properties within the property group `ttymon` to control the invocation, as follows:

```
NAME                 TYPE              TTYMON OPTION
-------------------------------------------------------------
device               astring           [-d device]
nohangup             boolean           [-h]
label                astring           [-l label]
modules              astring           [-m module1,module2]
prompt               astring           [-p prompt]
timeout              count             [-t timeout]
terminal_type        astring           [-T termtype]
```

If any value is the empty string or an integer set to zero, then the option is not passed to the ttymon invocation. The `-g` option is always specified for this invocation. The `-d` option always defaults to `/dev/console` if it is not set.

See `EXAMPLES`.

Security

`ttymon` uses `pam(3PAM)` for session management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the modules to be used for `ttymon`. Here is a partial `pam.conf` file with entries for `ttymon` using the UNIX session management module.

```
ttymon session  required /usr/lib/security/pam_unix_session.so.1
```

If there are no entries for the `ttymon` service, then the entries for the "other" service will be used.

Options The following options are supported:

`-g` A special invocation of `ttymon` is provided with the `-g` option. This form of the command should only be called by applications that need to set the correct baud rate and terminal settings on a port and then connect to `login` service, but that cannot be pre-configured under the SAC. The following combinations of options can be used with `-g`:

`-d`*device* *device* is the full path name of the port to which `ttymon` is to attach. If this option is not specified, file descriptor `0` must be set up by the invoking process to a TTY port.

`-h` If the -h flag is not set, `ttymon` will force a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed.

`-l`*ttylabel* *ttylabel* is a link to a speed and TTY definition in the `ttydefs` file. This definition tells `ttymon` at what speed to run initially, what the initial TTY settings are, and what speed to try next if the user indicates that the speed is inappropriate by pressing the BREAK key. The default speed is 9600 baud.

`-m`*modules* When initializing the port, `ttymon` will pop all modules on the port, and then push *modules* in the order specified. *modules* is a comma-separated list of pushable modules. Default modules on the ports are usually set up by the Autopush Facility.

`-p`*prompt* Allows the user to specify a prompt string. The default prompt is `Login:`.

`-t`*timeout* Specifies that `ttymon` should exit if no one types anything in *timeout* seconds after the prompt is sent.

`-T`*termtype* Sets the `TERM` environment variable to *termtype*.

Examples **EXAMPLE 1** Setting the Terminal Type

The following example sets the value of the terminal type (`-T`) option for the system console `ttymon` invocation:

```
svccfg -s svc:/system/console-login setprop \
    ttymon/terminal_type = "xterm"
svcadm refresh svc:/system/console-login:default
```

Environment Variables If any of the `LC_*` variables (`LC_CTYPE`, `LC_MESSAGES`, `LC_TIME`, `LC_COLLATE`, `LC_NUMERIC`, and `LC_MONETARY`) (see `environ(5)`) are not set in the environment, the operational behavior of `ttymon` for each corresponding locale category is determined by the value of the `LANG` environment variable. If `LC_ALL` is set, its contents are used to override both the `LANG` and the other `LC_*` variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how `ttymon` behaves.

`LC_CTYPE` Determines how `ttymon` handles characters. When `LC_CTYPE` is set to a valid value, `ttymon` can display and handle text and filenames containing valid characters for that locale. `ttymon` can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. `ttymon` can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

Files `/etc/logindevperm`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Stability	See below.

The command-line syntax is Stable. The SMF properties are Evolving.

See Also `ct(1C)`, `cu(1C)`, `autopush(1M)`, `pmadm(1M)`, `sac(1M)`, `sacadm(1M)`, `sttydefs(1M)`, `ttyadm(1M)`, `uucico(1M)`, `pam(3PAM)`, `logindevperm(4)`, `pam.conf(4)`, `utmpx(4)`, `attributes(5)`, `environ(5)`, `pam_authtok_check(5)`, `pam_authtok_get(5)`, `pam_authtok_store(5)`, `pam_dhkeys(5)`, `pam_passwd_auth(5)`, `pam_unix_account(5)`, `pam_unix_auth(5)`, `pam_unix_session(5)`, `smf(5)`

System Administration Guide: Basic Administration

Notes If a port is monitored by more than one `ttymon`, it is possible for the `ttymons` to send out prompt messages in such a way that they compete for input.

The `pam_unix`(5) module is no longer supported. Similar functionality is provided by `pam_authtok_check(5)`, `pam_authtok_get(5)`, `pam_authtok_store(5)`, `pam_dhkeys(5)`, `pam_passwd_auth(5)`, `pam_unix_account(5)`, `pam_unix_auth(5)`, and `pam_unix_session(5)`.

Name tunefs – tune an existing UFS file system

Synopsis

```
tunefs [-a maxcontig] [-d rotdelay] [-e maxbpg]
     [-m minfree] [-o space | time] special | filesystem
```

Description `tunefs` is designed to change the dynamic parameters of a file system that affect the layout policies. When using `tunefs` with *filesystem*, *filesystem* must be in `/etc/vfstab`. The parameters that can be changed are indicated by the options given below.

Options The following options are supported:

`-a` *maxcontig* The maximum number of logical blocks, belonging to one file, that is allocated contiguously. The default is calculated as follows:

`maxcontig =` *disk drive maximum transfer size / disk block size*

If the disk drive's maximum transfer size cannot be determined, the default value for `maxcontig` is calculated from kernel parameters as follows:

If `maxphys` is less than `ufs_maxmaxphys`, which is 1 Mbyte, then `maxcontig` is set to `maxphys`. Otherwise, `maxcontig` is set to `ufs_maxmaxphys`.

You can set `maxcontig` to any positive integer value.

The actual value will be the lesser of what has been specified and what the hardware supports.

`-d` *rotdelay* This parameter is obsolete as of the Solaris 10 release. The value is always set to `0`, regardless of the input value.

`-e` *maxbpg* Indicates the maximum number of contiguous logical blocks any single file can allocate from a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to approximately one quarter of the total contiguous logical blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group.

The effect of this limit is to cause big files to do long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

`-m` *minfree* Specifies the minimum free space threshold, or the percentage of space held back from normal users. This value can be set to `0`. However, up to a factor of three in throughput will be lost over the performance obtained at a 10% threshold. *Note:* If the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

-o *space* | *time* The file system can either be instructed to try to minimize the *time* spent allocating blocks, or to try to minimize the *space* fragmentation on the disk. The default is *time*.

Generally, you should optimize for time unless the file system is over 90% full.

Usage See `largefile(5)` for the description of the behavior of `tunefs` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `mkfs_ufs(1M)`, `newfs(1M)`, `attributes(5)`, `largefile(5)`

Name txzonemgr – Trusted Extensions Zone Manager Configuration Utility

Synopsis

```
/usr/sbin/txzonemgr
```

Description The `txzonemgr` shell script provides a simple menu-based GUI wizard for creating, installing, initializing, and booting labeled zones on a system where Trusted Extensions is enabled. The script provides contextual menus with appropriate choices.

`txzonemgr` is run by roles granted in the Zone Management Rights Profile, or by root in the global zone. It takes no options and returns no values.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Command Interface Stability	Committed
Interactive Dialogue	Not an Interface

See Also `zenity`(1), `ifconfig(1M)`, `zoneadm(1M)`, `zonecfg(1M)`, `attributes(5)`, `rbac(5)`, `zones(5)`

Solaris Trusted Extensions Administrator's Procedures

Notes If administering zones from JDS, use `txzonemgr` rather than CDE actions.

`txzonemgr` uses the `zenity` command. For details, see the `zenity`(1) man page.

Name tzreload – notify timezone update

Synopsis

```
/usr/sbin/tzreload [-a]
```

Description The `tzreload` command notifies active (running) processes to reread timezone information. The timezone information is cached in each process, absent a `tzreload` command, is never reread until a process is restarted. In response to a `tzreload` command, active processes reread the current timezone information at the next call to `ctime(3C)` and `mktime(3C)`. By default, the `tzreload` notification is sent to the processes within the current zone.

In addition to notifying active processes, the `tzreload` command also notifies cron(1M), to reinitialize the job scheduler with the new timezone information.

Options The following options are supported:

`-a`
Notify processes in all zones.

Files `/usr/share/lib/zoneinfo`
Standard zone information directory.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcs
Interface Stability	Committed

See Also `cron(1M)`, `zdump(1M)`, `zic(1M)`, `zoneadm(1M)`, `ctime(3C)`, `mktime(3C)`, `attributes(5)`

`rpc.cmsd(1M)`, part of the CDE man page set.

Notes Although `tzreload` reinitializes `cron(1M)`, applications that are affected by timezone changes still need to be restarted or reinitialized if they do not reread the new timezone information before timezone changes take place.

Name tzselect – select a time zone

Synopsis

```
/usr/bin/tzselect
```

Description The `tzselect` program asks you a series of questions about the current location and outputs the resulting time zone description to standard output. The output is suitable as a value for the `TZ` environment variable.

All user interaction is through standard input and standard error.

Options The `tzselect` command has no options.

Exit Status The following exit values are returned:

`0` Timezone information was successfully obtained.

`>0` An error occurred.

Files `/usr/share/lib/zoneinfo` directory containing timezone data files

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Stability	Evolving

See Also `zdump(1M)`, `zic(1M)`, `ctime(3C)`, `attributes(5)`

Name uadmin – administrative control

Synopsis

```
/usr/sbin/uadmin cmd fcn [mdep]

/sbin/uadmin cmd fcn [mdep]
```

Description The `uadmin` command provides control for basic administrative functions. This command is tightly coupled to the system administration procedures and is not intended for general use. It may be invoked only by the super-user.

Both the *cmd* (command) and *fcn* (function) arguments are converted to integers and passed to the `uadmin` system call. The optional *mdep* (machine dependent) argument is only available for the *cmd* values of `1` (`A_REBOOT`), `2` (`A_SHUTDOWN`), or `5` (`A_DUMP`). For any other *cmd* value, no *mdep* command-line argument is allowed.

When passing an *mdep* value that contains whitespaces, the string must be grouped together as a single argument enclosed within quotes, for example:

```
uadmin 1 1 “-s kernel/unix”
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `halt(1M)`, `reboot(1M)`, `uadmin(2)`, `attributes(5)`

Warnings Shutting down or halting the system by means of `uadmin` does not update the boot archive. Avoid using this command after

- editing of files such as `/etc/system`
- installing new driver binaries or kernel binaries
- updating existing driver binaries or kernel binaries.

Use `reboot(1M)` or `halt(1M)` instead.

Name ucodeadm – update processor microcode

Synopsis

```
/usr/sbin/ucodeadm -v

/usr/sbin/ucodeadm -umicrocode-text-file

/usr/sbin/ucodeadm -i [-R path] microcode-text-file
```

Description The `ucodeadm` utility can be used to report running microcode revision on the processors, update microcode, or install microcode on the target system to be used during the boot process.

The *microcode-text-file* can be obtained from processor vendors.

Options

`-v` Report microcode revision.

`-u` *microcode-text-file* Update microcode on all cross-call interrupt ready processors.

`-i` *microcode-text-file* Install microcode files on target system to be used during the next boot cycle. The text file name must have the vendor name prefix, such as "`intel`" or "`amd`".

By default the microcode files will be installed at:

```
/platform/i86pc/ucode/$VENDORSTR/
```

where `VENDORSTR` is either "`GenuineIntel`" or "`AuthenticAMD`".

`-R` *alternate path* Install *microcode* path in the *alternate path*.

Examples

EXAMPLE 1 Reporting the Microcode Revision

The following example displays the microcode revision that is currently running:

```
# ucodeadm -v
```

EXAMPLE 2 Updating the Processor Microcode

The following example updates the processor microcode to `intel-ucode.txt`:

```
# ucodeadm -u intel-ucode.txt
```

EXAMPLE 3 Installing the Microcode on the Target System

The following example installs the microcode on the target system, `/export/ucode-path`:

```
# ucodeadm -i -R /export/ucode-path intel-ucode.txt
```

If an alternate path is used when installing the microcode on the target system, the installed microcode file is not used on the next boot cycle.

Exit Status The following exit values are returned:

0 Successful completion.

>0 An error occurred.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Stable

See Also psradm(1M), psrinfo(1M), attributes(5)

Name ufsdump – incremental file system dump

Synopsis `/usr/sbin/ufsdump [`*options*`] [`*arguments*`]` *files_to_dump*

Description `ufsdump` backs up all files specified by *files_to_dump* (usually either a whole file system or files within a file sytem changed after a certain date) to magnetic tape, diskette, or disk file.

The `ufsdump` command can only be used on unmounted file systems, or those mounted read-only. Attempting to dump a mounted, read-write file system might result in a system disruption or the inability to restore files from the dump. Consider using the `fssnap(1M)` command to create a file system snapshot if you need a point-in-time image of a file system that is mounted.

If a filesystem was mounted with the logging option, it is strongly recommended that you run `ufsdump` as the root user. Running the command as a non-root user might result in the creation of an inconsistent dump.

options is a single string of one-letter `ufsdump` options.

arguments may be multiple strings whose association with the options is determined by order. That is, the first argument goes with the first option that takes an argument; the second argument goes with the second option that takes an argument, and so on.

files_to_dump is required and must be the last argument on the command line. See `OPERANDS` for more information.

With most devices `ufsdump` can automatically detect the end-of-media. Consequently, the `d`, `s`, and `t` options are not necessary for multi-volume dumps, unless `ufsdump` does not understand the way the device detects the end-of-media, or the files are to be restored on a system with an older version of the `restore` command.

Options The following options are supported:

`0`–`9`
The "dump level." All files specified by *files_to_dump* that have been modified since the last `ufsdump` at a lower dump level are copied to the *dump_file* destination (normally a magnetic tape device). For instance, if a "level `2`" dump was done on Monday, followed by a "level `4`" dump on Tuesday, a subsequent "level `3`" dump on Wednesday would contain all files modified or added since the "level `2`" (Monday) backup. A "level `0`" dump copies the entire file system to the *dump_file*.

`a` *archive_file*
Archive file. Archive a dump table-of-contents in the specified *archive_file* to be used by `ufsrestore(1M)` to determine whether a file is in the dump file that is being restored.

`b` *factor*
Blocking factor. Specify the blocking factor for tape writes. The default is 20 blocks per write for tapes of density less than 6250BPI (bytes-per-inch). The default blocking factor for tapes of density 6250BPI and greater is 64. The default blocking factor for cartridge

tapes (c option) is 126. The highest blocking factor available with most tape drives is 126. Note: the blocking factor is specified in terms of 512-byte blocks, for compatibility with tar(1).

c

Cartridge. Set the defaults for cartridge instead of the standard half-inch reel. This sets the density to 1000BPI and the blocking factor to 126. Since ufsdump can automatically detect the end-of-media, only the blocking parameter normally has an effect. When cartridge tapes are used, and this option is *not* specified, ufsdump will slightly miscompute the size of the tape. If the b, d, s or t options are specified with this option, their values will override the defaults set by this option.

d *bpi*

Tape density. Not normally required, as ufsdump can detect end-of-media. This parameter can be used to keep a running tab on the amount of tape used per reel. The default density is 6250BPI except when the c option is used for cartridge tape, in which case it is assumed to be 1000BPI per track. Typical values for tape devices are:

1/2 inch tape
6250 BPI

1/4 inch cartridge
1000 BPI The tape densities and other options are documented in the st(7D) man page.

D

Diskette. Dump to diskette.

f *dump_file*

Dump file. Use *dump_file* as the file to dump to, instead of /dev/rmt/0. If *dump_file* is specified as –, dump to standard output.

If the name of the file is of the form *machine*:*device,* the dump is done from the specified machine over the network using rmt(1M). Since ufsdump is normally run by root, the name of the local machine must appear in the /.rhosts file of the remote machine. If the file is specified as *user*@*machine*:*device,* ufsdump will attempt to execute as the specified user on the remote machine. The specified user must have a .rhosts file on the remote machine that allows the user invoking the command from the local machine to access the remote machine.

l

Autoload. When the end-of-tape is reached before the dump is complete, take the drive offline and wait up to two minutes for the tape drive to be ready again. This gives autoloading (stackloader) tape drives a chance to load a new tape. If the drive is ready within two minutes, continue. If it is not, prompt for another tape and wait.

L *string*

Sets the tape label to *string*, instead of the default `none`. *string* may be no more than sixteen characters long. If it is longer, it is truncated and a warning printed; the dump will still be done. The tape label is specific to the `ufsdump` tape format, and bears no resemblance to IBM or ANSI-standard tape labels.

n

Notify all operators in the `sys` group that `ufsdump` requires attention by sending messages to their terminals, in a manner similar to that used by the `wall(1M)` command. Otherwise, such messages are sent only to the terminals (such as the console) on which the user running `ufsdump` is logged in.

N *device_name*

Use *device_name* when recording information in `/etc/dumpdates` (see the `u` option) and when comparing against information in `/etc/dumpdates` for incremental dumps. The *device_name* provided can contain no white space as defined in `scanf(3C)` and is case-sensitive.

o

Offline. Take the drive offline when the dump is complete or the end-of-media is reached and rewind the tape, or eject the diskette. In the case of some autoloading 8mm drives, the tape is removed from the drive automatically. This prevents another process which rushes in to use the drive, from inadvertently overwriting the media.

s *size*

Specify the *size* of the volume being dumped to. Not normally required, as `ufsdump` can detect end-of-media. When the specified size is reached, `ufsdump` waits for you to change the volume. `ufsdump` interprets the specified size as the length in feet for tapes and cartridges, and as the number of 1024-byte blocks for diskettes. The values should be a little smaller than the actual physical size of the media (for example, 425 for a 450-foot cartridge). Typical values for tape devices depend on the `c` option, for cartridge devices, and the `D` option for diskettes:

1/2 inch tape
: 2300 feet

60-Mbyte 1/4 inch cartridge
: 425 feet

150-Mbyte 1/4 inch cartridge
: 700 feet

diskette
: 1422 blocks (Corresponds to a 1.44-Mbyte diskette, with one cylinder reserved for bad block information.)

S
Size estimate. Determine the amount of space that is needed to perform the dump without actually doing it, and display the estimated number of bytes it will take. This is useful with incremental dumps to determine how many volumes of media will be needed.

t *tracks*
Specify the number of tracks for a cartridge tape. Not normally required, as `ufsdump` can detect end-of-media. The default is 9 tracks. The `t` option is not compatible with the `D` option. Values for Sun-supported tape devices are:

60-Mbyte 1/4 inch cartridge
9 tracks

150-Mbyte 1/4 inch cartridge
18 tracks

T *time_wait*[hms]
Sets the amount of time to wait for an `autoload` command to complete. This option is ignored unless the `l` option has also been specified. The default time period to wait is two minutes. Specify time units with a trailing `h` (for hours), `m` (for minutes), or `s` (for seconds). The default unit is minutes.

u
Update the dump record. Add an entry to the file `/etc/dumpdates`, for each file system successfully dumped that includes the file system name (or *device_name* as specified with the `N` option), date, and dump level.

v
Verify. After each tape or diskette is written, verify the contents of the media against the source file system. If any discrepancies occur, prompt for new media, then repeat the dump/verification process. The file system *must* be unmounted. This option cannot be used to verify a dump to standard output.

w
Warning. List the file systems that have not been backed up within a day. This information is gleaned from the files `/etc/dumpdates` and `/etc/vfstab`. When the `w` option is used, all other options are ignored. After reporting, `ufsdump` exits immediately.

W
Warning with highlight. Similar to the `w` option, except that the `W` option includes all file systems that appear in `/etc/dumpdates`, along with information about their most recent dump dates and levels. File systems that have not been backed up within a day are highlighted.

Operands The following operand is supported:

files_to_dump
Specifies the files to dump. Usually it identifies a whole file system by its raw device name (for example, `/dev/rdsk/c0t3d0s6`). Incremental dumps (levels `1` to `9`) of files changed

after a certain date only apply to a whole file system. Alternatively, *files_to_dump* can identify individual files or directories. All named directories that may be examined by the user running `ufsdump`, as well as any explicitly-named files, are dumped. This dump is equivalent to a level `0` dump of the indicated portions of the filesystem, except that `/etc/dumpdates` is not updated even if the `-u` option has been specified. In all cases, the files must be contained in the same file system, and the file system must be local to the system where `ufsdump` is being run.

files_to_dump is required and must be the last argument on the command line.

If no *options* are given, the default is `9uf /dev/rmt/0` *files_to_dump*.

Usage See `largefile(5)` for the description of the behavior of `ufsdump` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Examples **EXAMPLE 1** Using `ufsdump`

The following command makes a full dump of a root file system on `c0t3d0`, on a 150-MByte cartridge tape unit `0`:

```
example# ufsdump 0cfu /dev/rmt/0 /dev/rdsk/c0t3d0s0
```

The following command makes and verifies an incremental dump at level `5` of the `usr` partition of `c0t3d0`, on a 1/2 inch reel tape unit `1,`:

```
example# ufsdump 5fuv /dev/rmt/1 /dev/rdsk/c0t3d0s6
```

Exit Status While running, `ufsdump` emits many verbose messages. `ufsdump` returns the following exit values:

`0`
Normal exit.

`1`
Startup errors encountered.

`3`
Abort – no checkpoint attempted.

Files `/dev/rmt/0`
default unit to dump to

`/etc/dumpdates`
dump date record

`/etc/group`
to find group `sys`

`/etc/hosts`
to gain access to remote system with drive

`/etc/vfstab`
list of file systems

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `cpio(1)`, `tar(1)`, `dd(1M)`, `devnm(1M)`, `fssnap(1M)`, `prtvtoc(1M)`, `rmt(1M)`, `shutdown(1M)`, `ufsrestore(1M)`, `volcopy(1M)`, `wall(1M)`, `scanf(3C)`, `ufsdump(4)`, `attributes(5)`, `largefile(5)`, `st(7D)`

Notes

Read Errors Fewer than 32 read errors on the file system are ignored.

Process Per Reel Because each reel requires a new process, parent processes for reels that are already written hang around until the entire tape is written.

Operator Intervention `ufsdump` requires operator intervention on these conditions: end of volume, end of dump, volume write error, volume open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the `n` option, `ufsdump` interacts with the operator on `ufsdump`'s control terminal at times when `ufsdump` can no longer proceed, or if something is grossly wrong. All questions `ufsdump` poses *must* be answered by typing `yes` or `no`, as appropriate.

Since backing up a disk can involve a lot of time and effort, `ufsdump` checkpoints at the start of each volume. If writing that volume fails for some reason, `ufsdump` will, with operator permission, restart itself from the checkpoint after a defective volume has been replaced.

Suggested Dump Schedule It is vital to perform full, "level `0`", dumps at regular intervals. When performing a full dump, bring the machine down to single-user mode using `shutdown(1M)`. While preparing for a full dump, it is a good idea to clean the tape drive and heads. Incremental dumps should be performed with the system running in single-user mode.

Incremental dumps allow for convenient backup and recovery of active files on a more frequent basis, with a minimum of media and time. However, there are some tradeoffs. First, the interval between backups should be kept to a minimum (once a day at least). To guard against data loss as a result of a media failure (a rare, but possible occurrence), capture active files on (at least) two sets of dump volumes. Another consideration is the desire to keep unnecessary duplication of files to a minimum to save both operator time and media storage. A third consideration is the ease with which a particular backed-up version of a file can be located and restored. The following four-week schedule offers a reasonable tradeoff between these goals.

```
           Sun     Mon     Tue     Wed     Thu     Fri
Week 1:    Full    5       5       5       5       3
Week 2:            5       5       5       5       3
Week 3:            5       5       5       5       3
Week 4:            5       5       5       5       3
```

Although the Tuesday through Friday incrementals contain “extra copies” of files from Monday, this scheme assures that any file modified during the week can be recovered from the previous day's incremental dump.

Process Priority of ufsdump

`ufsdump` uses multiple processes to allow it to read from the disk and write to the media concurrently. Due to the way it synchronizes between these processes, any attempt to run dump with a `nice` (process priority) of ‘–5' or better will likely make `ufsdump` run *slower* instead of faster.

Overlapping Partitions

Most disks contain one or more overlapping slices because slice 2 covers the entire disk. The other slices are of various sizes and usually do not overlap. For example, a common configuration places `root` on slice 0, `swap` on slice 1, `/opt` on slice 5 and `/usr` on slice 6.

It should be emphasized that `ufsdump` dumps one `ufs` file system at a time. Given the above scenario where slice 0 and slice 2 have the same starting offset, executing `ufsdump` on slice 2 with the intent of dumping the entire disk would instead dump only the `root` file system on slice 0. To dump the entire disk, the user must dump the file systems on each slice separately.

Bugs

The `/etc/vfstab` file does not allow the desired frequency of backup for file systems to be specified (as `/etc/fstab` did). Consequently, the `w` and `W` options assume file systems should be backed up daily, which limits the usefulness of these options.

Name ufsrestore – incremental file system restore

Synopsis

```
/usr/sbin/ufsrestore i | r | R | t | x [abcdfhlmostvyLT]
     [archive_file] [factor] [dumpfile] [n] [label]
     [timeout] [filename]...
```

Description The `ufsrestore` utility restores files from backup media created with the `ufsdump` command. `ufsrestores`'s actions are controlled by the *key* argument. The *key* is exactly one `function` *letter* (`i`, `r`, `R`, `t`, or `x`) and zero or more *function modifiers* (letters). The *key* string contains no SPACE characters. Function modifier arguments are listed on the command line in the same order as their corresponding function modifiers appear in the *key* string.

filename arguments which appear on the command line, or as arguments to an interactive command, are treated as shell `glob` patterns by the `x` and `t` functions; any files or directories matching the patterns are selected. The metacharacters `*`, `?`, and `[ ]` must be protected from the shell if they appear on the command line. There is no way to quote these metacharacters to explicitly match them in a filename.

The temporary files `rstdir*` and `rstmode*` are placed in `/tmp` by default. If the environment variable `TMPDIR` is defined with a non-empty value, that location is used instead of `/tmp`.

Options

Function Letters You must specify one (and only one) of the function letters listed below. Note that `i`, `x`, and `r` are intended to restore files into an empty directory. The `R` function is intended for restoring into a populated directory.

`i` Interactive. After reading in the directory information from the media, `ufsrestore` invokes a shell-like interface that allows you to browse through the dump file's directory hierarchy and select individual files to be extracted. Restoration has the same semantics as `x` (see below). See `Interactive Commands`, below, for a description of available commands.

`r` Recursive. Starting with an empty directory and a level 0 dump, the `r` function recreates the filesystem relative to the current working directory, exactly as it appeared when the dump was made. Information used to restore incremental dumps on top of the full dump (for example, `restoresymtable`) is also included. Several `ufsrestore` runs are typical, one for each higher level of dump (0, 1, ..., 9). Files that were deleted between the level 0 and a subsequent incremental dump will not exist after the final restore. To completely restore a file system, use the `r` function restore the level 0 dump, and again for each incremental dump. Although this function letter is intended for a complete restore onto a new file system (one just created with `newfs(1M)`), if the file system contains files not on the backup media, they are preserved.

`R` Resume restoring. If an `r`-mode `ufsrestore` was interrupted, this function prompts for the volume from which to resume restoring and continues the restoration from where it was left off. Otherwise identical to `r`.

t Table of contents. List each *filename* that appears on the media. If no *filename* argument is given, the root directory is listed. This results in a list of all files on the media, unless the `h` function modifier is in effect. The table of contents is taken from the media or from the specified archive file, when the `a` function modifier is used. The `a` function modifier is mutually exclusive with the `x` and `r` function letters.

x Extract the named files from the media. Files are restored to the same relative locations that they had in the original file system.

If the *filename* argument matches a directory whose contents were written onto the media, and the `h` modifier is not in effect, the directory is recursively extracted, relative to the current directory, which is expected to be empty. For each file, the owner, modification time, and mode are restored (if possible).

If you omit the *filename* argument or specify `.`, the root directory is extracted. This results in the entire tape being extracted, unless the `h` modifier is in effect. . With the `x` function, existing files are overwritten and `ufsrestore` displays the names of the overwritten files. Overwriting a currently-running executable can have unfortunate consequences.

Use the `x` option to restore partial file system dumps, as they are (by definition) not entire file systems.

Function Modifiers

a *archive_file* Read the table of contents from *archive_file* instead of the media. This function modifier can be used in combination with the `t`, `i`, or `x` function letters, making it possible to check whether files are on the media without having to mount the media. When used with the `x` and interactive (`i`) function letters, it prompts for the volume containing the file(s) before extracting them.

b *factor* Blocking factor. Specify the blocking factor for tape reads. For variable length SCSI tape devices, unless the data was written with the default blocking factor, a blocking factor at least as great as that used to write the tape must be used; otherwise, an error will be generated. Note that a tape block is 512 bytes. Refer to the man page for your specific tape driver for the maximum blocking factor.

c Convert the contents of the media in 4.1BSD format to the new `ufs` file system format.

d Debug. Turn on debugging output.

f *dump_file* Use *dump_file* instead of `/dev/rmt/0` as the file to restore from. Typically *dump_file* specifies a tape or diskette drive. If *dump_file* is specified as '–', `ufsrestore` reads from the standard input. This allows `ufsdump(1M)` and `ufsrestore` to be used in a pipeline to copy a file system:

```
example# ufsdump 0f - /dev/rdsk/c0t0d0s7 \
 | (cd /home;ufsrestore xf -)
```

If the name of the file is of the form *machine*:*device*, the restore is done from the specified machine over the network using `rmt(1M)`. Since `ufsrestore` is normally run by root, the name of the local machine must appear in the `/.rhosts` file of the remote machine. If the file is specified as *user*`@`*machine*:*device*, `ufsrestore` will attempt to execute as the specified user on the remote machine. The specified user must have a `.rhosts` file on the remote machine that allows the user invoking the command from the local machine to access the remote machine.

h — Extract or list the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.

l — Autoload. When the end-of-tape is reached before the restore is complete, take the drive off-line and wait up to two minutes (the default, see the `T` function modifier) for the tape drive to be ready again. This gives autoloading (stackloader) tape drives a chance to load a new tape. If the drive is ready within two minutes, continue. If it is not, prompt for another tape and wait.

L *label* — The label that should appear in the header of the dump file. If the labels do not match, `ufsrestore` issues a diagnostic and exits. The tape label is specific to the `ufsdump` tape format, and bears no resemblance to IBM or ANSI-standard tape labels.

m — Extract by inode numbers rather than by filename to avoid regenerating complete pathnames. Regardless of where the files are located in the dump hierarchy, they are restored into the current directory and renamed with their inode number. This is useful if only a few files are being extracted.

o — Offline. Take the drive off-line when the restore is complete or the end-of-media is reached and rewind the tape, or eject the diskette. In the case of some autoloading 8mm drives, the tape is removed from the drive automatically.

s *n* — Skip to the *n*th file when there are multiple dump files on the same tape. For example, the command:

```
example# ufsrestore xfs /dev/rmt/0hn 5
```

would position you to the fifth file on the tape when reading volume 1 of the dump. If a dump extends over more than one volume, all volumes except the first are assumed to start at position 0, no matter what "`s` *n*" value is specified.

If "`s` *n*" is specified, the backup media must be at BOT (beginning of tape). Otherwise, the initial positioning to read the table of contents will fail, as it is performed by skipping the tape forward *n*`-1` files rather than by using absolute positioning. This is because on some devices absolute positioning is very time consuming.

T timeout [hms] — Sets the amount of time to wait for an autoload command to complete. This function modifier is ignored unless the `l` function modifier has also been specified. The default timeout period is two minutes. The time units may be specified as a trailing `h` (hours), `m` (minutes), or `s` (seconds). The default unit is minutes.

v — Verbose. `ufsrestore` displays the name and inode number of each file it restores, preceded by its file type.

y — Do not ask whether to abort the restore in the event of tape errors. `ufsrestore` tries to skip over the bad tape block(s) and continue as best it can.

Interactive Commands

`ufsrestore` enters interactive mode when invoked with the `i` function letters. Interactive commands are reminiscent of the shell. For those commands that accept an argument, the default is the current directory. The interactive options are:

add [*filename*] — Add the named file or directory to the list of files to extract. If a directory is specified, add that directory and its files (recursively) to the extraction list (unless the `h` modifier is in effect).

cd *directory* — Change to *directory* (within the dump file).

delete [*filename*] — Delete the current directory, or the named file or directory from the list of files to extract. If a directory is specified, delete that directory and all its descendents from the extraction list (unless the `h` modifier is in effect). The most expedient way to extract a majority of files from a directory is to add that directory to the extraction list, and then delete specific files to omit.

extract — Extract all files on the extraction list from the dump media. `ufsrestore` asks which volume the user wishes to mount. The fastest way to extract a small number of files is to start with the last volume and work toward the first. If "`s` *n*" is given on the command line, volume 1 will automatically be positioned to file *n* when it is read.

help — Display a summary of the available commands.

ls [*directory*] — List files in *directory* or the current directory, represented by a '`.`' (period). Directories are appended with a '`/`' (slash). Entries marked for extraction are prefixed with a '`*`' (asterisk). If the verbose option is in effect, inode numbers are also listed.

marked [*directory*]	Like `ls`, except only files marked for extraction are listed.
pager	Toggle the pagination of the output from the `ls` and `marked` commands. The pager used is that defined by the `PAGER` environment variable, or more(1) if that envar is not defined. The `PAGER` envar may include white-space-separated arguments for the pagination program.
pwd	Print the full pathname of the current working directory.
quit	`ufsrestore` exits immediately, even if the extraction list is not empty.
setmodes	Prompts: `set owner/mode for '.'` (period). Type `y` for yes to set the mode (permissions, owner, times) of the current directory '.' (period) into which files are being restored equal to the mode of the root directory of the file system from which they were dumped. Normally, this is what you want when restoring a whole file system, or restoring individual files into the same locations from which they were dumped. Type `n` for no, to leave the mode of the current directory unchanged. Normally, this is what you want when restoring part of a dump to a directory other than the one from which the files were dumped.
setpager *command*	Sets the command to use for paginating output instead of the default or that inherited from the environment. The *command* string may include arguments in addition to the command itself.
verbose	Toggle the status of the `v` modifier. While `v` is in effect, the `ls` command lists the inode numbers of all entries, and `ufsrestore` displays information about each file as it is extracted.
what	Display the dump header on the media.

Operands The following operands are supported.

filename Specifies the pathname of files (or directories) to be restored to disk. Unless the `h` function modifier is also used, a directory name refers to the files it contains, and (recursively) its subdirectories and the files they contain. *filename* is associated with either the `x` or `t` function letters, and must come last.

Usage See `largefile(5)` for the description of the behavior of `ufsrestore` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred. Verbose messages are displayed.

Environment Variables PAGER The command to use as a filter for paginating output. This can also be used to specify the options to be used. Default is `more(1)`.

TMPDIR Selects the directory for temporary files. Defaults to `/tmp` if not defined in the environment.

Files

`/dev/rmt/0`	the default tape drive
`$TMPDIR/rstdir*`	file containing directories on the tape
`$TMPDIR/rstmode*`	owner, mode, and timestamps for directories
`./restoresymtable`	information passed between incremental restores

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `more(1)`, `mkfs(1M)`, `mount(1M)`, `rmt(1M)`, `ufsdump(1M)`, `ufsdump(4)`, `attributes(5)`, `largefile(5)`

Diagnostics `ufsrestore` complains about bad option characters.

Read errors result in complaints. If `y` has been specified, or the user responds `y`, `ufsrestore` will attempt to continue.

If the dump extends over more than one tape, `ufsrestore` asks the user to change tapes. If the `x` or `i` function letter has been specified, `ufsrestore` also asks which volume the user wishes to mount. If the `s` modifier has been specified, and volume 1 is mounted, it is automatically positioned to the indicated file.

There are numerous consistency checks that can be listed by `ufsrestore`. Most checks are self-explanatory or can "never happen". Common errors are given below.

`Converting to new file system format`
A dump tape created from the old file system has been loaded. It is automatically converted to the new file system format.

filename`: not found on tape`
The specified file name was listed in the tape directory, but was not found on the tape. This is caused by tape read errors while looking for the file, using a dump tape created on an active file system, or restoring a partial dump with the `r` function.

`expected next file` *inumber*`, got` *inumber*
A file that was not listed in the directory showed up. This can occur when using a dump tape created on an active file system.

`Incremental tape too low`
When doing an incremental restore, a tape that was written before the previous incremental tape, or that has too low an incremental level has been loaded.

`Incremental tape too high`
When doing incremental restore, a tape that does not begin its coverage where the previous incremental tape left off, or one that has too high an incremental level has been loaded.

`media read error: invalid argument`
Blocking factor specified for read is smaller than the blocking factor used to write data.

`Tape read error while restoring`
`Tape read error while skipping over inode inumber`
`Tape read error while trying to resynchronize`
`A tape read error has occurred`
If a file name is specified, then its contents are probably partially wrong. If an inode is being skipped or the tape is trying to resynchronize, then no extracted files have been corrupted, though files may not be found on the tape.

`resync ufsrestore, skipped` *num*
After a tape read error, `ufsrestore` may have to resynchronize itself. This message lists the number of blocks that were skipped over.

`Incorrect tape label. Expected 'foo', got 'bar'.`
The `L` option was specified, and its value did not match what was recorded in the header of the dump file.

Notes `ufsrestore` can get confused when doing incremental restores from dump tapes that were made on active file systems.

A `level 0` dump must be done after a full restore. Because `ufsrestore` runs in user mode, it has no control over inode allocation. This means that `ufsrestore` repositions the files, although it does not change their contents. Thus, a full dump must be done to get a new set of directories reflecting the new file positions, so that later incremental dumps will be correct.

Name unshare – make local resource unavailable for mounting by remote systems

Synopsis

```
unshare [-F FSType] [-o specific_options]
      [pathname | resourcename]
```

Description The `unshare` command makes a shared local resource unavailable as file system type *FSType*. If the option `-F`*FSType* is omitted, then the first file system type listed in file `/etc/dfs/fstypes` will be used as the default. *Specific_options*, as well as the semantics of *resourcename*, are specific to particular distributed file systems.

Options

`-F` *FSType* Specify the file system type.

`-o` *specific_options* Specify options specific to the file system provided by the `-F` option.

Files

`/etc/dfs/fstypes`

`/etc/dfs/sharetab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `share(1M)`, `shareall(1M)`, `attributes(5)`

Notes If *pathname* or *resourcename* is not found in the shared information, an error message will be sent to standard error.

When an `unshare` command completes successfully, a client mounting a file system specified in that `unshare` command no longer has access to that file system.

Name unshare_nfs – make local NFS file systems unavailable for mounting by remote systems

Synopsis

```
unshare [-F nfs] pathname
```

Description The `unshare` command makes local file systems unavailable for mounting by remote systems. The shared file system must correspond to a line with NFS as the *FSType* in the file `/etc/dfs/sharetab`.

Options The following options are supported:

`-F` This option may be omitted if NFS is the first file system type listed in the file `/etc/dfs/fstypes`.

Files

`/etc/dfs/fstypes`

`/etc/dfs/sharetab`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnfssu

See Also `nfsd(1M)`, `share(1M)`, `attributes(5)`

Notes If the file system being unshared is a symbolic link to a valid pathname, the canonical path (the path which the symbolic link follows) will be unshared.

For example, if `/export/foo` is a symbolic link to `/export/bar` (`/export/foo -> /export/bar`), the following `unshare` command will result in `/export/bar` as the unshared pathname (and not `/export/foo`):

```
example#  unshare -F nfs /export/foo
```

For file systems that are accessed by NFS Version 4 clients, once the unshare is complete, all NFS Version 4 state (open files and file locks) are released and unrecoverable by the clients. If the intent is to share the file system after some administrative action, the NFS daemon (`nfsd`) should first be stopped and then the file system unshared. After the administrative action is complete, the file system would then be shared and the NFS daemon restarted. See `nfsd(1M)`

Name update_drv – modify device driver attributes

Synopsis

```
update_drv [-f | -v] driver_module

update_drv [-b basedir] [-f | -v] -a [-m 'permission']
     [-i 'identify-name'] [-P 'privilege'] [-p 'policy'] driver_module

update_drv [-b basedir] [-f | -v] -d [-m 'permission']
     [-i 'identify-name'] [-P 'privilege'] [-p 'policy'] driver_module
```

Description The `update_drv` command informs the system about attribute changes to an installed device driver. It can be used to re-read the `driver.conf(4)` file, or to add, modify, or delete a driver's minor node permissions or aliases.

Without options, `update_drv` reloads the `driver.conf` file.

Upon successfully updating the aliases, the driver binding takes effect upon reconfig boot or hotplug of the device.

Upon successfully updating the permissions, only the new driver minor nodes get created with the modified set of file permissions. Existing driver minor nodes do not get modified.

Options The following options are supported:

`-a` Add a *permission*, *aliases*, *privilege* or *policy* entry.

With the `-a` option specified, a permission entry (using the `-m` option), or a driver's aliases entry (using the `-i` option), a device privilege (using the `-P` option) or a a device policy (using the `-p` option), can be added or updated. If a matching minor node permissions entry is encountered (having the same driver name and the minor node), it is replaced. If a matching aliases entry is encountered (having a different driver name and the same alias), an error is reported.

The `-a` and `-d` options are mutually exclusive.

`-b` *basedir* Installs or modifies the driver on the system with a root directory of basedir rather than installing on the system executing `update_drv`.

Note – The root file system of any non-global zones must not be referenced with the `-b` option. Doing so might damage the global zone's file system, might compromise the security of the global zone, and might damage the non-global zone's file system. See `zones(5)`.

`-d` Delete a *permission*, *aliases*, *privilege* or *policy* entry.

The `-m` *permission*, `-i` *identify-name*, `-P` *privilege* or the `-p` *policy* option needs to be specified with the `-d` option.

The `-d` and `-a` options are mutually exclusive.

	If the entry doesn't exist `update_drv` returns an error.
`-f`	Force the system to reread the `driver.conf` file even if the driver module cannot be unloaded. See `NOTES` section for details.
`-i` '*identify-name*'	A white-space separated list of aliases for the driver. If `-a` or `-d` option is not specified then this option is ignored. The *identify-name* string is mandatory. If all aliases need to be removed, `rem_drv(1M)` is recommended.
`-m` '*permission*'	Specify a white-space separated list of file system permissions for the device node of the device driver. If `-a` or `-d` option is not specified then, this option is ignored. The permission string is mandatory.
`-p` '*policy*'	With the `-a` option, policy is a white-space separated list of complete device policies. For the `-d` option, policy is a white space separated list of minor device specifications. The minor device specifications are matched exactly against the entries in `/etc/security/device_policy`, that is., no wildcard matching is performed.
`-P` '*privilege*'	With the `-a` option, privilege is a comma separated list of additional driver privileges. For the `-d` option, privilege is a single privilege. The privileges are added to or removed from the `/etc/security/extra_privs` file.
`-v`	Verbose.

Examples

EXAMPLE 1 Adding or Modifying an Existing Minor Permissions Entry

The following command adds or modifies the exisitng minor permissions entry of the `clone` driver:

```
example# update_drv -a -m 'llc1 777 joe staff' clone
```

EXAMPLE 2 Removing All Minor Permissions Entries

The following command removes all minor permission entries of the `usbprn` driver, the USB printer driver:

```
example# update_drv -d -m '* 0666 root sys' usbprn
```

EXAMPLE 3 Adding a Driver Aliases Entry

The following command adds a driver aliases entry of the `ugen` driver with the identity string of `usb459,20`:

```
example# update_drv -a -i '"usb459,20"' ugen
```

EXAMPLE 4 Re-reading the `driver.conf` File For the `ohci` Driver

The following command re-reads the `driver.conf`(4) file.

```
example# update_drv ohci
```

EXAMPLE 5 Requiring a Self-defined Privilege to Open a tcp Socket

The following command requires a self-defined privilege to open a tcp socket:

```
example#   update_drv -a -P net_tcp -p \
     'write_priv_set=net_tcp read_priv_set=net_tcp' tcp
```

EXAMPLE 6 Establishing a Path-oriented Alias

The following command establishes a path-oriented alias to force a specific driver, `qlt`, to be used for a particular device path:

```
example#   update_drv -a -i '"/pci@8,600000/SUNW,qlc@4"' qlt
```

Exit Status

The following exit values are returned:

`0` Successful completion.

>`0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also

`add_drv`(1M), `modunload`(1M), `rem_drv`(1M), `driver.conf`(4), `attributes`(5), `privileges`(5)

Notes

If `-a` or `-d` options are specified, `update_drv` does not re-read the `driver.conf` file.

A forced update of the `driver.conf` file reloads the `driver.conf` file without reloading the driver binary module. In this case, devices which can not be detached reference driver global properties from the old `driver.conf` file, while the remaining driver instances reference global properties in the new `driver.conf` file.

It is possible to add an alias , which changes the driver binding of a device already being managed by a different driver. A force update with the `-a` option tries to bind to the new driver and report error if it cannot. If you specify more than one of the `-m`, `-i`, `-P` or `-p` options, a force flag tries to modify aliases or permissions. This is done even if the other operation fails and vice-versa. A force update with the `-d` option tries to delete entries and report the error if it cannot.

Name updatehome – update the home directory copy and link files for the current label

Synopsis

```
/usr/bin/updatehome  [-cirs]
```

Description `updatehome` reads the user's minimum-label copy and link-control files (`.copy_files` and `.link_files`). These files contain a list of files to be copied and symbolically linked from the user's minimum-label home directory to the user's home directory at the current label.

The Solaris Trusted Extensions `dtsession` program performs an `updatehome` whenever a newly labeled workspace is created so that the user's favorite files are available for use. For example, the user probably wants a symbolic link to such files as `.profile`, `.login`, `.cshrc`, `.exrc`, `.mailrc`, and `~/bin`. The `updatehome` command provides a convenient mechanism for accomplishing this symlink. The user can add files to those to be copied (`.copy_files`) and to those to be symbolically linked (`.link_files`).

Options

`-c` Replace existing home-directory copies at the current label. The default is to skip over existing copies.

`-i` Ignore errors encountered. The default aborts on error.

`-r` Replace existing home-directory copies or symbolic links at the current label. This option implies options `-c` and `-s`. The default is to skip over existing copies or symbolic links.

`-s` Replace existing home-directory symbolic links at the current label. The default is to skip over existing symbolic links.

Exit Status Upon success, `updatehome` returns `0`. Upon failure, `updatehome` returns `1` and writes diagnostic messages to standard error.

Examples

EXAMPLE 1 A Sample `.copy_files` File

The files that are listed in `.copy_files` can be modified at every user's label.

```
.cshrc
.mailrc
.mozilla/bookmarks.html
```

EXAMPLE 2 A Sample `.link_files` File

The files that are listed in `.link_files` can be modified at the lowest label. The changes propagate to the other labels that are available to the user.

```
~/bin
.mozilla/preferences
.xrc
.rhosts
```

EXAMPLE 3 Updating the Linked and Copied Files

The `.copy_files` and `.link_files` were updated by the user at the minimum label. At a higher label, the user refreshes the copies and the links. No privileges are required to run the command.

```
% updatehome -r
```

Files

`$HOME/.copy_files`	List of files to be copied
`$HOME/.link_files`	List of files to be symbolically linked

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu
Interface Stability	Committed

See Also `attributes(5)`

"`.copy_files` and `.link_files` Files" in *Oracle Solaris Trusted Extensions Administrator's Procedures*

Notes The functionality described on this manual page is available only if the system is configured with Trusted Extensions.

Name updatemanager – start the Sun Update Manager application

Synopsis

```
updatemanager
```

Description The `updatemanager` command starts the Sun Update Manager application. The Sun Update Manager is a graphical user interface (GUI) that you can use to manage updates and patches on your Solaris system.

For more information about using this application, see the *Sun Update Manager 1.0 Administration Guide* in the *Sun Update Connection, System Edition Collection* on `docs.sun.com`.

You can also manage updates and patches on the command line by using the `smpatch` command. See the `smpatch(1M)` man page.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWupdatemgru
Interface Stability	Unstable

See Also `smpatch(1M)`, `attributes(5)`

Name updatemedia – modify Solaris media with patches and packages

Synopsis

```
/usr/bin/updatemedia  -d media-root [-v] [-o iso] [-l label]
     pkg_or_patch [pkg_or_patch ...]
```

Description The `updatemedia` utility takes a list of patches and packages as input and updates the install miniroot in *media-root* (the root directory of an on-disk image of a Solaris installation media) to include the specified patches and packages. These patches and packages are also placed in a subdirectory called `DU` under the Solaris install image. For example:

media-root`/Solaris_10/DU`

When booting a system from the updated media, the patches and packages will be part of the booted Solaris image. They will also be applied to the target system being installed at the end of the installation process.

If `-o` is specified, a bootable ISO image is created in the file `media.iso` that contains the Solaris install media. The ISO image can then be burned onto a CD/DVD with utilities such as `cdrw(1)` or `cdrecord(1)`. (The latter is not a SunOS man page.)

Options The following options are supported:

`-d` *media-root*
Top-level directory of on-disk image of Solaris installation media. This is option must be specified.

`-o` *iso*
Create a Solaris ISO image of *media-root*.

`-l` *label*
Label/volume name of the ISO image (if `-o` option is specified). If `-o` is not specified, the name of Solaris directory under *media-root*, for example, `Solaris_10`, will be used.

`-v`
Verbose. Multiple `-v` options increase verbosity.

Operands The following operands are supported:

pkg_or_patch [*pkg_or_patch* ...]
One or more patches or packages (you can have both patches and packages in a single command) with which the Solaris installation media *media-root* will be updated.

Examples **EXAMPLE 1** Updating a Solaris Install Image with Patch and Package

The following command updates the Solaris install image in `s10u1` by adding patch `123456-07` and package `SUNWfoo`.

```
# /usr/bin/updatemedia -d s10u1 SUNWfoo 123456-07
```

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed

See Also cdrw(1), mkbootmedia(1M), attributes(5)

mkisofs(8), (/usr/share/man/man8/mkisofs.8), in the SUNWfsman package (not a SunOS man page)

Name useradd – administer a new user login on the system

Synopsis

```
useradd [-c comment] [-d dir] [-e expire] [-f inactive]
 [-g group] [-G group [, group...]] [-m [-k skel_dir]]
 [-u uid [-o]] [-s shell] [-A authorization [,authorization...]]
 [-P profile [,profile...]] [-R role [,role...]]
 [-p projname] [-K key=value] login

useradd -D [-b base_dir] [-e expire] [-f inactive]
 [-g group] [-A authorization [,authorization...]]
 [-P profile [,profile...]] [-R role [,role...]]
 [-p projname] [-K key=value]
```

Description `useradd` adds a new user to the `/etc/passwd` and `/etc/shadow` and `/etc/user_attr` files. The `-A` and `-P` options respectively assign authorizations and profiles to the user. The `-R` option assigns roles to a user. The `-p` option associates a project with a user. The `-K` option adds a *key=value* pair to `/etc/user_attr` for the user. Multiple *key=value* pairs may be added with multiple `-K` options.

`useradd` also creates supplementary group memberships for the user (`-G` option) and creates the home directory (`-m` option) for the user if requested. The new login remains locked until the `passwd(1)` command is executed.

Specifying `useradd -D` with the `-g`, `-b`, `-f`, `-e`, `-A`, `-P`, `-p`, `-R`, or `-K` option (or any combination of these options) sets the default values for the respective fields. See the `-D` option, below. Subsequent `useradd` commands without the `-D` option use these arguments.

The system file entries created with this command have a limit of 2048 characters per line. Specifying long arguments to several options can exceed this limit.

`useradd` requires that usernames be in the format described in `passwd(4)`. A warning message is displayed if these restrictions are not met. See `passwd(4)` for the requirements for usernames.

Options The following options are supported:

`-A` *authorization*
: One or more comma separated authorizations defined in `auth_attr(4)`. Only a user or role who has `grant` rights to the authorization can assign it to an account.

`-b` *base_dir*
: The default base directory for the system if `-d` *dir* is not specified. *base_dir* is concatenated with the account name to define the home directory. If the `-m` option is not used, *base_dir* must exist.

`-c` *comment*
: Any text string. It is generally a short description of the login, and is currently used as the field for the user's full name. This information is stored in the user's `/etc/passwd` entry.

`-d` *dir*
The home directory of the new user. It defaults to *base_dir*/*account_name*, where *base_dir* is the base directory for new login home directories and *account_name* is the new login name.

`-D`
Display the default values for `group`, `base_dir`, `skel_dir`, `shell`, `inactive`, `expire`, `proj`, `projname` and `key=value` pairs. When used with the `-g`, `-b`, `-f`, `-e`, `-A`, `-P`, `-p`, `-R`, or `-K` options, the `-D` option sets the default values for the specified fields. The default values are:

group	`other` (GID of 1)
base_dir	`/home`
skel_dir	`/etc/skel`
shell	`/bin/sh`
inactive	`0`
expire	null
auths	null
profiles	null
proj	`3`
projname	`default`
key=value (pairs defined in `user_attr(4)`	not present
roles	null

`-e` *expire*
Specify the expiration date for a login. After this date, no user will be able to access this login. The expire option argument is a date entered using one of the date formats included in the template file `/etc/datemsk`. See `getdate(3C)`.

If the date format that you choose includes spaces, it must be quoted. For example, you can enter `10/6/90` or "`October 6, 1990`". A null value ("") defeats the status of the expired date. This option is useful for creating temporary logins.

`-f` *inactive*
The maximum number of days allowed between uses of a login ID before that ID is declared invalid. Normal values are positive integers. A value of `0` defeats the status.

`-g` *group*
An existing group's integer ID or character-string name. Without the `-D` option, it defines the new user's primary group membership and defaults to the default group. You can reset this default value by invoking `useradd -D -g` *group*. GIDs 0-99 are reserved for allocation by the Solaris Operating System.

`-G` *group*
An existing group's integer ID or character-string name. It defines the new user's supplementary group membership. Duplicates between *group* with the `-g` and `-G` options are ignored. No more than `NGROUPS_MAX` groups can be specified. GIDs 0-99 are reserved for allocation by the Solaris Operating System.

`-K` *key=value*
A *key=value* pair to add to the user's attributes. Multiple `-K` options may be used to add multiple *key=value* pairs. The generic `-K` option with the appropriate key may be used instead of the specific implied key options (`-A`, `-P`, `-R`, `-p`). See `user_attr(4)` for a list of valid *key=value* pairs. The "type" key is not a valid key for this option. Keys may not be repeated.

`-k` *skel_dir*
A directory that contains skeleton information (such as `.profile`) that can be copied into a new user's home directory. This directory must already exist. The system provides the `/etc/skel` directory that can be used for this purpose.

`-m`
Create the new user's home directory if it does not already exist. If the directory already exists, it must have read, write, and execute permissions by *group*, where *group* is the user's primary group.

`-o`
This option allows a UID to be duplicated (non-unique).

`-P` *profile*
One or more comma-separated execution profiles defined in `prof_attr(4)`.

`-p` *projname*
Name of the project with which the added user is associated. See the *projname* field as defined in `project(4)`.

`-R` *role*
One or more comma-separated execution profiles defined in `user_attr(4)`. Roles cannot be assigned to other roles.

`-s` *shell*
Full pathname of the program used as the user's shell on login. It defaults to an empty field causing the system to use `/bin/sh` as the default. The value of *shell* must be a valid executable file.

`-u` *uid*
The UID of the new user. This UID must be a non-negative decimal integer below `MAXUID` as defined in `<sys/param.h>`. The UID defaults to the next available (unique) number above the highest number currently assigned. For example, if UIDs 100, 105, and 200 are assigned, the next default UID number will be 201. UIDs `0-99` are reserved for allocation by the Solaris Operating System.

Files

`/etc/datemsk`

`/etc/passwd`

`/etc/shadow`

`/etc/group`

`/etc/skel`

`/usr/include/limits.h`

`/etc/user_attr`

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Evolving

See Also

`passwd(1)`, `profiles(1)`, `roles(1)`, `users(1B)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `grpck(1M)`, `logins(1M)`, `pwck(1M)`, `userdel(1M)`, `usermod(1M)`, `getdate(3C)`, `auth_attr(4)`, `passwd(4)`, `prof_attr(4)`, `project(4)`, `user_attr(4)`, `attributes(5)`

Diagnostics

In case of an error, `useradd` prints an error message and exits with a non-zero status.

The following indicates that `login` specified is already in use:

```
UX: useradd: ERROR: login is already in use. Choose another.
```

The following indicates that the *uid* specified with the `-u` option is not unique:

```
UX: useradd: ERROR: uid uid is already in use. Choose another.
```

The following indicates that the *group* specified with the `-g` option is already in use:

```
UX: useradd: ERROR: group group does not exist. Choose another.
```

The following indicates that the *uid* specified with the `-u` option is in the range of reserved UIDs (from `0-99`):

```
UX: useradd: WARNING: uid uid is reserved.
```

The following indicates that the *uid* specified with the `-u` option exceeds `MAXUID` as defined in `<sys/param.h>`:

```
UX: useradd: ERROR: uid uid is too big. Choose another.
```

The following indicates that the `/etc/passwd` or `/etc/shadow` files do not exist:

```
UX: useradd: ERROR: Cannot update system files - login cannot be created.
```

Notes The `useradd` utility adds definitions to only the local `/etc/group`, `etc/passwd`, `/etc/passwd`, `/etc/shadow`, `/etc/project`, and `/etc/user_attr` files. If a network name service such as NIS or NIS+ is being used to supplement the local `/etc/passwd` file with additional entries, `useradd` cannot change information supplied by the network name service. However `useradd` will verify the uniqueness of the user name (or role) and user id and the existence of any group names specified against the external name service.

Name userdel – delete a user's login from the system

Synopsis

```
userdel [-r] login
```

Description The `userdel` utility deletes a user account from the system and makes the appropriate account-related changes to the system file and file system.

Options The following options are supported:

`-r` Remove the user's home directory from the system. This directory must exist. The files and directories under the home directory will no longer be accessible following successful execution of the command.

Operands The following operands are supported:

login An existing login name to be deleted.

Exit Status The following exit values are returned:

`0` Successful completion.

`2` Invalid command syntax. A usage message for the `userdel` command is displayed.

`6` The account to be removed does not exist.

`8` The account to be removed is in use.

`10` Cannot update the `/etc/group` or `/etc/user_attr` file but the login is removed from the `/etc/passwd` file.

`12` Cannot remove or otherwise modify the home directory.

Files

`/etc/passwd`	system password file
`/etc/shadow`	system file contain users' encrypted passwords and related information
`/etc/group`	system file containing group definitions
`/etc/user_attr`	system file containing additional user attributes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `auths(1)`, `passwd(1)`, `profiles(1)`, `roles(1)`, `users(1B)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `logins(1M)`, `roleadd(1M)`, `rolemod(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `passwd(4)`, `prof_attr(4)`, `user_attr(4)`, `attributes(5)`

Notes The `userdel` utility only deletes an account definition that is in the local `/etc/group`, `/etc/passwd`, `/etc/shadow`, and `/etc/user_attr` file. file. If a network name service such as NIS or NIS+ is being used to supplement the local `/etc/passwd` file with additional entries, `userdel` cannot change information supplied by the network name service.

Name usermod – modify a user's login information on the system

Synopsis

```
usermod [-u uid [-o]] [-g group] [-G group [, group...]]
     [-d dir [-m]] [-s shell] [-c comment] [-l new_name]
     [-f inactive] [-e expire]
     [-A authorization [, authorization]]
     [-P profile [, profile]] [-R role [, role]]
     [-K key=value] login
```

Description The `usermod` utility modifies a user's login definition on the system. It changes the definition of the specified login and makes the appropriate login-related system file and file system changes.

The system file entries created with this command have a limit of 512 characters per line. Specifying long arguments to several options might exceed this limit.

Options The following options are supported:

`-A` *authorization*
: One or more comma separated authorizations as defined in `auth_attr(4)`. Only a user or role who has `grant` rights to the `authorization` can assign it to an account. This replaces any existing authorization setting. If no authorization list is specified, the existing setting is removed.

`-c` *comment*
: Specify a comment string. *comment* can be any text string. It is generally a short description of the login, and is currently used as the field for the user's full name. This information is stored in the user's `/etc/passwd` entry.

`-d` *dir*
: Specify the new home directory of the user. It defaults to *base_dir/login*, where *base_dir* is the base directory for new login home directories, and `login` is the new login.

`-e` *expire*
: Specify the expiration date for a login. After this date, no user will be able to access this login. The expire option argument is a date entered using one of the date formats included in the template file `/etc/datemsk`. See `getdate(3C)`.

  For example, you may enter `10/6/90` or `October 6, 1990`. A value of "`''` defeats the status of the expired date.

`-f` *inactive*
: Specify the maximum number of days allowed between uses of a login ID before that login ID is declared invalid. Normal values are positive integers. A value of `0` defeats the status.

`-g` *group*
: Specify an existing group's integer ID or character-string name. It redefines the user's primary group membership.

`-G` *group*
Specify an existing group's integer "ID" "," or character string name. It redefines the user's supplementary group membership. Duplicates between *group* with the `-g` and `-G` options are ignored. No more than `NGROUPS_UMAX` groups may be specified as defined in `<param.h>`.

`-K` *key=value*
Replace existing or add to a user's *key=value* pair attributes. Multiple `-K` options can be used to replace or add multiple *key=value* pairs. However, keys must not be repeated. The generic `-K` option with the appropriate key can be used instead of the specific implied key options (`-A`, `-P`, `-R`, `-p`). See `user_attr(4)` for a list of valid *key*s. Values for these keys are usually found in man pages or other sources related to those keys. For example, see `project(4)` for guidance on values for the `project` key. Use the command `ppriv(1)` with the `-v` and `-l` options for a list of values for the keys `defaultpriv` and `limitpriv`.

The keyword `type` can be specified with the value `role` or the value `normal`. When using the value `role`, the account changes from a normal user to a role; using the value `normal` keeps the account a normal user.

As a `role` account, no roles (`-R` or *roles=value*) can be present.

`-l` *new_logname*
Specify the new login name for the user. See `passwd(4)` for the requirements for usernames.

`-m`
Move the user's home directory to the new directory specified with the `-d` option. If the directory already exists, it must have permissions read/write/execute by *group*, where *group* is the user's primary group.

`-o`
This option allows the specified UID to be duplicated (non-unique).

`-P` *profile*
One or more comma-separated rights profiles defined in `prof_attr(4)`. This replaces any existing profile setting in `user_attr(4)`. If an empty profile list is specified, the existing setting is removed.

`-R` *role*
One or more comma-separated roles (see `roleadd(1M)`). This replaces any existing role setting. If no role list is specified, the existing setting is removed.

`-s` *shell*
Specify the full pathname of the program that is used as the user's shell on login. The value of *shell* must be a valid executable file.

`-u` *uid*
Specify a new UID for the user. It must be a non-negative decimal integer less than `MAXUID` as defined in `<param.h>`. The UID associated with the user's home directory is not modified with this option; a user will not have access to their home directory until the UID is manually reassigned using `chown(1)`.

Operands The following operands are supported:

`login`
An existing login name to be modified.

Examples

EXAMPLE 1 Assigning Privileges to a User

The following command adds the privilege that affects high resolution times to a user's initial, inheritable set of privileges.

```
# usermod -K defaultpriv=basic,proc_clock_highres jdoe
```

This command results in the following entry in `user_attr`:

```
jdoe::::type=normal;defaultpriv=basic,proc_clock_highres
```

EXAMPLE 2 Removing a Privilege from a User's Limit Set

The following command removes the privilege that allows the specified user to create hard links to directories and to unlink directories.

```
# usermod -K limitpriv=all,!sys_linkdir jdoe
```

This command results in the following entry in `user_attr`:

```
jdoe::::type=normal;defaultpriv=basic,limitpriv=all,!sys_linkdir
```

EXAMPLE 3 Removing a Privilege from a User's Basic Set

The following command removes the privilege that allows the specified user to examine processes outside the user's session.

```
# usermod -K defaultpriv=basic,!proc_session jdoe
```

This command results in the following entry in `user_attr`:

```
jdoe::::type=normal;defaultpriv=basic,!proc_session;limitpriv=all
```

EXAMPLE 4 Assigning a Role to a User

The following command assigns a role to a user. The role must have been created prior to this command, either through use of the Solaris Management Console GUI or through `roleadd(1M)`.

```
# usermod -R mailadm jdoe
```

This command results in the following entry in `user_attr`:

```
jdoe::::type=normal;roles=mailadm;defaultpriv=basic;limitpriv=all
```

EXAMPLE 5 Removing All Profiles from a User

The following command removes all profiles that were granted to a user directly. The user will still have any rights profiles that are granted by means of the `PROFS_GRANTED` key in `policy.conf(4)`.

EXAMPLE 5 Removing All Profiles from a User *(Continued)*

```
# usermod -P "" jdoe
```

Exit Status In case of an error, `usermod` prints an error message and exits with one of the following values:

`2`
The command syntax was invalid. A usage message for the `usermod` command is displayed.

`3`
An invalid argument was provided to an option.

`4`
The *uid* given with the `-u` option is already in use.

`5`
The password files contain an error. `pwconv(1M)` can be used to correct possible errors. See `passwd(4)`.

`6`
The login to be modified does not exist, the *group* does not exist, or the login shell does not exist.

`8`
The login to be modified is in use.

`9`
The *new_logname* is already in use.

`10`
Cannot update the `/etc/group` or `/etc/user_attr` file. Other update requests will be implemented.

`11`
Insufficient space to move the home directory (`-m` option). Other update requests will be implemented.

`12`
Unable to complete the move of the home directory to the new home directory.

Files `/etc/group`
system file containing group definitions

`/etc/datemsk`
system file of date formats

`/etc/passwd`
system password file

`/etc/shadow`
system file containing users' encrypted passwords and related information

`/etc/user_attr`
system file containing additional user and role attributes

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed

See Also `chown(1)`, `passwd(1)`, `users(1B)`, `groupadd(1M)`, `groupdel(1M)`, `groupmod(1M)`, `logins(1M)`, `pwconv(1M)`, `roleadd(1M)`, `roledel(1M)`, `rolemod(1M)`, `useradd(1M)`, `userdel(1M)`, `getdate(3C)`, `auth_attr(4)`, `passwd(4)`, `policy.conf(4)`, `prof_attr(4)`, `user_attr(4)`, `attributes(5)`

Notes The `usermod` utility modifies `passwd` definitions only in the local `/etc/passwd` and `/etc/shadow` files. If a network nameservice such as `NIS` or `NIS+` is being used to supplement the local files with additional entries, `usermod` cannot change information supplied by the network nameservice. However `usermod` will verify the uniqueness of user name and user ID against the external nameservice.

The `usermod` utility uses the `/etc/datemsk` file, available with SUNWaccr, for date formatting.

Name utmpd – utmpx monitoring daemon

Synopsis

```
utmpd [-debug]
```

Description The `utmpd` daemon monitors the `/var/adm/utmpx` file. See `utmpx(4)` (and `utmp(4)` for historical information).

`utmpd` receives requests from `pututxline(3C)` by way of a named pipe. It maintains a table of processes and uses `poll(2)` on `/proc` files to detect process termination. When `utmpd` detects that a process has terminated, it checks that the process has removed its `utmpx` entry from `/var/adm/utmpx`. If the process' `utmpx` entry has not been removed, `utmpd` removes the entry. By periodically scanning the `/var/adm/utmpx` file, `utmpd` also monitors processes that are not in its table.

Options

`-debug`
: Run in debug mode, leaving the process connected to the controlling terminal. Write debugging information to standard output.

Exit Status The following exit values are returned:

`0`
: Successful completion.

`>0`
: An error occurred.

Files

`/etc/default/utmpd`
: You can set default values for the flags listed below. For example: `SCAN_PERIOD=600`

  `SCAN_PERIOD`
  : The number of seconds that `utmpd` sleeps between checks of `/proc` to see if monitored processes are still alive. The default is 300.

  `MAX_FDS`
  : The maximum number of processes that `utmpd` attempts to monitor. The default value is 4096.

  `WTMPX_UPDATE_FREQ`
  : The number of seconds that `utmpd` sleeps between read accesses of the `wtmpx` file. The `wtmpx` file's last access time is used by `init(1M)` on reboot to determine when the operating system became unavailable. The default is 60.

`/var/adm/utmpx`
: File containing user and accounting information for commands such as `who(1)`, `write(1)`, and `login(1)`.

`/proc`
: Directory containing files for processes whose `utmpx` entries are being monitored.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also svcs(1), init(1M), svcadm(1M), poll(2), pututxline(3C), proc(4), utmp(4), utmpx(4), attributes(5), smf(5)

Notes If the filesystem holding /var/adm/wtmpx is mounted with options which inhibit or defer access time updates, an unknown amount of error will be introduced into the utmp DOWN_TIME record's timestamp in the event of an uncontrolled shutdown (for example, a crash or loss of power). Controlled shutdowns will update the modify time of /var/adm/wtmpx, which will be used on the next boot to determine when the previous shutdown ocurred, regardless of access time deferral or inhibition.

The utmpd service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/system/utmp:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name uucheck – check the uucp directories and permissions file

Synopsis

```
/usr/lib/uucp/uucheck [-v] [-x debug-level]
```

Description `uucheck` checks for the presence of the `uucp` system required files and directories. `uucheck` also does error checking of the `Permissions` file (`/etc/uucp/Permissions`).

`uucheck` is executed during package installation. `uucheck` can only be used by the super-user or `uucp`.

Options The following options are supported:

`-v` Give a detailed (verbose) explanation of how the `uucp` programs will interpret the `Permissions` file.

`-x` *debug-level* Produce debugging output on the standard output. *debug-level* is a number from `0` to `9`. Higher numbers give more detailed debugging information.

Files

`/etc/uucp/Devices`

`/etc/uucp/Limits`

`/etc/uucp/Permissions`

`/etc/uucp/Systems`

`/var/spool/locks/*`

`/var/spool/uucp/*`

`/var/spool/uucppublic/*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

See Also `uucp(1C)`, `uustat(1C)`, `uux(1C)`, `uucico(1M)`, `uusched(1M)`, `attributes(5)`

Bugs The program does not check file/directory modes or some errors in the `Permissions` file such as duplicate login or machine name.

Name uucico – file transport program for the uucp system

Synopsis
```
/usr/lib/uucp/uucico [-f] [-c type] [-d spool-directory]
     [-i interface] [-r role-number] [-s system-name]
     [-x debug-level]
```

Description `uucico` is the file transport program for `uucp` work file transfers.

Options The following options are supported:

`-f` This option is used to "force execution" of `uucico` by ignoring the limit on the maximum number of uucicos defined in the `/etc/uucp/Limits` file.

`-c` *type* The first field in the `Devices` file is the "Type" field. The `-c` option forces `uucico` to only use entries in the "Type" field that match the user specified `type`. The specified `type` is usually the name of a local area network.

`-d` *spool-directory* This option specifies the directory *spool-directory* that contains the `uucp` work files to be transferred. The default spool directory is `/var/spool/uucp`.

`-i` *interface* This option defines the *interface* used with `uucico`. The interface only affects slave mode. Known interfaces are UNIX (default), TLI (basic Transport Layer Interface), and TLIS (Transport Layer Interface with Streams modules, read/write).

`-r` *role-number* The *role-number* `1` is used for master mode. *role-number* `0` is used for slave mode (default). When `uucico` is started by a program or `cron`, *role-number* `1` should be used for master mode.

`-s` *system-name* The `-s` option defines the remote system (*system-name*) that `uucico` will try to contact. It is required when the role is master; *system-name* must be defined in the `Systems` file.

`-x` *debug-level* Both `uux` and `uucp` queue jobs that will be transferred by `uucico`. These jobs are normally started by the `uusched` scheduler, for debugging purposes, and can be started manually. For example, the shell `Uutry` starts `uucico` with debugging turned on. The *debug-level* is a number between `0` and `9`. Higher numbers give more detailed debugging information.

Files `/etc/uucp/Devconfig`

`/etc/uucp/Devices`

`/etc/uucp/Limits`

`/etc/uucp/Permissions`

`/etc/uucp/Sysfiles`

`/etc/uucp/Systems`

`/var/spool/locks/*`

`/var/spool/uucp/*`

`/var/spool/uucppublic/*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

See Also `uucp(1C)`, `uustat(1C)`, `uux(1C)`, `Uutry(1M)`, `cron(1M)`, `uusched(1M)`, `attributes(5)`

Name uucleanup – uucp spool directory clean-up

Synopsis

```
/usr/lib/uucp/uucleanup [-Ctime] [-Dtime] [-mstring]
     [-otime] [-ssystem] [-Wtime] [-xdebug-level] [-Xtime]
```

Description `uucleanup` will scan the spool directories for old files and take appropriate action to remove them in a useful way:

- Inform the requester of send/receive requests for systems that can not be reached.
- Return undeliverable mail to the sender.
- Deliver `rnews` files addressed to the local system.
- Remove all other files.

In addition, there is a provision to warn users of requests that have been waiting for a given number of days (default 1 day). Note: `uucleanup` will process as if all option `times` were specified to the default values unless `time` is specifically set.

This program is typically started by the shell `uudemon.cleanup`, which should be started by `cron(1M)`.

Options

`-Ctime`	Remove any `C.` files greater or equal to `time` days old and send appropriate information to the requester (default 7 days).
`-Dtime`	Remove any `D.` files greater or equal to `time` days old, make an attempt to deliver mail messages, and execute `rnews` when appropriate (default 7 days).
`-m`*string*	Include *string* in the warning message generated by the `-W` option. The default line is "See your local administrator to locate the problem".
`-otime`	Delete other files whose age is more than `time` days (default 2 days).
`-s`*system*	Execute for *system* spool directory only.
`-Wtime`	Any `C.` files equal to `time` days old will cause a mail message to be sent to the requester warning about the delay in contacting the remote. The message includes the *JOBID*, and in the case of mail, the mail message. The administrator may include a message line telling whom to call to check the problem (`-m` option) (default 1 day).
`-x`*debug-level*	Produce debugging output on standard ouput. *debug-level* is a single digit between 0 and 9; higher numbers give more detailed debugging information. (This option may not be available on all systems.)
`-Xtime`	Any `X.` files greater or equal to `time` days old will be removed. The `D.` files are probably not present (if they were, the `X.` could get executed). But if there are `D.` files, they will be taken care of by D. processing (default 2 days).

Files

`/usr/lib/uucp`	directory with commands used by `uucleanup` internally
`/var/spool/uucp`	spool directory

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

See Also `uucp(1C)`, `uux(1C)`, `cron(1M)`, `attributes(5)`

Name uusched – uucp file transport program scheduler

Synopsis

```
/usr/lib/uucp/uusched [-u debug-level] [-x debug-level]
```

Description `uusched` is the `uucp(1C)` file transport scheduler. It is usually started by the daemon *uudemon.hour* that is started by `cron(1M)` from an entry in user `uucp`'s crontab file:

```
11,41 * * * * /etc/uucp/uucp/uudemon.hour
```

Options The options are for debugging purposes only. *debug-level* is a number between `0` and `9`. Higher numbers give more detailed debugging information:

The following options are supported:

`-u` *debug-level* Passes the `-u` *debug-level* option `uucico(1M)` as `-x` *debug-level*.

`-x` *debug-level* Outputs debugging messages from `uusched`.

Files

```
/etc/uucp/Devices
/etc/uucp/Permissions
/etc/uucp/Systems
/var/spool/locks/*
/var/spool/uucp/*
/var/spool/uucppublic/*
```

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

See Also `uucp(1C)`, `uustat(1C)`, `uux(1C)`, `cron(1M)`, `uucico(1M)`, `attributes(5)`

Name Uutry, uutry – attempt to contact remote system with debugging on

Synopsis `/usr/lib/uucp/Uutry [-r] [-c` *type*`] [-x` *debug-level*`]` *system-name*

Description `Uutry` is a shell script that is used to invoke `uucico`(1M) to call a remote site. Debugging is initially turned on and is set to the default value of `5`. The debugging output is put in file `/tmp/`*system-name*.

Options The following options are supported:

`-r` This option overrides the retry time that is set in file `/var/uucp/.Status/`*system-name*.

`-c` *type* The first field in the `Devices` file is the "Type" field. The `-c` option forces `uucico` to use only entries in the "Type" field that match the user-specified `type`. The specified `type` is usually the name of a local area network.

`-x` *debug-level* *debug-level* is a number from `0` to `9`. Higher numbers give more detailed debugging information.

Files `/etc/uucp/Devices`

`/etc/uucp/Limits`

`/etc/uucp/Permissions`

`/etc/uucp/Systems`

`/tmp/`*system-name*

`/var/spool/locks/*`

`/var/spool/uucp/*`

`/var/spool/uucppublic/*`

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

See Also `uucp`(1C), `uux`(1C), `uucico`(1M), `attributes`(5)

Name uuxqt – execute remote command requests

Synopsis `/usr/lib/uucp/uuxqt [-s system] [-x debug-level]`

Description `uuxqt` is the program that executes remote job requests from remote systems generated by the use of the `uux` command. (`mail` uses `uux` for remote mail requests). `uuxqt` searches the spool directories looking for execution requests. For each request, `uuxqt` checks to see if all the required data files are available, accessible, and the requested commands are permitted for the requesting system. The `Permissions` file is used to validate file accessibility and command execution permission.

There are two environment variables that are set before the `uuxqt` command is executed:

- `UU_MACHINE` is the machine that sent the job (the previous one).
- `UU_USER` is the user that sent the job.

These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

Options The following options are supported:

`-s` *system* Specifies the remote *system* name.

`-x` *debug-level* *debug-level* is a number from `0` to `9`. Higher numbers give more detailed debugging information.

Files

`/etc/uucp/Limits`

`/etc/uucp/Permissions`

`/var/spool/locks/*`

`/var/spool/uucp/*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWbnuu

See Also `mail(1)`, `uucp(1C)`, `uustat(1C)`, `uux(1C)`, `uucico(1M)`, `attributes(5)`

Name vmstat – report virtual memory statistics

Synopsis

```
vmstat [-cipqsS] [disks] [interval [count]]
```

Description `vmstat` reports virtual memory statistics regarding kernel thread, virtual memory, disk, trap, and CPU activity.

On MP (multi-processor) systems, `vmstat` averages the number of CPUs into the output. For per-processor statistics, see `mpstat(1M)`.

`vmstat` only supports statistics for certain devices. For more general system statistics, use `sar(1)`, `iostat(1M)`, or `sar(1M)`.

Without options, `vmstat` displays a one-line summary of the virtual memory activity since the system was booted.

During execution of the kernel status command, the `state` of the system can change. If relevant, a state change message is included in the `vmstat` output, in one of the following forms:

```
<<device added: sd0>>
<<device removed: sd0>>
<<processors added: 1, 3>>
<<processors removed: 1, 3>>
```

See *System Administration Guide: Advanced Administration* for device naming conventions for disks.

Options The following options are supported:

`-c` Report cache flushing statistics. This option is obsolete, and no longer meaningful. This option might be removed in a future version of Solaris.

`-i` Report the number of interrupts per device. *count* and *interval* does not apply to the `-i` option.

`-p` Report paging activity in details. This option will display the following, respectively:

- epi Executable page-ins.
- epo Executable page-outs.
- epf Executable page-frees.
- api Anonymous page-ins.
- apo Anonymous page-outs.
- apf Anonymous page-frees.
- fpi File system page-ins.
- fpo File system page-outs.

fpf File system page-frees.

When executed in a zone and if the pools facility is active, all of the above only report actitivity on the processors in the processor set of the zone's pool.

`-q` Suppress messages related to state changes.

`-s` Display the total number of various system events since boot. *count* and *interval* does not apply to the `-s` option.

`-S` Report on swapping rather than paging activity. This option will change two fields in `vmstat`'s "paging" display: rather than the "re" and "mf" fields, `vmstat` will report "si" (swap-ins) and "so" (swap-outs).

Operands The following operands are supported:

count Specifies the number of times that the statistics are repeated. *count* does not apply to the `-i` and `-s` options.

disks Specifies which disks are to be given priority in the output (only four disks fit on a line). Common disk names are `id`, `sd`, `xd`, or `xy,` followed by a number (for example, `sd2`, `xd0`, and so forth).

interval Specifies the last number of seconds over which `vmstat` summarizes activity. This number of seconds repeats forever. *interval* does not apply to the `-i` and `-s` options.

Examples **EXAMPLE 1** Using `vmstat`

The following command displays a summary of what the system is doing every five seconds.

```
example% vmstat 5


kthr   memory          page             disk      faults        cpu
r b w swap  free re mf pi p fr de sr s0 s1 s2 s3  in  sy  cs us sy id
0 0 0 11456 4120 1  41 19 1  3  0  2  0  4  0  0  48 112 130  4 14 82
0 0 1 10132 4280 0   4 44 0  0  0  0  0 23  0  0 211 230 144  3 35 62
0 0 1 10132 4616 0   0 20 0  0  0  0  0 19  0  0 150 172 146  3 33 64
0 0 1 10132 5292 0   0  9 0  0  0  0  0 21  0  0 165 105 130  1 21 78
1 1 1 10132 5496 0   0  5 0  0  0  0  0 23  0  0 183  92 134  1 20 79
1 0 1 10132 5564 0   0 25 0  0  0  0  0 18  0  0 131 231 116  4 34 62
1 0 1 10124 5412 0   0 37 0  0  0  0  0 22  0  0 166 179 118  1 33 67
1 0 1 10124 5236 0   0 24 0  0  0  0  0 14  0  0 109 243 113  4 56 39
^C

example%
```

The fields of `vmstat`'s display are

EXAMPLE 1 Using `vmstat` *(Continued)*

`kthr` Report the number of kernel threads in each of the three following states:

- `r` the number of kernel threads in run queue
- `b` the number of blocked kernel threads that are waiting for resources I/O, paging, and so forth
- `w` the number of swapped out lightweight processes (LWPs) that are waiting for processing resources to finish.

`memory` Report on usage of virtual and real memory.

- `swap` available swap space (Kbytes)
- `free` size of the free list (Kbytes)

`page` Report information about page faults and paging activity. The information on each of the following activities is given in units per second.

- `re` page reclaims — but see the `-S` option for how this field is modified.
- `mf` minor faults — but see the `-S` option for how this field is modified.
- `pi` kilobytes paged in
- `po` kilobytes paged out
- `fr` kilobytes freed
- `de` anticipated short-term memory shortfall (Kbytes)
- `sr` pages scanned by clock algorithm

When executed in a `zone` and if the pools facility is active, all of the above (except for "de") only report activity on the processors in the processor set of the `zone`'s pool.

`disk` Report the number of disk operations per second. There are slots for up to four disks, labeled with a single letter and number. The letter indicates the type of disk (s = SCSI, i = IPI, and so forth); the number is the logical unit number.

`faults` Report the trap/interrupt rates (per second).

- `in` interrupts
- `sy` system calls
- `cs` CPU context switches

When executed in a `zone` and if the pools facility is active, all of the above only report actitivity on the processors in the processor set of the `zone`'s pool.

EXAMPLE 1 Using `vmstat` *(Continued)*

`cpu` Give a breakdown of percentage usage of CPU time. On MP systems, this is an average across all processors.

`us` user time

`sy` system time

`id` idle time

When executed in a `zone` and if the pools facility is active, all of the above only report actitivity on the processors in the processor set of the `zone`'s pool.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	See below.

Invocation is evolving. Human readable output is unstable.

See Also

`sar(1)`, `iostat(1M)`, `mpstat(1M)`, `sar(1M)`, `attributes(5)`

System Administration Guide: Basic Administration

System Administration Guide: Advanced Administration

Notes

The sum of CPU utilization might vary slightly from 100 because of rounding errors in the production of a percentage figure.

The `-c` option (Report cache flushing statistics) is not supported in this release.

Name vntsd – virtual network terminal server daemon for Logical Domains

Synopsis

```
/usr/lib/ldoms/vntsd
```

Description The `vntsd` daemon is a server that supports connections to the Logical Domains (LDoms) console by using `telnet(1)`. When a `telnet` session starts, `vntsd` sends `telnet` options to the client indicating a willingness to remotely echo characters and to suppress go ahead.

Consoles are organized into groups by the LDoms Manager. Each console group is assigned a unique group name and TCP port number. `vntsd` uses the group's port number to export access to the consoles within that group. To establish a connection with a console or console group, a user starts a `telnet(1)` session with the corresponding group's port number. Depending on the number of consoles within that group, `vntsd` does one of two things:

- If there is only one console in the group, `vntsd` connects a session to that LDoms console.
- If there are multiple consoles in the group, `vntsd` prompts the user to select the console to which they would like to connect, as shown in "Multiple-Console Options," below.

For each console, `vntsd` provides write access only to the first user connecting to the console. Subsequent users connecting to the console are allowed only to read from the console and wait for write access. When the first user disconnects, write privileges are transferred to the next user waiting in the queue. If a user who does not have write privileges attempts to write to a console, the `vntsd` displays the following message:

```
You do not have write access
```

A user who has no write access can acquire write access forcibly by using the `~w` special console command, described in "Special Console Commands," below.

`vntsd` can be invoked only with superuser privileges or by someone in the Primary Administrator role.

Options The options for `vntsd` are divided into multiple-console options and console commands.

Multiple-Console Options

The options listed below are supported when there are multiple LDoms consoles in a group. The syntax for the use of these options is:

<*hostname*>`-vnts-`<*group-name*>`:` <*option*>

For example:

```
myhost-vnts-salesgroup: h
```

The `h` option invokes help, as described below.

`h`

Display the following help text:

```
h -- this help
l -- list of consoles
q -- quit
c{id}, n{name} -- connect to console of domain {id} or domain name
```

`l`
List all consoles in the group. For example:

```
DOMAIN ID       DOMAIN NAME     DOMAIN STATE
0               ldg1            online
1               ldg2            connected
...             ...             ...
```

The two domain states and their meanings are:

`online`
No one is connected to the console.

`connected`
At least one user is already connected to the console.

`q`
Disconnect from `vntsd`.

c*{id}*, n*{name}*
Connect to specified console. Upon connection, the following message is displayed:

```
Connecting to console <domain-name> in group <group-name>
Press ~? for control options ....
```

Special Console Commands

A tilde (~) appearing as the first character of a line is an escape signal that directs `vntsd` to perform a special console command. The tilde-tilde (~~) sequence outputs a tilde. In conjunction with the initial tilde, `vntsd` accepts the following special console commands:

`~.`
Disconnect from the console or console group.

`~w`
Force write access to the console.

`~p`
Disconnect from this console, and connect to the console that precedes this console in the list of consoles.

`~n`
Disconnect from this console, and connect to the console that follows this console in the list of consoles.

`~#`
Send break.

`~^B`
Send alternate break.

`~?`
Display `vntsd` help, as follows:

```
~# - Send break
~^B - Send alternate break
~. - Exit from this console
~w - Force write access
~n - Console next
~p - Console previous
~? - Help
```

Files `/usr/lib/ldoms/vntsd`
Binary executable `vntsd` file.

`/usr/lib/ldoms/vntsd.xml`
Service management facility (`smf(5)`) manifest file for `vntsd`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWldoms
Interface Stability	Evolving

See Also `telnet(1)`, `svccfg(1M)`, `usermod(1M)`, `auth_attr(4)`, `attributes(5)`, `smf(5)`

Notes The `vntsd` is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/ldoms/vntsd
```

You can change the following properties using the `svccfg(1M)` command:

`vntsd/vcc_device`
Set an instance of the virtual console concentrator (`vcc`) driver to which `vntsd` is connected.

`vntsd/listen_addr`
Set the IP address to which `vntsd` listens, using the following syntax:

```
vntsd/listen_addr:"xxx.xxx.xxx.xxx"
```

...where *xxx.xxx.xxx.xxx* is a valid IP address. The default value of this property is to listen on IP address 127.0.0.1. Users can connect to a guest console over a network if the value is set to the IP address of the control domain.

Note – Enabling network access to a console has security implications. Any user can connect to a console and for this reason it is disabled by default.

`vntsd/timeout_minutes`
Set timeout in minutes. `vntsd` will timeout (close) `telnet` connection if there is no activity (input or output) on the console. The default value is 0, which disables timeout.

`vntsd/authorization`

Enable the authorization checking of users and roles for the domain console or consoles that are being accessed. The default value of this property is `false` to maintain backward compatibility. To enable authorization checking, use the `svccfg(1M)` command to set the property value to `true`. While this option is enabled, `vntsd` listens and accepts connections on `localhost`. If the `listen_addr` property specifies an alternate IP address when this option is enabled, `vntsd` ignores the alternate IP address and continues to listen on `localhost`. Connections that are initiated from other hosts will also fail. Authorizations are available to access all consoles or console groups, or to access specific consoles or console groups. When the `vntsd` service is enabled, the following authorization is added to the authorization description database, `auth_attr(4)`:

```
solaris.vntsd.consoles:::Access All LDoms Guest Consoles::
```

Add any fine-grained authorizations based on the name of the console group. For example, if the name of the console group to be authorized is `ldg1`, add the following entry to the `auth_attr(4)` file:

```
solaris.vntsd.console-ldg1:::Access Specific LDoms Guest Console::
```

By default, the authorization to access all consoles is assigned to the root user or role. The Primary Administrator (superuser) can use the `usermod(1M)` command to assign the required authorization or authorizations to other users or roles.

The following example gives user `user1` the authorization to access all domain consoles:

```
# usermod -A "solaris.vntsd.consoles" user1
```

The following example gives user `user1` the authorization to access the console group named `ldg1`:

```
# usermod -A "solaris.vntsd.console-ldg1" user1
```

Name volcopy – make an image copy of file system

Synopsis

```
volcopy [-F FSType] [-V] [generic_options]
     [-o FSType-specific_options] operands
```

Description `volcopy` makes a literal copy of the file system. This command may not be supported for all *FSTypes*.

Options The following options are supported:

`-F` *FSType* — Specify the *FSType* on which to operate. The *FSType* should either be specified here or be determinable from `/etc/vfstab` by matching the *operands* with an entry in the table. Otherwise, the default file system type specified in `/etc/default/fs` will be used.

`-V` — Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from `/etc/vfstab`. This option should be used to verify and validate the command line.

generic_options — Options that are commonly supported by most *FSType*-specific command modules. The following options are available:

- `-a` Require the operator to respond "yes" or "no" instead of simply waiting ten seconds before the copy is made.
- `-s` (Default) Invoke the DEL if wrong verification sequence.

`-o` *FSType-specific_options* — Specify *FSType*-specific options in a comma separated (without spaces) list of suboptions and keyword-attribute pairs for interpretation by the *FSType*-specific module of the command.

Operands The following operands are supported:

operands — generally include the device and volume names and are file system specific. A detailed description of the *operands* can be found on the *FSType*-specific man pages of `volcopy`.

Exit Status The following exit values are returned:

- `0` Successful file system copy
- `1` An error has occurred.

Files

`/etc/vfstab`	list of default parameters for each file system
`/etc/default/fs`	default local file system type. Default values can be set for the following flags in `/etc/default/fs`. For example: `LOCAL=ufs`.
`LOCAL:`	The default partition for a command if no *FSType* is specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `labelit(1M)`, `vfstab(4)`, `attributes(5)` Manual pages for the *FSType*-specific modules of `volcopy`.

Name volcopy_ufs – make an image copy of a ufs file system

Synopsis `volcopy [-F ufs] [`*generic_options*`] `*fsname srcdevice volname1 destdevice volname2*

Description `volcopy` makes a literal copy of the `ufs` file system using a blocksize matched to the device.

Options The following option is supported:

generic_options options supported by the generic `volcopy` command. See `volcopy(1M)`.

Operands The following operands are supported:

fsname represents the mount point (for example, `root`, `u1`, etc.) of the file system being copied.

srcdevice or *destdevice* the disk partition specified using the raw device (for example, `/dev/rdsk/cld0s8`, `/dev/rdsk/cld1s8`, etc.).

srcdevice and *volname1* the device and physical volume from which the copy of the file system is being extracted.

destdevice and *volname2* the target device and physical volume.

fsname and *volname* are limited to six or fewer characters and recorded in the superblock. *volname* may be '–' to use the existing volume name.

Exit Status The following exit values are returned:

`0` Successful file system copy.

non-zero An error has occurred.

Files `/var/adm/filesave.log` a record of file systems/volumes copied

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `cpio(1)`, `dd(1M)`, `labelit(1M)`, `volcopy(1M)`, `attributes(5)`, `ufs(7FS)`

Notes `volcopy` does not support copying to tape devices. Use `dd(1M)` for copying to and from tape devices.

Name vold – Volume Management daemon to manage removable media devices

Synopsis

```
/usr/sbin/vold  [-n] [-t] [-v] [-f config-file] [-l log-file]
 [-d root-dir] [-L debug-level]
```

Description The Volume Management daemon, `vold`, creates and maintains a file system image rooted at *root-dir* that contains symbolic names for removable media devices. These devices include CD-R, CD-RW, floppies, DVD, and USB and 1394 mass storage devices. The default *root-dir* is set to `/vol` if no directory is specified by the `-d` option.

`vold` reads the `/etc/vold.conf` configuration file upon startup. If the configuration file is modified later, `vold` must be told to reread the `/etc/vold.conf` file. Accomplish this rereading by entering:

```
example# svcadm refresh volfs
```

To tell `vold` to clean up and exit, enter:

```
example# svcadm disable volfs
```

To reenable:

```
example# svcadm enable volfs
```

To specify options (see OPTIONS):

```
example# svccfg
svc:> select volfs
svc:/system/filesystem/volfs> listprop vold/*
vold/config_file            astring
vold/log_debuglevel         count    0
vold/log_file               astring
vold/log_nfs_trace          boolean  false
vold/log_verbose            boolean  false
vold/never_writeback_label  boolean  false
vold/root_dir               astring
svc:/system/filesystem/volfs> setprop vold/never_writeback_label=true
svc:/system/filesystem/volfs> exit

# svcadm disable volfs
# svcadm enable volfs
```

`vold` is hotplug-aware for USB and 1394 mass storage devices, thus there is no need for stopping and restarting `vold`. It is recommended to `eject(1)` the "media" before hot-removing a device. The `eject` command unmounts any filesystems mounted from the media, making it safe to remove the device. (Note that all USB and 1394 devices, regardless of whether they contain removable media, are treated like removable media devices).

Options The following options are supported:

`-n`	Never writeback. Volume Management updates media labels with unique information if labels are not unique. This flag keeps Volume Management from changing your media. The default setting is FALSE.
`-t`	Dump NFS trace information to the log file. The default setting is FALSE.
`-v`	Provide lots of status information to the log file. The default setting is FALSE (do not provide status info to log file).
`-d` *root-dir*	Specify an alternate root directory. The default location is `/vol`. Setting this will also cause other Volume Management utilities to use this as the default root directory.
`-f` *config-file*	Specify an alternate configuration file. The default file is `/etc/vold.conf`.
`-l` *log-file*	Specify an alternate log file. The default log file is `/var/adm/vold.log`.
`-L` *debug-level*	Change the level (verbosity) of debug messages sent to the log file. The range is `0` to `99` where `0` is nothing and `99` is everything. The default level is `0`.

Environment Variables `vold` sets the following environment variables to aid programs which are called when events such as `insert`, `notify`, and `eject` occur:

`VOLUME_ACTION`	Event that caused this program to be executed.
`VOLUME_PATH`	Pathname of the matched *regex* from the `vold.conf` file.
`VOLUME_DEVICE`	Device (in `/vol/dev`) that applies to the media.
`VOLUME_NAME`	Name of the volume in question.
`VOLUME_USER`	User ID of the user causing the event to occur.
`VOLUME_SYMNAME`	Symbolic name of a device containing the volume.
`VOLUME_MEDIATYPE`	Name of the type of media (CD-ROM, floppy or rmdisk)

Files

`/etc/vold.conf`	Volume Management daemon configuration file. Directs the Volume Management daemon to control certain devices, and causes events to occur when specific criteria are met.
`/usr/lib/vold/*.so.1`	Shared objects called by Volume Management daemon when certain actions occur.
`/var/adm/vold.log`	the default log file location (see the `-l` option for a description).
`/vol`	the default Volume Management root directory.

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWvolu, SUNWvolr

See Also volcancel(1), volcheck(1), volmissing(1), rmmount(1M), rpc.smserverd(1M), rmmount.conf(4), vold.conf(4), attributes(5), smf(5), volfs(7FS), usb(7D), scsa1394(7D)

System Administration Guide: Basic Administration

Notes The Volume Management daemon and associated commands might not be included in a future Solaris release.

Name wall – write to all users

Synopsis `/usr/sbin/wall [-a] [-g grpname] [filename]`

Description `wall` reads its standard input until an end-of-file. It then sends this message to all currently logged-in users preceded by:

```
Broadcast Message from . . .
```

If *filename* is given, then the message is read in from that file. Normally, pseudo-terminals that do not correspond to rlogin sessions are ignored. Thus, when using a window system, the message appears only on the console window. However, `-a` will send the message even to such pseudo-terminals.

It is used to warn all users, typically prior to shutting down the system.

The sender must be superuser to override any protections the users may have invoked See `mesg(1)`.

`wall` runs `setgid()` to the group ID `tty`, in order to have write permissions on other user's terminals. See `setuid(2)`.

`wall` will detect non-printable characters before sending them to the user's terminal. Control characters will appear as a " ^ " followed by the appropriate ASCII character; characters with the high-order bit set will appear in "meta" notation. For example, '`\003`' is displayed as '`^C`' and '`\372`' as '`M–z`'.

Options The following options are supported:

`-a` Broadcast message to the console and pseudo-terminals.

`-g` *grpname* Broadcast to the users in a specified group only, per the group database (see `group(4)`).

Environment Variables If the `LC_*` variables (`LC_CTYPE`, `LC_TIME`, `LC_COLLATE`, `LC_NUMERIC`, and `LC_MONETARY`) are not set in the environment, the operational behavior of `wall` for each corresponding locale category is determined by the value of the `LANG` environment variable. See `environ(5)`. If `LC_ALL` is set, its contents are used to override both the `LANG` and the other `LC_*` variables. If none of the above variables are set in the environment, the "C" (U.S. style) locale determines how `wall` behaves.

Files `/dev/tty*`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also mesg(1), write(1), setuid(2), attributes(5), environ(5)

Notes wall displays "Cannot send to . . ." when the open on a user's tty file fails.

Name wanboot_keygen – create and display client and server keys for WAN booting

Synopsis

```
/usr/lib/inet/wanboot/keygen -c -o net=a.b.c.d ,cid=client_ID,type=3des

/usr/lib/inet/wanboot/keygen -c -o net=a.b.c.d ,cid=client_ID,type=aes

/usr/lib/inet/wanboot/keygen -m

/usr/lib/inet/wanboot/keygen -c -o net=a.b.c.d ,cid=client_ID,type=sha1

/usr/lib/inet/wanboot/keygen -d -m

/usr/lib/inet/wanboot/keygen -c -o net=a.b.c.d ,cid=client_ID,type=keytype
```

Description The keygen utility has three purposes:

- Using the `-c` flag, to generate and store per-client 3DES/AES encryption keys, avoiding any DES weak keys.
- Using the `-m` flag, to generate and store a "master" HMAC SHA-1 key for WAN install, and to derive from the master key per-client HMAC SHA-1 hashing keys, in a manner described in RFC 3118, Appendix A.
- Using the `-d` flag along with either the `-c` or `-m` flag to indicate the key repository, to display a key of type specified by *keytype*, which must be one of `3des`, `aes`, or `sha1`.

The `net` and `cid` arguments are used to identify a specific client. Both arguments are optional. If the `cid` option is not provided, the key being created or displayed will have a per-network scope. If the `net` option is not provided, then the key will have a global scope. Default net and code values are used to derive an HMAC SHA-1 key if the values are not provided by the user.

Options The following options are supported:

`-c` Generate and store per-client 3DES/AES encryption keys, avoiding any DES weak keys. Also generates and stores per-client HMAC SHA-1 keys. Used in conjunction with `-o`.

`-d` Display a key of type specified by *keytype*, which must be one of `3des`, `aes`, or `sha1`. Use `-d` with `-m` or with `-c` and `-o`.

`-m` Generate and store a "master" HMAC SHA-1 key for WAN install.

`-o` Specifies the WANboot client and/or keytype.

Examples

EXAMPLE 1 Generate a Master HMAC SHA-1 Key

```
# keygen -m
```

EXAMPLE 2 Generate and Then Display a Client-Specific Master HMAC SHA-1 Key

```
# keygen -c -o net=172.16.174.0,cid=010003BA0E6A36,type=sha1
# keygen -d -c -o net=172.16.174.0,cid=010003BA0E6A36,type=sha1
```

EXAMPLE 3 Generate and Display a 3DES Key with a Per-Network Scope

```
# keygen -c -o net=172.16.174.0,type=3des
# keygen -d -o net=172.16.174.0,type=3des
```

Exit Status

`0` Successful operation.

`>0` An error occurred.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbsup
Interface Stability	Obsolete

See Also

`attributes(5)`

Name wanboot_keymgmt – insert and extract keys

Synopsis

```
/usr/lib/inet/wanboot/keymgmt -i -k key_file -s keystore -o type=keytype

/usr/lib/inet/wanboot/keymgmt -x -f outfile -s keystore -o type=keytype
```

Description The `keymgmt` utility has two purposes:

- To take a raw key, stored in *key_file*, and insert it in the repository specified by *keystore*.
- To extract a key of a specified type from the repository specified by *keystore*, depositing it in *outfile*.

outfile will be created if it does not already exist. The type of key being added or extracted is specified by *keytype* and may have one of four values: `3des`, `aes`, `rsa`, or `sha1` (the last used by HMAC SHA-1). When extracting a key, the first key with an OID matching the supplied type is used.

Arguments The following arguments are supported:

`-i`	Used in conjunction with `-k` to insert a raw key in *keystore*.
`-f` *outfile*	Used to specify a file to receive an extracted key.
`-k` *key_file*	Used in conjunction with `-i` to specify the file in which a raw key is stored. This key will be inserted in *keystore*.
`-o` type=*keytype*	Specifies the type of key being inserted or extracted. Must be one of `3des`, `aes`, `rsa`, or `sha1`.
`-s` *keystore*	Specifies a repository in which a key will be inserted or from which a key will be extracted.
`-x`	Used in conjunction with `-f` to extract a key of a specified type and deposit it in *outfile*.

Exit Status

`0`	Successful operation.
`>0`	An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbsup
Interface Stability	Obsolete

See Also `attributes(5)`

ITU-T Recommendation X.208

Name wanboot_p12split – split a PKCS #12 file into separate certificate and key files

Synopsis

```
/usr/lib/inet/wanboot/p12split -i p12file -c out_cert -k out_key
     [-t out_trust -l id -v]
```

Description The `p12split` utility extracts a certificate and private key from the repository specified by *p12file*, depositing the certificate in *out_cert* and the key in *out_key*. If supplied, the `-l` option specifies the value for the `LocalKeyId` that will be used in the new certificate and key files. `p12split` can optionally extract a trust certificate into the *out_trust* file if the `-t` option is specified. Use the `-v` option to get a verbose description of the split displayed to standard output.

Options The following arguments and options are supported:

`-c` *out_cert* Specifies a repository that receives a extracted certificate.

`-i` *p12file* Specifies a repository from which a certificate and private key is extracted.

`-k` *out_key* Specifies a repository that receives a extracted private key.

`-l` *id* Specifies the value for the `LocalKeyId` that will be used in the new certificate and key files.

`-t` *out_trust* Specifies a file for receiving an extracted trust certificate.

`-v` Displays a verbose description of the split to sdtout.

Exit Status

`0` Successful operation.

`>0` An error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbsup
Interface Stability	Unstable

See Also `attributes(5)`

Name wanbootutil – manage keys and certificates for WAN booting

Synopsis

```
wanbootutil [keygen] [option_specific_arguments]

wanbootutil [keymgmt] [option_specific_arguments]

wanbootutil [p12split] [option_specific_arguments]
```

Description The `wanbootutil` command creates and manages WANboot encyrption and hashing keys and manipulates PKCS #12 files for use by WAN boot.

`wanbootutil` has three subcommands, each covered in a separate man page:

`wanboot_keygen(1M)`	Generates encryption and hashing keys.
`wanboot_keymgmt(1M)`	Inserts and extracts keys from WAN boot key repositories.
`wanboot_p12split(1M)`	Splits a PKCS #12 file into separate certificate and key files for use by WAN boot.

Options The options are supported for `wanbootutil` are the use of `keygen`, `keymgmt`, or `p12split`. The options for these subcommands are described in their respective man pages.

Examples

EXAMPLE 1 Generate a 3DES Client Key

```
# wanbootutil keygen -c -o net=172.16.174.0,cid=010003BA0E6A36,type=3des
```

EXAMPLE 2 Insert an RSA Private Client Key

```
wanbootutil keymgmt -i -k keyfile \
  -s /etc/netboot/172.16.174.0/010003BA0E6A36/keystore -o type=rsa
```

EXAMPLE 3 Split a PKCS #12 File into Certificate and Key Components

```
# wanbootutil p12split -i p12file -c out_cert -k out_key
```

Exit Status

`0`	Successful operation.
`non-zero`	An error occurred. Writes an appropriate error message to standard error.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbsup
Interface Stability	Obsolete

See Also `wanboot_keygen(1M)`, `wanboot_keymgmt(1M)`, `wanboot_p12split(1M)`, `attributes(5)`

Name wbemadmin – start Sun WBEM User Manager

Synopsis

```
/usr/sadm/bin/wbemadmin
```

Description The `wbemadmin` utility starts Sun WBEM User Manager, a graphical user interface that enables you to add and delete authorized WBEM users and to set their access privileges. Use this application to manage access to groups of managed resources, such as disks and installed software, in the Solaris operating environment.

The `wbemadmin` utility allows you to perform the following tasks:

Manage user access rights	Use the `wbemadmin` utility to add, delete, or modify an individual user's access rights to a namespace on a WBEM-enabled system.
Manage namespace access rights	Use the `wbemadmin` utility to add, delete, or modify access rights for all users to a namespace.

The Sun WBEM User Manager displays a Login dialog box. You must log in as root or a user with write access to the `root\security` namespace to grant access rights to users. By default, Solaris users have guest privileges, which grants them read access to the default namespaces.

Managed resources are described using a standard information model called Common Information Model (CIM). A CIM object is a computer representation, or model, of a managed resource, such as a printer, disk drive, or CPU. CIM objects can be shared by any WBEM-enabled system, device, or application. CIM objects are grouped into meaningful collections called schema. One or more schemas can be stored in directory-like structures called namespaces.

All programming operations are performed within a namespace. Two namespaces are created by default during installation:

- `root\cimv2` — Contains the default CIM classes that represent objects on your system.
- `root\security` — Contains the security classes used by the CIM Object Manager to represent access rights for users and namespaces.

When a WBEM client application connects to the CIM Object Manager in a particular namespace, all subsequent operations occur within that namespace. When you connect to a namespace, you can access the classes and instances in that namespace (if they exist) and in any namespaces contained in that namespace.

When a WBEM client application accesses CIM data, the WBEM system validates the user's login information on the current host. By default, a validated WBEM user is granted read access to the Common Information Model (CIM) Schema. The CIM Schema describes managed objects on your system in a standard format that all WBEM-enabled systems and applications can interpret.

You can set access privileges on individual namespaces or for a user-namespace combination. When you add a user and select a namespace, by default the user is granted read access to CIM objects in the selected namespace. An effective way to combine user and namespace access rights is to first restrict access to a namespace. Then grant individual users read, read and write, or write access to that namespace.

You cannot set access rights on individual managed objects. However you can set access rights for all managed objects in a namespace as well as on a per-user basis.

If you log in to the root account, you can set the following types of access to CIM objects:

- Read Only — Allows read-only access to CIM Schema objects. Users with this privilege can retrieve instances and classes, but cannot create, delete, or modify CIM objects.
- Read/Write — Allows full read, write, and delete access to all CIM classes and instances.
- Write — Allows write and delete, but not read access to all CIM classes and instances.
- None — Allows no access to CIM classes and instances.

Context help is displayed in the left side of the `wbemadmin` dialog boxes. When you click on a field, the help content changes to describe the selected field. No context help is available on the main User Manager window.

The `wbemadmin` security administration tool updates the following Java classes in the `root\security` namespace:

- `Solaris_UserAcl` — Updated when access rights are granted or changed for a user.
- `Solaris_namespaceAcl` — Updated when access rights are granted or changed for a namespace.

Usage The `wbemadmin` utility is not the tool for a distributed environment. It is used for local administration on the machine on which the CIM Object Manager is running.

Exit Status The `wbemadmin` utility terminates with exit status `0`.

Warning The `root\security` namespace stores access privileges. If you grant other users access to the `root\security` namespace, those users can grant themselves or other users rights to all other namespaces.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbcor

See Also mofcomp(1M), wbemlogviewer(1M), init.wbem(1M), attributes(5)

Name wbemconfig – convert a JavaSpaces datastore to the newer Reliable Log datastore format

Synopsis

```
/usr/sadm/lib/wbem/wbemconfig convert
```

Description A Reliable Log directory is created that contains the converted data. This directory is named `/var/sadm/wbem/logr`.

The `convert` argument is the only supported option of this command. You should only run this command after stopping WBEM (CIM Object Manager) with the `init.wbem stop` command. Otherwise your data may be corrupted.

This command successfully converts any proprietary custom MOFs you have created in the datastore, but not any CIM or Solaris MOFs you have modified. These will be destroyed. To recompile any modified CIM or Solaris MOFs into the new datastore, run the `mofcomp` command on the MOF files containing the class definitions.

Because the `wbemconfig convert` command invokes the JVM (Java Virtual Machine) to perform conversion of the JavaSpaces datastore, you must be running the same version of the JVM as when the original JavaSpaces storage was created. After the `wbemconfig convert` command is completed, you can change to any version of the JVM you want.

To see what version of the JVM you are running, issue the `java -version` command.

Options The following options are supported:

convert Convert a JavaSpaces datastore to the newer Reliable Log datastore format.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbcou

See Also `init.wbem(1M)`, `wbemadmin(1M)`, `wbemlogviewer(1M)`, `mofcomp(1M)`, `attributes(5)`

Name wbemlogviewer – start WBEM Log Viewer

Synopsis

```
/usr/sadm/bin/wbemlogviewer
```

Description The `wbemlogviewer` utility starts the WBEM Log Viewer graphical user interface, which enables administrators to view and maintain log records created by WBEM clients and providers. The WBEM Log Viewer displays a Login dialog box. You must log in as root or a user with write access to the `root\cimv2` namespace to view and maintain log files. Namespaces are described in `wbemadmin(1M)`.

Log events can have three severity levels.

- Errors
- Warnings
- Informational

The WBEM log file is created in the `/var/sadm/wbem/log` directory, with the name `wbem_log`. The first time the log file is backed up, it is renamed `wbem_log.1`, and a new `wbem_log` file is created. Each succeeding time the `wbem_log` file is backed up, the file extension number of each backup log file is increased by 1, and the oldest backup log file is removed *if* the limit, which in turn is specified in the log service settings, on the number of logfiles is exceeded. Older backup files have higher file extension numbers than more recent backup files.

The log file is renamed with a `.1` file extension and saved when one of the following two conditions occur:

- The current file reaches the specified file size limit.
- A WBEM client application uses the `clearLog()` method in the `Solaris_LogService` class to clear the current log file.
- A WBEM client application uses the `clearLog()` method in the Solaris_LogService class to clear the current log file.
- A user chooses Action->Back Up Now in the Log Viewer application.

Help is displayed in the left panel of each dialog box. Context help is not displayed in the main Log Viewer window.

Usage The WBEM Log Viewer is not the tool for a distributed environment. It is used for local administration.

The WBEM Log Viewer allows you to perform the following tasks:

View the logs

Set properties of log files	Click Action->Log File Settings to specify log file parameters and the log file directory.
Back up a log file	Click Action->Back Up Now to back up and close the current log file and start a new log file.

Open historical log files	Click Action->Open Log File to open a backed-up log file.
Delete an old log file	Open the file and then click Action->Delete Log File. You can only delete backed-up log files.
View log record details	Double-click a log entry or click View->Log Entry Details to display the details of a log record.
Sort the logs	Click View->Sort By to sort displayed entries. You can also click any column heading to sort the list. By default, the log entries are displayed in reverse chronological order (new logs first).

Exit Status The `wbemlogviewer` utility terminates with exit status `0`.

Files `/var/sadm/wbem/log/wbem_log` WBEM log file

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwbcor

See Also `wbemadmin(1M)`, `init.wbem(1M)`, `mofcomp(1M)`, `attributes(5)`

Name wcadmin – manage the configuration of the Sun Java Web Console

Synopsis

```
/usr/sbin/wcadmin subcommand options
/usr/sbin/wcadmin [-h| --help]
/usr/sbin/wcadmin [-V | --version]
```

Description `wcadmin` is a Command Line Interface (CLI) — based tool for managing the configuration of the Sun Java Web Console.

Subcommands The following subcommands are supported:

add — The `wcadmin add` subcommand adds a new shared jar file, a new JAAS login module, or a new shared service property to the console configuration. An optional identifier may be specified; if omitted, the identifier is derived from the resource name. The resource is added when the console is next started.

The format of the `add` subcommand is:

add `-l -a` *appname* [`-n` *id*] *jarpath*

add `-m -a` *appname* [`-n` *id*] `-b` *behavior* `-s` *service* [`-o` *name=value*] *classname*

add `-p -a` *appname name=value* [... *name=value*]

deploy — The `wcadmin deploy` subcommand deploys the specified web application into the console's web server instance. Applications are deployed directly from their installation directories.

The format of the `deploy` subcommand is:

deploy [`-D`] `-a` *appname* `-x` *context app_path*

disable — The `wcadmin disable` subcommand disables access to the specified web application in the console's web server instance.

The format of the `disable` subcommand is:

disable `-x` *context*

enable — The `wcadmin enable` subcommand enables access to the specified web application in the console's web server instance.

The format of the `enable` subcommand is:

enable `-x` *context*

list — The `wcadmin list` subcommand lists the resources currently configured for the console; including deployed web applications, shared jar files, login modules, and shared service properties. If no option is specified, all resources are listed.

The format of the `list` subcommand is:

list [`-a`] [`-d`] [`-l`] [`-m`] [`-p`]

password — The `wcadmin password` subcommand manages the administrator and security keystore passwords for the console. Keystore passwords should not be changed while the console is running.

The format of the `password` subcommand is:

password [`-a`] [`-k`] [`-t`]

password `-f` *password_file*

reload — The `wcadmin reload` subcommand unloads the specified web application from the console's web server instance and reloads the application from its original installation directory.

The format of the `reload` subcommand is:

reload `-x` *context*

remove — The `wcadmin remove` subcommand removes a shared jar file, a login module, or a shared service property from the console configuration. The resource may be specified by its identifier or by its full jarpath or classname. The resource is removed when the console is next started.

The format of the `remove` subcommand is:

remove `-l -a` *appname* [`-n` *id*] *jarpath*

remove `-m -a` *appname* [`-n` *id*] *classname*

remove `-p -a` *appname property* [...]

undeploy — The `wcadmin undeploy` subcommand undeploys the specified web application from the console's web server instance.

The format of the `undeploy` subcommand is:

undeploy [`-D`] `-a` *appname* `-x` *context*

Options The following options are supported:

`-h` | `--help` | `-?` — Display runtime help.

`-V` | `--version` — Display console version information.

`-D` | `--defer` — When used with the deploy and undeploy subcommands, defers the deployment or undeployment until the next console restart.

The operation is deferred by simply adding or removing the corresponding resource registration notification file. If defer is not specified, a runtime deployment or undeployment is performed, so that the application becomes available or unavailable in the running console. If the console instance is not currently running, the operation is automatically deferred.

`-a|--adminpassword` Specify that the administrator password should be changed, when used with password subcommand. You are prompted for a new password, which must be 8 to 32 characters.

`-a|--application` Specify the application name, when used with subcommands other than password subcommand.

`-d|--detail` Specify that configuration details of each resource should be displayed.

`-f` Specify the fully qualified path name to a file containing one or more password property values. See the description of the password_file argument.

`-k|--keypassword` Specify that the keystore password should be changed. You are prompted for a new password, which must be 8 to 32 characters.

`-l|--library` Specify that the resource is a shared jar file.

`-m|--module` Specify that the resource is a JAAS login module.

`-n|--name` Specify the short-hand identifier name for the resource. If omitted, the identifier name is derived from the full resource name.

`-o|--option` Specify the name and value of a login module option property, separated by the equals character.

`-p|--property` Specify that the resource is one or more shared service properties.

`-s|--service` Specify the name of the JAAS login service definition. If omitted, the default console login service definition is assumed.

`-t|--trustpassword` Specify that the truststore password should be changed. You are prompted for a new password, which must be 8 to 32 characters.

`-x|--context` Specify the web application context path name under which the application is deployed.

Arguments

app_path The fully qualified file system path to the web application installation directory.

appname The application name. The name must be unique among all web applications registered with the console. It is also used as the name of the subdirectory under the console's pre-registration directory which contains

all the resource registration notification files for that application. Typically, the application package name, plugin identifier, or context path name is specified for the application name.

behavior The JAAS login module control flag behavior. Must be one of "optional", "required", "requisite", or "sufficient".

classname The fully qualified Java package class name of the JAAS login module. The specified class must be included in a shared jar file added to the console.

context The web application context path name under which the application is deployed. With the .reg suffix, the context forms the file name of the registration notification file for that application.

id The short-hand identifier name for a jar file or login module resource to be added or removed. The identifier name must be unique among the resources shared for a given application name. With the .reg suffix, it forms the file name of the registration notfication file for that resource.

jarpath The fully qualified file system path to the jar file resource. When the resource is added to the console, its path is included in the classpath of the console's shared class loader.

option The JAAS login module option property name.

property The shared service property name.

password_file The fully qualified path to a password text file that contains the new administrator, keystore, and truststore passwords in property file format. The administrator password is specified using the "adminpassword" property. The keystore password is specified using the "keypassword" property. The truststore password is specified using the "trustpasswd" property. At least one password property must be contained in the password file.

value The login module option or shared service property value. If the value contains white space, it must be quoted.

Examples The following command adds a jar file to be shared in the console:

```
wcadmin add -l -a myapp_1.0 -n wbem /usr/sadm/lib/wbem.jar
```

The following command deploys a new web application:

```
wcadmin deploy -a myapp_1.0 -x myapp /opt/SUNWmyapp/myapp
```

The following command reloads an existing web application:

```
wcadmin reload -x myapp
```

The following command undeploys a web application at the next server restart:

```
wcadmin undeploy -D -a myapp_1.0 -x myapp
```

The following command lists all the deployed web applications in the console. If the status field is "running", the web application is available. If the status field is "stopped", the web application is disabled and is not available. If all web applications are "stopped", this typically indicates the console web server instance is not running.

```
wcadmin list -a
```

The following command removes a shared jar file:

```
wcadmin remove -l -a myapp_1.0 -n wbem
```

The following command changes passwords that are specified in a file:

```
wcadmin password -f /home/mydir/console-passwords
```

Exit Status The following exit values are returned:

`0` Subcommand succeeded without error

1 Usage error: missing or malformed arguments

2 Fatal error: subcommand failed with one or more errors

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWmcon
Interface Stability	Stable

See Also `smcwebserver(1M)`, `smreg(1M)`, `attributes(5)`, `environ(5)`

Name whodo – who is doing what

Synopsis

```
/usr/sbin/whodo [-h] [-l] [user]
```

Description The `whodo` command produces formatted and dated output from information in the `/var/adm/utmpx` and `/proc/pid` files.

The display is headed by the date, time, and machine name. For each user logged in, device name, user-ID and login time is shown, followed by a list of active processes associated with the user-ID. The list includes the device name, process-ID, CPU minutes and seconds used, and process name.

If *user* is specified, output is restricted to all sessions pertaining to that user.

Options The following options are supported:

`-h` Suppress the heading.

`-l` Produce a long form of output. The fields displayed are: the user's login name, the name of the tty the user is on, the time of day the user logged in (in *hours*:*minutes*), the idle time — that is, the time since the user last typed anything (in *hours*:*minutes*), the CPU time used by all processes and their children on that terminal (in *minutes*:*seconds*), the CPU time used by the currently active processes (in *minutes*:*seconds*), and the name and arguments of the current process.

Examples **EXAMPLE 1** Using the whodo Command

The command:

```
example% whodo
```

produces a display like this:

```
Tue Mar 12 15:48:03 1985
bailey
tty09    mcn       8:51
    tty09   28158    0:29 sh

tty52    bdr      15:23
    tty52   21688    0:05 sh
    tty52   22788    0:01 whodo
    tty52   22017    0:03 vi
    tty52   22549    0:01 sh

xt162    lee      10:20
    tty08    6748    0:01 layers
    xt162    6751    0:01 sh
    xt163    6761    0:05 sh
    tty08    6536    0:05 sh
```

Environment Variables If any of the LC_* variables (LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY) (see environ(5)) are not set in the environment, the operational behavior of tar(1) for each corresponding locale category is determined by the value of the LANG environment variable. If LC_ALL is set, its contents are used to override both the LANG and the other LC_* variables. If none of the above variables is set in the environment, the C (U.S. style) locale determines how whodo behaves.

LC_CTYPE — Determines how whodo handles characters. When LC_CTYPE is set to a valid value, whodo can display and handle text and filenames containing valid characters for that locale. The whodo command can display and handle Extended Unix code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. whodo can also handle EUC characters of 1, 2, or more column widths. In the C locale, only characters from ISO 8859-1 are valid.

LC_MESSAGES — Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the C locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

LC_TIME — Determines how whodo handles date and time formats. In the C locale, date and time handling follow the U.S. rules.

Exit Status The following exit values are returned:

0 — Successful completion.

non-zero — An error occurred.

Files

/etc/passwd — System password file

/var/adm/utmpx — User access and administration information

/proc/pid — PID information for individual users

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also ps(1), who(1), attributes(5), environ(5)

Name wracct – write extended accounting records for active processes and tasks

Synopsis

```
/usr/sbin/wracct -i id_list [-t record_type]
     {process | task}
```

Description The `wracct` utility allows the administrator to invoke the extended accounting system, if active, to write intermediate records representing the resource usage of a selected set of processes or tasks. For tasks, a *record_type* option is also supported, allowing the administrator to request the writing of:

- an interval record, which reflects task usage since a previous interval record (or since task creation if there is no interval record), or
- a partial record, which reflects usage since task creation.

Options The following options are supported:

`-i` *id_list* Select the IDs of the tasks or processes to write records for. Specify *id_list* as a comma- or space-separated list of IDs, presented as a single argument. For some shells, this requires appropriate quoting of the argument.

`-t` *record_type* Select type of record to write for the selected task or process. For tasks, *record_type* can be `partial` or `interval`. `partial` is the default type, and the only type available for process records.

Operands The following operands are supported:

`process` Treat the given ID as a process ID for the purposes of constructing and writing an extended accounting record.

`task` Treat the given ID as a task ID for the purposes of constructing and writing an extended accounting record.

Examples **EXAMPLE 1** Writing a Partial Record

Write a partial record for all active `sendmail` processes.

```
# /usr/sbin/wracct -i "‘pgrep sendmail‘" process
```

EXAMPLE 2 Writing an Interval Record

Write an interval record for the task with `ID 182`.

```
# /usr/sbin/wracct -t interval -i 182 task
```

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

`2` Invalid command line options were specified.

3 Pertinent components of extended accounting facility are not active.

Files `/var/adm/exacct/task`

`/var/adm/exacct/proc` Extended accounting data files.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

See Also `acctadm(1M)`, `attributes(5)`

Name wrsmconf – manage WCI RSM controller configurations

Synopsis

```
/usr/platform/sun4u/sbin/wrsmconf create -c controller_id -f config_file

wrsmconf initial -f config_file [-c controller_id]

wrsmconf remove [-c controller_id]

wrsmconf topology [-c controller_id]

wrsmconf dump -c controller_id -f config_file
```

Description `wrsmconf` provides a means to create, install, retrieve, and remove configurations for WCI remote shared memory (RSM) controllers.

Options The following options are supported:

`create -c` *controller_id* `-f` *config_file*

Create a set of per-node configurations for the specified controller and store them to the file *config_file*. The created file contains a per-node configuration for each node specified in the input for the specified controller. The file has a checksum on it and cannot be modified directly. This file can be used in a `wrsmconf initial` call on each node to install the node's configuration into the local driver.

A list of nodes and WCI devices connected to those nodes is read from standard input. For each connected pair of links, specify the nodename (`uname -n`), safari port id, and link number on both sides of the connection. The format of the information looks like this:

```
<nodename>.<wrsm-portid>.<linkno>=<nodename>.<wrsm-portid>.<linkno>
<nodename>.<wrsm-portid>.<linkno>=<nodename>.<wrsm-portid>.<linkno>
```

This interface is intended for installing configurations for testing (such as for SunVTS). Only direct-connect, non-striped configurations for 1 to 3 nodes can be specified. FM node ids and RSM hardware addresses are assigned to the specified nodes contiguously and in order starting from 0.

`initial -f` *config_file* [`-c` *controller_id*]

Install the configuration for the local node stored in the file *config_file* into the driver as the initial configuration for the specified controller. This command fails under the following circumstances:

- If *controller_id* is specified and the configuration in the file is not for the specified controller.
- If the file does not contain a valid configuration for the local node or if the checksum in the file shows it has been modified.
- If a configuration has already been installed for the controller. If this happens, use `wrsmconf remove` to remove the existing configuration.

`remove` [`-c` *controller_id*]

Disable communication through the installed configuration for all controllers or the specified controller and remove the configuration(s) from the driver.

`topology -c` *controller_id*

For each installed controller (or for the specified controller), print to stdout the set of nodes this controller is configured to reach, including the nodename, FM node id, and RSM hardware address for each node. The following is example output:

```
FM Node ID   Node Name  Controller Instance    Controller HW Addr
0            hpc00              0                    101
0            hpc00              1                    333
1            hpc01              0                    102
1            hpc01              1                    54
1            hpc01              2                    34
2            hpc03              0                    103
2            hpc03              1                    103
2            hpc03              2                    103
```

`dump -c` *controller-id* `-f` *config_file*

Fetch the installed configuration for the specified controller from the driver and store it into the file *config_file* along with a checksum to protect the data. This configuration can later be installed with the command `wrsmconf initial`.

Exit Status This command returns 0 on successful completion, and a non-zero value if an error occured.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwrsm

See Also kstat(1M), wrsmstat(1M), attributes(5)

Name wrsmstat – report WCI RSM driver statistics

Synopsis

```
/usr/platform/sun4u/sbin/wrsmstat controller
     [-c controller_id]

wrsmstat wrsm [-i wrsm_instance_num] [-v]

wrsmstat route [-c controller_id]
     [-h node_hostname]

wrsmstat set [-i wrsm_instance_num] -c cmmu -s start -e end
```

Description The `wrsmstat` command provides statistics on remote shared memory (RSM) controllers, routes to nodes, and WCI interfaces managed by the WCI RSM driver (`wrsm`). It also provides an interface for setting extended performance counter control registers that constrain the `wrsm` counters available through `busstat(1M)`.

Options The following options are supported:

`controller` [`-c` *controller_id*]
Displays information describing the state of the specified controller, or of all controllers if none is specified. The following is sample output:

$ wrsmstat controller -c 5 Controller 5 ---------- Controller state: up Local RSM Hardware address: 0x4 Exported segments: 0 # published: 0 # connections: 0 total bound memory: 0 Imported segments: 0 Send Queues: 0 Registered Handlers: 0 Assigned WCIs: 4 Available WCIs: 2

`wrsm` [`-i` *wrsm_instance_num*] [`-v`]
Displays information describing the state of the specified RSM WCI, or of all RSM WCIs if none is specified. The following is sample output:

```
$ wrsmstat wrsm -i 2
    WCI instance 2
    --------------
    Portid: 5
    Controller ID: 0
    Config Version: 5
    Link Error Shutdown Trigger: 40000
    Link #0 is not present.
    Link #1
        Link Enabled: yes
        Link State: up
        Remote RSM HW addr: 1
        Remote wnode ID: 1
        Remote link num: 1
        Remote WCI port ID: 3
        Error takedowns: 0
        Bad Config takedowns: 0
        Failed bringups: 0
```

```
        Total link errors: 0
        Maximum link errors: 0
        Average link errors: 0
        Auto shutdown enabled: yes
    Link #2 is not present.
```

If you specify the `-v` option, the following additional information is displayed:

```
Cluster Error Count:        0
Uncorrectable SRAM ECC error:   no
Maximum SRAM ECC errors:    0
Average SRAM ECC errors:    0
```

`route` [`-c` *controller_id*] [`-h` *nodename*]
Displays the route to the specified node through the specified controller. If no node is specified, displays the routes to all nodes. If no controller is specified, displays the specified node's route through all controllers. If neither is specified, displays the routes to all nodes through all controllers. The following is sample output:

```
$ wrsmstat -c 3 -h fred
Controller 3 - Route to fred
----------------------------
Config Version: 1
FM node id: 0x345543
RSM hardware address: 0x9
Route Changes: 3
Route Type: Passthrough
Number of WCIs: 2
Stripes: 4
WCI #0
    Port ID: 3
    Instance: 0
    Num of hops: 2
    Num of links: 2
        link# 1, first hop RSM HW addr: 0x4
        link# 2, first hop RSM HW addr: 0x2
WCI #1
    Port ID: 13
    Instance: 1
    Num of hops: 2
    Num of links: 2
        link# 0, first hop RSM HW addr: 0x4
        link# 2, first hop RSM HW addr: 0x2
```

`set` [`-i` *wrsm_instance_num*] `-c cmmu -s` <*start*> `-e` <*end*>
For the specified WCI (or for each RSM WCI if none specified), configure the specified range of CMMU entries so that transactions through them are counted by *wrsm* `busstat kstats`. Each call will cause transactions to be counted through the new ranges of `cmmu` entries in addition to the previously specified ranges. To count transactions through all CMMUs, specify a start value of 0 and end value of 0. To clear all ranges (and not count

transactions through any `cmmu` entries), specify a start value of 0 and end value of -1.

Exit Status This command returns 0 on successful completion, and a non-zero value if an error occurred.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWwrsm

See Also `busstat(1M)`, `kstat(1M)`, `wrsmconf(1M)`, `attributes(5)`

Name xntpd – Network Time Protocol daemon

Synopsis

```
/usr/lib/inet/xntpd [-aAbdm] [-c conffile] [-e authdelay]
     [-f driftfile] [-k keyfile] [-l logfile] [-p pidfile]
     [-r broadcastdelay] [-s statsdir] [-t trustedkey]
     [-v variable] [-V variable]
```

Description `xntpd` is a daemon which sets and maintains a UNIX system time-of-day in agreement with Internet standard time servers. `xntpd` is a complete implementation of the Network Time Protocol (NTP) version 3 standard, as defined by *RFC 1305*. It also retains compatibility with version 1 and 2 servers as defined by *RFC 1059* and *RFC 1119*, respectively. The computations done in the protocol and clock adjustment code are carried out with high precision and with attention to the details which might introduce systematic bias into the computations. This is done to try to maintain an accuracy suitable for synchronizing with even the most precise external time source.

Ordinarily, `xntpd` reads its configuration from a configuration file at startup time. The default configuration file name is `/etc/inet/ntp.conf`, although this may be overridden from the command line. It is also possible to specify a working, although limited, `xntpd` configuration entirely on the command line, obviating the need for a configuration file. This may be particularly appropriate when `xntpd` is to be configured as a broadcast or multicast client, with all peers being determined by listening to broadcasts at run time. Through the use of the `ntpq(1M)` program, various internal `xntpd` variables can be displayed and configuration options altered while the daemon is running.

The daemon can operate in any of several modes, including symmetric active/passive, client/server and broadcast/multicast. A broadcast/multicast client can automatically discover remote servers, compute one-way delay correction factors and configure itself automatically. This makes it possible to deploy a fleet of workstations without specifying a configuration file or configuration details specific to its environment.

Options The following command line arguments are understood by `xntpd`. See `Configuration Commands` for a more complete functional description:

`-a`	Run in authentication mode.
`-A`	Disable authentication mode.
`-b`	Listen for broadcast NTP and sync to this if available.
`-c` *conffile*	Specify an alternate configuration file.
`-d`	Specify debugging mode. This flag may occur multiple times, with each occurrence indicating greater detail of display.
`-e` *authdelay*	Specify the time (in seconds) it takes to compute the NTP encryption field on this computer.
`-f` *driftfile*	Specify the location of the drift file.

`-k` *keyfile*	Specify the location of the file which contains the NTP authentication keys.
`-l` *logfile*	Specify a log file instead of logging to syslog.
`-m`	Listen for multicast messages and synchronize to them if available (requires multicast kernel).
`-p` *pidfile*	Specify the name of the file to record the daemon's process id.
`-r` *broadcast*	Ordinarily, the daemon automatically compensates for the network delay between the broadcast/multicast server and the client; if the calibration procedure fails, use the specified default delay (in seconds).
`-s` *statsdir*	Specify the directory to be used for creating statistics files.
`-t` *trustedkey*	Add a key number to the trusted key list.
`-v` *variable*	Add a system variable.
`-V` *variable*	Add a system variable listed by default.

Usage `xntpd`'s configuration file format is similar to other Unix configuration files. Comments begin with a '#' character and extend to the end of the line. Blank lines are ignored. Configuration commands consist of an initial keyword followed by a list of arguments, separated by whitespace. Some arguments may be optional. These commands may not be continued over multiple lines. Arguments may be host names, host addresses written in dotted–decimal, integers, floating point numbers (when specifying times in seconds) and text strings.

Configuration Commands

In the following descriptions, optional arguments are delimited by '[ ]', while alternatives are separated by '|'. The first three commands specify various time servers to be used and time services to be provided.

`peer` *host_address* [`key` #] [`version` #][`prefer`]

Specifies that the local server is to operate in "symmetric active" mode with the remote server *host_address* named in the command. In this mode, the local server can be synchronized to the remote server. In addition, the remote server can be synchronized by the local server. This is useful in a network of servers where, depending on various failure scenarios, either the local or remote server host may be the better source of time. The `peer` command, and the `server` and `broadcast` commands that follow, can take the following arguments:

`key`	Indicates that all packets sent to the address are to include authentication fields, encrypted using the specified key number. The range of this number is that of an unsigned 32 bit integer. By default, an encryption field is not included.
`version`	Specifies the version number to be used for outgoing NTP packets. Versions 1, 2, and 3 are the choices; version 3 is the default.
`prefer`	Marks the host as a preferred host. This host will be preferred for synchronization over other comparable hosts.

server *host_address* [key #] [version f1#] [prefer] [mode f1#] server	Specifies that the local server is to operate in "client" mode with the remote server named in the command. In this mode the local server can be synchronized to the remote server, but the remote server can never be synchronized to the local server.
broadcast *host_address* [key #] [version #] [ttl #]	Specifies that the local server is to operate in "broadcast" mode where the local server sends periodic broadcast messages to a client population at the broadcast/multicast address named in the command. Ordinarily, this specification applies only to the local server operating as a transmitter. For operation as a broadcast client, see `broadcastclient` or `multicastclient` commands elsewhere in this document. In

broadcast mode the *host_address* is usually the broadcast address on a local network or a multicast address assigned to NTP. The IANA has assigned the network, 224.0.1.1 to NTP. This is presently the only network that should be used. The following option is used only with the broadcast mode:

`ttl` Specifies the time-to-live (TTL) to use on multicast packets. Selection of the proper value, which defaults to 127, is something of a black art and must be coordinated with the network administrator(s).

`broadcastclient` Directs the local server to listen for broadcast messages on the local network, in order to discover other servers on the same subnet. Upon hearing a broadcast message for the first time, the local server measures the nominal network delay using a brief client/server exchange with the remote server. Then the server enters the "broadcastclient" mode, in which it listens for and synchronizes to succeeding broadcast messages. In order to avoid accidental or malicious disruption in this mode, both the local and remote servers must operate using authentication, with the same trusted key and key identifier.

`multicastclient` [*IP address* . . .] Used in the same way as the `broadcastclient` command, but operates using IP multicasting. Support for this command requires the use of authentication. If one or more IP addresses are given, the server joins the respective multicast group(s).

If none are given, the IP address assigned to NTP (224.0.1.1) is assumed.

`driftfile` *filename*

Specifies the name of the file used to record the frequency offset of the local clock oscillator. If the file exists, it is read at startup in order to set the initial frequency offset. Then the file is updated once per hour with the current offset computed by the daemon. If the file does not exist or this command is not given, the initial frequency offset is assumed to be zero. In this case, it may take some hours for the frequency to stabilize and the residual timing errors to subside. The file contains a single floating point value equal to the offset in parts-per-million (ppm). The file is updated by first writing the current drift value into a temporary file and then using `rename(2)` to replace the old version. This implies that `xntpd` must have write permission for the directory the drift file is located in, and that file system links, symbolic or otherwise, should probably be avoided.

`enable`
`auth | bclient | pll | monitor | stats [...]`
`disable`
`auth | bclient | pll | monitor | stats`
`[...]`

Provides a way to enable or disable various server options. To do so, execute a two word command, where the first word is `enable` or `disable` and the second is the flag. Flags not mentioned are unaffected. Flags that can be changed are described below, along with their default values.

Flag	Default	Description
auth	disable	Causes the server to synchronize with unconfigured peers only if the peer has been correctly authenticated using a trusted key and key identifier.
bclient	disable	Causes the server to listen for a message from a broadcast or multicast server. After this occurs, an association is automatically instantiated for that server. default for this flag is disable (off).
pll	enable	Enables the server to adjust its local clock. If not set, the local clock free-runs at its intrinsic time and frequency offset. This flag is useful in case the local clock is controlled by some other device or protocol and NTP is used only to provide synchronization to other clients.
monitor	disable	Enables the monitoring facility (see elsewhere).
stats	enable	Enables statistics facility filegen (see `Monitoring Commands` below).

`slewalways` [`yes`|`no`] Force `xntpd` to always slew the time.

Authentication Commands

`keys` *filename* Specifies the name of a file which contains the encryption keys and key identifiers used by `xntpd` when operating in authenticated mode. The format of this file is described later in this document.

`trustedkey` # [# . . .] Specifies the encryption key identifiers which are trusted for the purposes of authenticating peers suitable for synchronization. The

authentication procedures require that both the local and remote servers share the same key and key identifier, defined to be used for this purpose. However, different keys can be used with different servers. The arguments are 32 bit unsigned integers. Note, however, that key `0` is fixed and globally known. If meaningful authentication is to be performed, the `0` key should not be trusted.

`controlkey #` Specifies the key identifier to use with the `ntpq`(1M) program, which is useful to diagnose and repair problems that affect `xntpd` operation. The operation of the `ntpq` program and `xntpd` conform to those specified in *RFC 1305*. Requests from a remote `ntpq` program which affect the state of the local server must be authenticated. This requires that both the remote program and local server share a common key and key identifier. The argument to this command is a 32 bit unsigned integer. If no `controlkey` command is included in the configuration file, or if the keys don't match. These requests are ignored.

Access Control Commands

`restrict` *address* [`mask` *numeric_mask*] [*flag*] [. . .]

`xntpd` implements a general purpose address–and–mask based restriction list. The list is sorted by IP address and mask, and the list is searched in this order for matches, with the last match found defining the restriction flags associated with the incoming packets. The source address of incoming packets is used for the match, with the 32 bit address being logically and'ed with the mask associated with the restriction entry and then compared with the entry's address (which has also been and'ed with the mask) to look for a match. The "mask" argument defaults to 255.255.255.255, meaning that the "address" is treated as the address of an individual host. A default entry (address 0.0.0.0, mask 0.0.0.0) is always included and, given the sort algorithm, is always the first entry in the list. Note that, while "address" is normally given in dotted–quad format, the text string "default", with no mask option, may be used to indicate the default entry.

In the current implementation, flags always restrict access, i.e., an entry with no flags indicates that free access to the server is to be given. The flags are not orthogonal, in that more restrictive flags often make less restrictive ones redundant. The flags can generally be classed into two categories, those which restrict time service and those which restrict informational queries and attempts to do run time reconfiguration of the server.

One or more of the following flags may be specified:

`ignore` Ignore all packets from hosts which match this entry. If this flag is specified neither queries nor time server polls will be responded to.

`noquery` Ignore all NTP mode 7 packets (i.e., information queries and configuration requests) from the source. Time service is not affected.

`nomodify`	Ignore all NTP mode 7 packets which attempt to modify the state of the server (i.e., run time reconfiguration). Queries which return information are permitted.
`notrap`	Decline to provide mode 6 control message trap service to matching hosts. The trap service is a subsystem of the mode 6 control message protocol which is intended for use by remote event logging programs.
`lowpriotrap`	Declare traps set by matching hosts to be low priority. The number of traps a server can maintain is limited. The current limit is 3. Traps are usually assigned on a first come, first served basis, with later trap requestors being denied service. This flag modifies the assignment algorithm by allowing low priority traps to be overridden by later requests for normal priority traps.
`noserve`	Ignore NTP packets whose mode is other than 7. In effect, time service is denied, though queries may still be permitted.
`nopeer`	Provide stateless time service to polling hosts, but do not allocate peer memory resources to these hosts even if they otherwise might be considered useful as future synchronization partners.
`notrust`	Treat these hosts normally in other respects, but never use them as synchronization sources.
`limited`	These hosts are subject to a limitation on number of clients from the same net that will be accepted. Net in this context refers to the IP notion of net (class A, class B, class C, etc.). Only the first *client_limit* hosts that have shown up at the server and that have been active during the last *client_limit_period* seconds are accepted. Requests from other clients from the same net are rejected. Only time request packets are taken into account. "Private", "control", and "broadcast" packets are not subject to client limitation and therefore do not contribute to client count. A history of clients is kept using the monitoring capability of `xntpd`. Thus, monitoring is active as long as there is a restriction entry with the `limited` flag. The default value for *client_limit* is 3. The default value for *client_limit_period* is 3600 seconds. Currently both variables are not runtime configurable.
`ntpport`	This is actually a match algorithm modifier, rather than a restriction flag. Its presence causes the restriction entry to be matched only if the source port in the packet is the standard

NTP UDP port (123). Both `ntpport` and `non–ntpport` may be specified. The `ntpport` is considered more specific and is sorted later in the list.

Default restriction list entries, with the flags, `ignore`, `ntpport`, for each of the local host's interface addresses are inserted into the table at startup to prevent the server from attempting to synchronize to its own time. A default entry is also always present, though if it is otherwise unconfigured no flags are associated with the default entry (i.e., everything besides your own NTP server is unrestricted).

The restriction facility was added to allow the current access policies of the time servers running on the NSF net backbone to be implemented with `xntpd` as well. This facility may be useful for keeping unwanted or broken remote time servers from affecting your own. However, it should not be considered an alternative to the standard NTP authentication facility.

`clientlimit limit` Sets *client_limit* to `limit`; allows configuration of client limitation policy. This variable defines the number of clients from the same network that are allowed to use the server.

`clientperiod period` Sets *client_limit_period*; allows configuration of client limitation policy. This variable specifies the number of seconds after which a client is considered inactive and thus no longer is counted for client limit restriction.

Monitoring Commands `statsdir` */directory path/*

Indicates the full path of a directory where statistics files should be created (see below). This keyword allows the (otherwise constant) filegen filename prefix to be modified for file generation sets used for handling statistics logs (see `filegen` statement below).

`statistics` *name . . .*

Enables writing of statistics records. Currently, three kinds of statistics are supported. Each type is described below by giving its *name*, a sample line of data, and an explanation of each field:

`loopstats` enables recording of loop filter statistics information. Each update of the local clock outputs a line of the following form to the file generation set named “loopstats”:

```
48773  10847.650  0.0001307  17.3478  2
```

`Field No.`	`Description`
1	The date (Modified Julian day)
2	The time (seconds and fraction past UTC midnight)

3	Time offset in seconds
4	Frequency offset in parts-per-million
5	Time constant of the clock-discipline algorithm at each update of the clock

`peerstats` enables recording of peer statistics information. This includes statistics records of all peers of a NTP server and of the 1-pps signal, where present and configured. Each valid update appends a line similar to the one below, to the current element of a file generation set named "peerstats":

```
48773  10847.650  127.127.4.1  9714  -0.001605  \
     0.00000  0.00142
```

`Field No.`	`Description`
1	The date (Modified Julian Day)
2	The time (seconds and fraction past UTC midnight)
3	The peer address in dotted-quad notation
4	peer status. The status field is encoded in hex in the format described in Appendix B.2.2 of the NTP specification, *RFC 1305*.
5	Offset in seconds
6	Delay in seconds
7	Dispersion in seconds

`clockstats` enables recording of clock driver statistics information. Each update received from a clock driver outputs a line of the following form to the file generation set named "clockstats":

```
49213  525.624  127.127.4.1  93  226  \
   00:08:29.606  D
```

`Field No.`	`Description`
1	The date (Modified Julian Day)
2	The time (seconds and fraction past UTC midnight)
3	The clock address in dotted-quad notation
4	The last timecode received from the clock in decoded ASCII format, where meaningful

In some clock drivers a good deal of additional information can be gathered and displayed as well.

Statistic files are managed using file generation sets (see `filegen` below). The information obtained by enabling statistics recording allows analysis

of temporal properties of a `xntpd` server. It is usually only useful to primary servers or maybe main campus servers.

`filegen` *name* [`file` *filename*] [`type` *typename*][`flag` *flagval*] [`link` | `nolink`] [enable | disable]

Configures setting of generation file set *name*. Generation file sets provide a means for handling files that are continuously growing during the lifetime of a server. Server statistics are a typical example for such files. Generation file sets provide access to a set of files used to store the actual data. At any time at most one element of the set is being written to. The `type` given specifies when and how data will be directed to a new element of the set. This way, information stored in elements of a file set that are currently unused are available for administrational operations without the risk of disturbing the operation of `xntpd`. (Most important: they can be removed to free space for new data produced.)

Filenames of set members are built from three elements:

prefix This is a constant filename path. It is not subject to modifications via the `filegen` statement. It is defined by the server, usually specified as a compile time constant. It may, however, be configurable for individual file generation sets via other commands. For example, the prefix used with "loopstats" and "peerstats" filegens can be configured using the `statsdir` statement explained above.

filename This string is directly concatenated to the *prefix* mentioned above (no intervening '/' (slash)). This can be modified using the `file` argument to the `filegen` statement. No '. .' elements are allowed in this component to prevent filenames referring to parts outside the filesystem hierarchy denoted by *prefix*.

suffix This part is reflects individual elements of a file set. It is generated according to the `type` of a file set as explained below.

A file generation set is characterized by its type. The following types are supported:

`none` The file set is actually a single plain file.

`pid` One element of file set is used per incarnation of a `xntpd` server. This type does not perform any changes to file set members during runtime. However it provides an easy way of separating files belonging to different `xntpd` server incarnations. The set member filename is built by appending a '.' (dot) to concatenated *prefix* and *filename* strings, and appending the decimal representation of the process id of the `xntpd` server process.

`day` One file generation set element is created per day. The term *day* is based on UTC . A day is defined as the period between 00:00 and 24:00 UTC . The file set member suffix consists of a '.' (dot) and a day specification in the form, *YYYYMMDD*. *YYYY* is a 4 digit year number (e.g., 1992). *MM* is a two digit month number. *DD* is a two digit day number. Thus, all information written at December 10th, 1992 would end up in a file named, *PrefixFilename*.19921210.

`week` Any file set member contains data related to a certain week of a year. The term *week* is defined by computing "day of year" modulo 7. Elements of such a file generation set are distinguished by appending the following suffix to the file set filename base: a dot, a four digit year number, the letter 'W', and a two digit week number. For example, information from January, 5th 1992 would end up in a file with suffix ".1992W1".

`month` One generation file set element is generated per month. The file name suffix consists of a dot, a four digit year number, and a two digit month.

`year` One generation file elment is generated per year. The filename suffix consists of a dot and a 4 digit year number.

`age` This type of file generation sets changes to a new element of the file set every 24 hours of server operation. The filename suffix consists of a dot, the letter 'a', and an eight digit number. This number is taken to be the number of seconds the server is running at the start of the corresponding 24 hour period.

Information is only written to a file generation set when this set is `enabled`. Output is prevented by specifying, `disabled`.

It is convenient to be able to access the current element of a file generation set by a fixed name. This feature is enabled by specifying `link` and disabled using `nolink`. If `link` is specified, a hard link from the current file set element to a file without suffix is created. When there is already a file with this name and the number of links of this file is one, it is renamed appending a dot, the letter, 'C', and the pid of the `xntpd` server process. When the number of links is greater than one, the file is unlinked. This allows the current file to be accessed by a constant name.

Miscellaneous Commands

`broadcastdelay` *seconds*

The broadcast and multicast modes require a special calibration to determine the network delay between the local and remote servers. Ordinarily, this is done automatically by the initial protocol exchanges between the local and remote servers. In some cases, the calibration procedure may fail due to, for example, network or server access controls. This command specifies the default delay to be used under these circumstances. Typically (for Ethernet), a number between 0.003 and 0.007 is appropriate for *seconds*. When this command is not used, the default is 0.004 seconds.

`trap` *host_address* [`port` *port_number*] [`interface` *interface_address*]

Configures a trap receiver at the given *host_address* and *port_number* for sending

messages with the specified local *interface_address*. If the port number is unspecified, a value of 18447 is used. If the interface address is not specified, the message is sent with the source address of the local interface the message is sent through. On a multi-homed host, the interface used may change with routing changes.

C information from the server in a log file. While such monitor programs may also request their own trap dynamically, configuring a trap receiver ensures that no messages are lost when the server is started.

`setvar` *variable* [*default*]

This command adds an additional system variable. Variables like this can be used to distribute additional information such as the access policy. If the variable of the form, *variable_name=value* is followed by the *default* keyword, the variable will be listed as one of the default system variables (see the `ntpq`(1M) command). Additional variables serve informational purposes only. They can be listed; but they are not related to the protocol. The known protocol variables always override any variables defined via the `setvar` mechanism.

Three special variables contain the names of all variable of the same group. *sys_var_list* holds the names of all system variables. *peer_var_list* holds the names of all peer variables. And *clock_var_list* hold the names of the reference clock variables.

`monitor` [`yes` | `no`]
`authenticate` [`yes` | `no`]

These commands have been superseded by the `enable` and `disable` commands. They are listed here for historical purposes.

`logconfig` *configkeyword*

Controls the amount of output written to syslog or the logfile. By default all output is turned on. *configkeyword* is formed by concatenating the message class with the event class. It is permissible to use the prefix, `all`, instead of a message class. A message class may also be

followed by the keyword, `all`, meaning to enable/disable all of the respective message class. All *configkeywords* can be prefixed with the symbols, '=', '+' and '–' . Here, '=' sets the syslogmask, '+' adds messages, and '–' removes messages. Syslog messages can be controlled in four classes: *sys*, *peer*, *clock*, `sync`. Within these classes four types of messages can be controlled. Each is described below, along with its `configkeyword`:

`Configkeyword`	`Message type`
`info`	Informational messages control configuration information.
`events`	Event messages control logging of events (reachability, synchronization, alarm conditions).
`statistics`	Statistical messages control statistical output.
`status`	Status messages describe mainly the synchronization status.

A minimal log configuration might look like this:

```
logconfig =syncstatus +sysevents
```

A configuration like this lists, just the synchronization state of `xntp` and the major system events. For a simple reference server, the following minimum message configuration could be useful:

```
logconfig =syncall +clockall
```

This configuration lists all clock information and synchronization information. All other events and messages about peers, system events and so on, is suppressed.

Authentication Key File Format

The NTP standard specifies an extension to allow verification of the authenticity of received NTP packets, and to provide an indication of authenticity in outgoing packets. This is implemented in `xntpd` using the DES or MD5 algorithms to compute a digital signature, or message-digest. The specification allows any one of possibly 4 billion keys, numbered with 32 bit key identifiers, to be used to authenticate an association. The servers involved in an association must agree on the key and key identifier used to authenticate their data. However they must each learn the key and key identifier independently. In the case of DES, the keys are 56 bits long with, depending on type, a parity check on each byte. In the case of MD5, the keys are 64 bits (8 bytes). `xntpd` reads its keys from a file specified using the `-k` command line option or the `keys` statement in the configuration file. While key number 0 is fixed by the NTP standard (as 56 zero bits) and may not be changed, one or more of the keys numbered 1 through 15 may be arbitrarily set in the keys file.

The key file uses the same comment conventions as the configuration file. Key entries use a fixed format of the form, *keyno type key*. Here, *keyno* is a positive integer, `type` is a single character which defines the format the key is given in, and *key* is the key itself.

The *key* may be given in one of several different formats, controlled by the `type` character. The different key types, and corresponding formats, are described below:

`Key:` S

`Format:` A 64 bit hexadecimal number in DES format

In this format, the high order 7 bits of each octet are used to form the 56 bit key while the low order bit of each octet is given a value such that odd parity is maintained for the octet. Leading zeroes must be specified (i.e., the key must be exactly 16 hex digits long) and odd parity must be maintained. Hence a zero key, in standard format, would be given as: 0101010101010101.

`Key:` N

`Format:` A 64 bit hexadecimal number in NTP format

This format is the same as the DES format except the bits in each octet have been rotated one bit right so that the parity bit is now the high order bit of the octet. Leading zeroes must be specified and odd parity must be maintained. A zero key in NTP format would be specified as: `8080808080808080`.

`Key:` A

`Format:` A 1–to–8 character ASCII string

A key is formed from this by using the lower order 7 bits of the ASCII representation of each character in the string. Zeroes are added on the right when necessary to form a full width 56 bit key.

`Key:` M

`Format:` A 1–to–8 character ASCII string, using the MD5 authentication scheme.

Note that both the keys and the authentication schemes (DES or MD5) must be identical between a set of peers sharing the same key number.

Primary Clock Support

`xntpd` has been built to be compatible with all supported types of reference clocks. A reference clock is generally (though not always) a radio timecode receiver which is synchronized to a source of standard time such as the services offered by the NRC in Canada and NIST in the U.S. The interface between the computer and the timecode receiver is device dependent and will vary, but it is often a serial port.

For the purposes of configuration, `xntpd` treats reference clocks in a manner analogous to normal NTP peers as much as possible. Reference clocks are referred to by address, much as a normal peer is. However, an invalid IP address is used to distinguish them from normal peers. Reference clock addresses are of the form *127.127.t.u* where *t* is an integer denoting the clock type and *u* indicates the type–specific unit number. Reference clocks are configured using a `server` statement in the configuration file where the *host_address* is the clock address. The `key`, `version` and `ttl` options are not used for reference clock support. Some reference clocks require a `mode` option to further specify their operation. The `prefer` option can be useful to persuade the server to cherish a reference clock with somewhat more enthusiasm than other reference clocks or peers. Clock addresses may generally be used anywhere in the configuration file that a normal IP address can be used. For example, they can be used in `restrict` statements, although such use would normally be considered strange.

Reference clock support provides the `fudge` command, which can be used to configure reference clocks in special ways. The generic format that applies to this command is,

```
fudge 127.127.t.u [  time1 secs ] [  time2 secs ]\
     [  stratum int ] [  refid int ] \
     [  flag1 0 | 1 ] [  flag2 0 | 1 ] [  flag3 0 | 1 ] [  flag4  0 | 1 ]
```

with options described as follows:

time1
time2 Are specified in fixed point seconds and used in some clock drivers as calibration constants. By convention, and unless indicated otherwise, `time1` is used as a calibration constant to adjust the nominal time offset of a particular clock to agree with an external standard, such as a precision PPS signal. The specified offset is in addition to the propagation delay provided by other means, such as internal DIP switches.

stratum Is a number in the range zero to 15 and is used to assign a nonstandard operating stratum to the clock.

refid Is an ASCII string in the range one to four characters and is used to assign a nonstandard reference identifier to the clock.

flag1
flag2
flag3
flag4

Are binary flags used for customizing the clock driver. The interpretation of these values, and whether they are used at all, is a function of the needs of the particular clock driver. However, by convention, and unless indicated otherwise, *flag3* invokes the TIOCSPPS ioctl, while *flag4* is used to enable recording verbose monitoring data to the clockstats file configured with the `filegen` command.

Ordinarily, the stratum of a reference clock is zero, by default. Since the `xntpd` daemon adds one to the stratum of each peer, a primary server ordinarily displays stratum one. In order to provide engineered backups, it is often useful to specify the reference clock stratum as greater than zero. The `stratum` option is used for this purpose. Also, in cases involving both a reference clock and a 1-pps discipline signal, it is useful to specify the reference clock identifier as other than the default, depending on the driver. The *refid* option is used for this purpose. Except where noted, these options apply to all clock drivers.

`xntpd` on Unix machines currently supports several different types of clock hardware. It also supports a special pseudo–clock used for backup or when no other clock source is available. In the case of most of the clock drivers, support for a 1-pps precision timing signal is available as described in the `README` file in the `./doc directory of the xntp3` program distribution. The clock drivers, and the addresses used to configure them, are described in the file, `README.refclocks`, in the doc directory of the current program distribution.

Variables

Most variables used by the NTP protocol can be examined with `ntpq` (mode 6 messages). Currently very few variables can be modified via mode 6 messages. These variables are either created with the `setvar` directive or the leap warning variables. The leap warning bits that can be set in the *leapwarning* variable (up to one month ahead). Both, the *leapwarning* and in the *leapindication* variable, have a slightly different encoding than the usual `leap` bits interpretation:

`00` The daemon passes the leap bits of its synchronization source (usual mode of operation).

`01/10` A leap second is added/deleted (operator forced leap second).

`11` Leap information from the synchronization source is ignored (thus LEAP_NOWARNING is passed on).

Files

`/etc/inet/ntp.conf`	Default name of the configuration file
`/etc/ntp/ntp.drift`	Conventional name of the drift file
`/etc/inet/ntp.keys`	Conventional name of the key file
`/etc/inet/ntp.server`	Sample server configuration file

Attributes See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWntpu

See Also svcs(1), ntpdate(1M), ntpq(1M), ntptrace(1M), svcadm(1M), xntpdc(1M), rename(2), attributes(5), smf(5)

Notes The xntpd service is managed by the service management facility, smf(5), under the service identifier:

```
svc:/network/ntp:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using svcadm(1M). The service's status can be queried using the svcs(1) command.

Name xntpdc – special NTP query program

Synopsis `xntpdc [-ilnps] [-c` *command*`] [`*host*`] [...]`

Description `xntpdc` queries the `xntpd` daemon about its current state and requests changes in that state. You can run `xntpdc` in interactive mode or in controlled using command line arguments.

Extensive state and statistics information is available through the `xntpdc` interface. In addition, nearly all the configuration options which can be specified at start up using `xntpd`'s configuration file may also be specified at run time using xntpdc.

If one or more request options is included on the command line when `xntpdc` is executed, each of the requests is sent to the NTP servers running on each of the hosts given as command line arguments, or on the local host by default. If no request options are given, `xntpdc` attempts to read commands from the standard input and execute these on the NTP server running on the first host specified on the command line, again defaulting to the local host when no other host is specified. `xntpdc` prompts for commands if the standard input is a terminal device.

`xntpdc` uses NTP `mode 7` packets to communicate with the NTP server, and can be used to query any compatable server on the network which permits it. As NTP is a UDP protocol, this communication is somewhat unreliable, especially over large distances. `xntpdc` does not attempt to re-transmit requests, and times requests out if the remote host is not heard from within a suitable timeout time.

The operation of `xntpdc` is specific to the particular implementation of the `xntpd` daemon. You can expect `xntpdc` to work only with this and maybe some previous versions of the daemon. Requests from a remote `xntpdc` program that affect the state of the local server must be authenticated. This requires that both the remote program and local server share a common key and key identifier.

Options `xntpdc` reads interactive format commands from the standard input. If you specify the `-c`, `-l`, `-p` or `-s` option, the specified queries are sent to the hosts immediately.

The following command line options are supported:

`-c` *command*`-...` Add *command* to the list of commands to execute on the specified hosts. *command* is interpreted as an interactive format command.

Multiple `-c` options may be specified.

`-i` Force `xntpdc` to operate in interactive mode.

Prompts are written to the standard output. Commands are read from the standard input.

`-l` Obtain a list of peers which are known to the servers.

This option is equivalent to `-c listpeers`. See `listpeers` in `Control Message Commands`.

`-n`	Output all host addresses in dotted-quad numeric format rather than converting to the canonical host names.
`-p`	Print a list of the peers known to the server as well as a summary of their state. This option is equivalent to `-c peers`. See `peers` in `Control Message Commands`.
`-s`	Print a list of the peers known to the server as well as a summary of their state, but in a slightly different format than the `-p` option. This option is equivalent to `-c dmpeers`. See `dmpeers` in `Control Message Commands`.

Operands The following operands are supported:

Usage

Interactive Commands The interactive commands consist of a keyword (*command_keyword*) followed by zero to four arguments. You need to entry only enough characters of the *command_keyword* to uniquely identify it. The output of an interactive command is sent to the standard output by default. You can send the output of an interactive command to a file by appending a <, followed by a file name, to the command line.

A number of interactive format commands are executed entirely within the `xntpdc` program itself and do not result in NTP mode.

The following interactive commands are supported:

`?` [*command_keyword*]	Without an argument, print a list of `ntpq` command keywords. If *command_keyword* is specified, print function and usage information about the *command_keyword*.
delay *milliseconds*	Specify a time interval to add to timestamps included in requests which require authentication. This enables (unreliable) server reconfiguration over long delay network paths or between machines whose clocks are unsynchronized. Because the server no longer requires timestamps in authenticated requests, this command may be obsolete.
help [*command_keyword*]	Without an argument, print a list of `ntpq` command keywords. If *command_keyword* is specified, print function and usage information about the *command_keyword*.
host *hostname*	Set the host (*hostname*) to which future queries are sent. Specify *hostname* as a host name or a numeric address.

hostnames [`yes` | `no`]	Print hostnames or numeric addresses in information displays. Specify `yes` to print host names. Specify `no` to print numeric addresses. The default is `yes`, unless the `-n` command line option is specified.
keyid *keyid*	Enable specification of a key number (*keyid*) to authenticate configuration requests. *keyid* must correspond to a key number the server has been configured to use for this purpose.
passwd	Allow the user to specify a password at the command line to authenticate configuration requests. The password is not displayed, and must correspond to the key configured for use by the NTP server for this purpose. If the password does not correspond to the key configured for use by the NTP server, requests are not successful.
quit	Exit `xntpdc`.
timeout *millseconds*	Specify a timeout period for responses to server queries. The default is approximately `8000` milliseconds. As `xntpdc` retries each query once after a timeout, the total waiting time for a timeout is twice the timeout value set.

Control Message Commands

Query commands result in `NTP` mode 7 packets containing requests for information being sent to the server. These control message commands are read-only commands in that they make no modification of the server configuration state.

The following control message commands are supported:

clkbug	Obtain debugging information for a reference clock driver. This information is provided only by some clock drivers.
clockinfo *clock_peer_address* [...]	Obtain and print information concerning a peer clock. The values obtained provide information on the setting of fudge factors and other clock performance information.
dmpeers	Obtain a list of peers for which the sserver is maintaining state, along with a summary of that state.

	The peer summary list is identical to the output of the `peers` command, except for the character in the leftmost column. Characters only appear beside peers which were included in the final stage of the clock selection algorithm. A `.` indicates that this peer was cast off in the falseticker detection, while a + indicates that the peer made it through. A * denotes the peer with which the server is currently synchronizing.
iostats	Print statistics counters maintained in the input-output module.
kerninfo	Obtain and print kernel phase-lock loop operating parameters. This information is available only if the kernel has been specially modified for a precision timekeeping function.
listpeers	Obtain and print a brief list of the peers for which the server is maintaining state. These should include all configured peer associations as well as those peers whose stratum is such that they are considered by the server to be possible future synchonization candidates. candidates.
loopinfo [oneline | multiline]	Print the values of selected loop filter variables. The loop filter is the part of NTP which deals with adjusting the local system clock. The `oneline` and `multiline` options specify the format in which this information is printed. `multiline` is the default. The offset is the last offset given to the loop filter by the packet processing code. The frequency is the frequency error of the local clock in parts-per-million (ppm). The `time_const` controls the stiffness of the phase-lock loop and thus the speed at which it can adapt to oscillator drift. The watchdog timer value is the number of seconds which have elapsed since the last sample offset was given to the loop filter.
memstats	Print statistics counters related to memory allocation code.

monlist [version]

Obtain and print traffic counts collected and maintained by the monitor facility. The version number should not normally need to be specified.

peers

Obtain a list of peers for which the server is maintaining state, along with a summary of that state.

The following summary information is included:

- Address of the remote peer.
- Local interface address. If a local address has yet to be determined it is `0.0.0.0`.
- Stratum of the remote peer. A stratum of `16` indicates the remote peer is unsynchronized.
- Polling interval, in seconds.
- Reachability register, in octal.
- Current estimated delay, offset and dispersion of the peer, in seconds.
- Mode in which the peer entry is operating.

  This is represented by the character in the left margin. A + denotes symmetric active, a - indicates symmetric passive, a = means the remote server is being polled in client mode, a ^ indicates that the server is broadcasting to this address, a ~ denotes that the remote peer is sending broadcasts and a * marks the peer the server is currently synchonizing to.
- Host.

  This field may contain a host name, an IP address, a reference clock implementation name with its parameter or `REFCLK` (implementation number, parameter). On hostnames no only IP-addresses is displayed.

pstats *peer_address* [...]

Show the per-peer statistic counters associated with the specified peers.

reslist

Obtain and print the server's restriction list.

Generally, this list is printed in sorted order.

showpeer *peer_address* [...]	Show a detailed display of the current peer variables for one or more peers. Most of these values are described in the NTP Version 2 specification.
sysinfo	Print a variety of system state variables that are related to the local server. The output from `sysinfo` is described in NTP Version 3 specification, RFC-1305. All except the last four lines are described in the NTP Version 3 specification, RFC-1305. The `system flags` show various system flags, some of which can be set and cleared by the `enable` and `disable` configuration commands, respectively. These are the `auth`, `bclient`, `monitor`, `pll`, `pps` and `stats` flags. See the xntpd documentation for the meaning of these flags. There are two additional flags which are read only, the `kernel_pll` and `kernel_pps`. These flags indicate the synchronization status when the precision time kernel modifications are in use. The `kernel_pll` indicates that the local clock is being disciplined by the kernel, while the `kernel_pps` indicates the kernel discipline is provided by the `PPS` signal. The stability is the residual frequency error remaining after the system frequency correction is applied and is intended for maintenance and debugging. In most architectures, this value initially decreases from as high as `500` ppm to a nominal value in the range `.01` to `0.1` ppm. If it remains high for some time after starting the daemon, something may be wrong with the local clock, or the value of the kernel variable tick may be incorrect. The `broadcastdelay` shows the default broadcast delay, as set by the `broadcastdelay` configuration command. The `authdelay` shows the default authentication delay, as set by the `authdelay` configuration command.
sysstats	Print statistics counters maintained in the protocol module.
timerstats	Print statistics counters maintained in the timer/event queue support code.

Runtime Configuration Requests

The server authenticates all requests that cause state changes in the server. The server uses a configured NTP key to accomplish this. This facility can also be disabled by the server by not configuring a key).

You must make the key number and the corresponding key known to `xtnpdc`. Use the `keyid` or `passwd` commands to do so.

The `passwd` command prompts users for a password to use as the encryption key. It also prompts automatically for both the key number and password the first time a command which would result in an authenticated request to the server is given. Authentication provides verification that the requester has permission to make such changes. It also gives an extra degree of protection against transmission errors.

Authenticated requests always include a time stamp in the packet data. The time stamp is included in the computation of the authentication code. This timestamp is compared by the server to its receive time stamp. If the time stamps differ by more than a small amount the request is rejected.

Time stamps are rejected for two reasons. First, it makes simple replay attacks on the server, by someone who might be able to overhear traffic on your LAN, much more difficult. Second, it makes it more difficult to request configuration changes to your server from topologically remote hosts.

While the reconfiguration facility works well with a server on the local host, and may work adequately between time-synchronized hosts on the same LAN, it works very poorly for more distant hosts. If reasonable passwords are chosen, care is taken in the distribution and protection of keys and appropriate source address restrictions are applied, the run time reconfiguration facility should provide an adequate level of security.

The following commands make authenticated requests.

addpeer *peer_address* [*keyid*] [*version*] [*prefer*]
: Add a configured peer association at the given address and operating in symmetric active mode. An existing association with the same peer may be deleted when this command is executed, or may simply be converted to conform to the new configuration, as appropriate.

  If the optional *keyid* is a non-zero integer, all outgoing packets to the remote server will have an authentication field attached encrypted with this key. If the *keyid* is `0` or omitted, no authentication is done.

  Specify *version* as `1`, `2` or `3`. The default is `3`.

  The *prefer* keyword indicates a preferred peer. This keyword is used primarily for clock synchronisation if possible. The preferred peer also determines the validity of the PPS signal - if the preferred peer is suitable for synchronisation so is the PPS signal.

addserver *peer_address* [*keyid*] [*version*] [*prefer*]
: Identical to the `addpeer` command, except that the operating mode is client.

addtrap [*address* [*port*] [*interface*]
: Set a trap for asynchronous messages.

authinfo
Return information concerning the authentication module, including known keys and counts of encryptions and decryptions which have been done.

broadcast *peer_address* [*keyid*] [*version*] [*prefer*]
Identical to the `addpeer` command, except that the operating mode is `broadcast`. In this case a valid key identifier and key are required. The `peer_address` parameter can be the broadcast address of the local network or a multicast group address assigned to NTP. If a multicast address, a multicast-capable kernel is required.

clrtrap [*address* [*port*] [*interface*]
Clear a trap for asynchronous messages.

delrestrict *address mask* [*ntpport*]
Delete the matching entry from the restrict list.

fudge *peer_address* [*time1*] [*time2*] [*stratum*] [*refid*]
Provide a way to set certain data for a reference clock.

readkeys
Cause the current set of authentication keys to be purged and a new set to be obtained by re-reading the `keys` file. The `keys` file must have been specified in the `xntpd` configuration file. This enables encryption keys to be changed without restarting the server.

restrict *address mask flag* [*flag*]
This command operates in the same way as the restrict configuration file commands of `xntpd`.

reset
Clear the statistics counters in various modules of the server.

traps
Display the traps set in the server.

trustkey *keyid* [...]
untrustkey *keyid* [...]
These commands operate in the same way as the `trustedkey` and `untrustkey` configuration file commands of `xntpd`.

unconfig *peer_address* [...]
Cause the configured bit to be removed from the specified peers. In many cases this causes the peer association to be deleted. When appropriate, however, the association may persist in an unconfigured mode if the remote peer is willing to continue on in this fashion.

unrestrict *address mask flag* [*flag*]
Unrestrict the matching entry from the restrict list.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWntpu

See Also ntpdate(1M), ntpq(1M), ntptrace(1M), xntpd(1M), rename(2), attributes(5)

Name ypbind – NIS binder process

Synopsis

```
/usr/lib/netsvc/yp/ypbind [-broadcast | -ypset | -ypsetme]
```

Description NIS provides a simple network lookup service consisting of databases and processes. The databases are stored at the machine that runs an NIS server process. The programmatic interface to NIS is described in `ypclnt(3NSL)`. Administrative tools are described in `ypinit(1M)`, `ypwhich(1)`, and `ypset(1M)`. Tools to see the contents of NIS maps are described in `ypcat(1)`, and `ypmatch(1)`.

`ypbind` is a daemon process that is activated at system startup time from the `svc:/network/nis/client:default` service. By default, it is invoked as `ypbind -broadcast`. `ypbind` runs on all client machines that are set up to use NIS. See `sysidtool(1M)`. The function of `ypbind` is to remember information that lets all NIS client processes on a node communicate with some NIS server process. `ypbind` must run on every machine which has NIS client processes. The NIS server may or may not be running on the same node, but must be running somewhere on the network. If the NIS server is a NIS+ server in NIS (YP) compatibility mode, see the `NOTES` section of the `ypfiles(4)`man page for more information.

The information `ypbind` remembers is called a *binding* — the association of a domain name with a NIS server. The process of binding is driven by client requests. As a request for an unbound domain comes in, if started with the `-broadcast` option, the `ypbind` process broadcasts on the net trying to find an NIS server, either a `ypserv` process serving the domain or an `rpc.nisd` process in "YP-compatibility mode" serving NIS+ directory with name the same as (case sensitive) the domain in the client request. Since the binding is established by broadcasting, there must be at least one NIS server on the net. If started without the `-broadcast` option, `ypbind` process steps through the list of NIS servers that was created by `ypinit -c` for the requested domain. There must be an NIS server process on at least one of the hosts in the NIS servers file. All the hosts in the NIS servers file must be listed in the `/etc/hosts` file along with their IP addresses. Once a domain is bound by `ypbind`, that same binding is given to every client process on the node. The `ypbind` process on the local node or a remote node may be queried for the binding of a particular domain by using the `ypwhich(1)` command.

If `ypbind` is unable to speak to the NIS server process it is bound to, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to bind the domain once again. Requests received for an unbound domain will wait until the requested domain is bound. In general, a bound domain is marked as unbound when the node running the NIS server crashes or gets overloaded. In such a case, `ypbind` will try to bind to another NIS server using the process described above.`ypbind` also accepts requests to set its binding for a particular domain. The request is usually generated by the `ypset(1M)` command. In order for `ypset` to work, `ypbind` must have been invoked with flags `-ypset` or `-ypsetme`.

Options

`-broadcast` Send a broadcast datagram using UDP/IP that requests the information needed to bind to a specific NIS server. This option is analogous to `ypbind` with no options in earlier Sun releases and is recommended for ease of use.

`-ypset` Allow users from any remote machine to change the binding by means of the `ypset` command. By default, no one can change the binding. This option is insecure.

`-ypsetme` Only allow `root` on the local machine to change the binding to a desired server by means of the `ypset` command. `ypbind` can verify the caller is indeed a `root` user by accepting such requests only on the loopback transport. By default, no external process can change the binding.

Files `/var/yp/binding/`*ypdomain*`/ypservers`

`/etc/inet/hosts`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `svcs(1)`, `ypcat(1)`, `ypmatch(1)`, `ypwhich(1)`, `ifconfig(1M)`, `rpc.nisd(1M)`, `svcadm(1M)`, `ypinit(1M)`, `ypset(1M)`, `ypclnt(3NSL)`, `hosts(4)`, `ypfiles(4)`, `attributes(5)`, `smf(5)`

Notes `ypbind` supports multiple domains. The `ypbind` process can maintain bindings to several domains and their servers, the default domain is the one specified by the `domainname(1M)` command at startup time.

The `-broadcast` option works only on the UDP transport. It is insecure since it trusts “any” machine on the net that responds to the broadcast request and poses itself as an NIS server.

The `ypbind` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nis/client:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name ypinit – set up NIS client

Synopsis

```
/usr/sbin/ypinit [-c] [-m] [-s master_server]
```

Description `ypinit` can be used to set up an NIS client system. You must be the superuser to run this command. This script need not be used at all if `ypbind`(1M) is started with the `-broadcast` option (it is invoked with this option from the `svc:/network/nis/client:default service`).

Normally, `ypinit` is run only once after installing the system. It may be run whenever a new NIS server is added to the network or an existing one is decommissioned.

`ypinit` prompts for a list of NIS servers to bind the client to; this list should be ordered from the closest to the furthest server. Each of these NIS servers must be listed in the `/etc/hosts` file along with its IP address. `ypinit` stores the list in file `/var/yp/binding/`*domain*`/ypservers`. This file is used by `ypbind` when run without the `-broadcast` option.

Options

`-c`	Set up a `ypclient` system.
`-m`	Build a master `ypserver` data base.
`-s` *master_server*	Slave data base. *master_server* must be the same master configured in the YP maps and returned by the `ypwhich -m` command.

Files

`/etc/hosts`

`/var/yp/binding/`*domain*`/ypservers`

Attributes See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `svcs`(1), `svcadm`(1M), `ypbind`(1M), `sysinfo`(2), `hosts`(4), `attributes`(5), `smf`(5)

Notes The `NIS client` service is managed by the service management facility, `smf`(5), under the service identifier:

```
svc:/network/nis/client:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm`(1M). The service's status can be queried using the `svcs`(1) command.

Bugs `ypinit` sets up the list of NIS servers only for the current domain on the system when it is run, that is, the domain returned by the `SI_SRPC_DOMAIN` command to `sysinfo`(2). Care should be taken to ensure that this is the same as the desired domain for NIS client processes.

Name ypmake – rebuild NIS database

Synopsis

```
cd /var/yp ; make [map]
```

Description The file called `Makefile` in `/var/yp` is used by `make(1S)` to build the Network Information Service (NIS) database. With no arguments, `make` creates `dbm` databases for any NIS maps that are out-of-date, and then executes `yppush(1M)` to notify slave databases that there has been a change.

If you supply a *map* on the command line, `make` will update that map only. Typing `make passwd` will create and `yppush` the password database (assuming it is out of date). Likewise, `make ipnodes` and `make networks` will create and `yppush` the `ipnodes` and `network` files, `$(INETDIR)/ipnodes` and `$(DIR)/networks`.

There are four special variables used by `make`: `DIR`, which gives the directory of the source files; `NOPUSH`, which when non-null inhibits doing a `yppush` of the new database files; `INETDIR`, which gives the directory of the `ipnodes` source file; and `DOM`, which is used to construct a domain other than the master's default domain. The default for `DIR` is `/etc`, and the default for `INETDIR` is `/etc/inet`. The default for `NOPUSH` is the null string.

Refer to `ypfiles(4)` and `ypserv(1M)` for an overview of the NIS service.

If a NIS to LDAP (N2L) configuration file, `/var/yp/NISLDAPmapping`, is present, the NIS server components run in N2L mode. In N2L mode, the server components use a new set of map files with an LDAP-prefix, based on the LDAP DIT. In N2L mode, authoritative NIS information is obtained from the DIT. The NIS source files and `ypmake` have no role, and they should not be used. If `ypmake` is accidentally run, then the server components will detect this, and will log a warning message. For additional information, see `ypfiles(4)`.

Files

`/var/yp`	Directory containing NIS configuration files.
`/etc/inet/hosts`	System hosts file.
`/etc`	Default directory for source files.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu
Interface Stability	Evolving

See Also `make(1S)`, `NIS+(1)`, `makedbm(1M)`, `rpc.nisd(1M)`, `ypbind(1M)`, `yppush(1M)`, `ypserv(1M)`, `ypclnt(3NSL)`, `NISLDAPmapping(4)`, `ypfiles(4)`, `ypserv(4)`

Notes The NIS makefile is only used when running the ypserv(1M) server to provide NIS services. If these are being provided by the NIS+ server running in NIS compatibility mode, see rpc.nisd(1M); this makefile is not relevant. See ypfiles(4) for more details.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same. Only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

Name ypmap2src – convert NIS maps to NIS source files

Synopsis

```
/usr/lib/netsvc/yp/ypmap2src [-t]
     [ [-c custom-map-name]]... [-d domain] -o output-directory
     [ [ source-file]]...
```

Description Use the `ypmap2src` utility to convert standard NIS maps to approximations of the equivalent NIS source files. This utility functions like the reverse of `ypmake(1M)`.

The primary use for `ypmap2src` is to convert from a NIS server that uses the NIS to LDAP(N2L) transition mechanism, which does not use NIS source files, to traditional NIS, where source files are required. The `ypmap2src` utility is also used by NIS administrators who wish to discover the contents of NIS maps for which the sources are not available.

Generally, this operation is not necessary. More often, administrators will switch from traditional NIS to N2L in anticipation of the eventual transition to LDAP naming. When this switch is made, authoritative information is moved into the LDAP DIT, and the NIS sources have no further role. N2L supports NIS clients until such time as they can be converted to LDAP, and the NIS service suspended.

The `ypmap2src` utility does not guarantee that the files that are generated are identical to the original NIS source files. Some information might have been thrown away by `ypmake` and cannot be recovered. N2L also might have updated the maps to reflect changes made by LDAP clients. It is essential that the sources generated are checked to confirm no problems have occurred.

Per entry comment fields, from existing source files, are not merged into source files generated by `ypmap2src`. If a user wishes N2L to maintain comment information, then the `NISLDAPmapping` configuration file should be modified so that the comment fields are mapped into LDAP. This will ensure that the comments are visible to native LDAP clients and present in the N2L map files.

When `ypmap2src` is run, it will take up-to-date comments from the map file and insert them into the NIS source file generated.

Handling Custom Maps `ypmap2src` only knows about the standard NIS maps and standard source to map conversion. If an advanced user has changed these, that is, the user has modified the NIS makefile, the equivalent changes must also be made to the `ypmap2src` script.

Options `ypmap2src` supports the following options:

`-c` Specifies that *custom-map-name* should be converted to a source file by running `makedbm -u` on it. This is a short cut so that simple custom maps can be handled without editing `ypmap2src`.

`-d` *domain-name* Specifies the domain to convert. The *domain-name* can be a fully qualified file path, such as `/var/yp/a.b.c`, or just a domain name, a.b.c. In the latter case, `ypmaptosrc` looks in `/var/yp` for the domain directory.

`-o` *dest* — Specifies the destination directory for the converted files. A directory other than `/etc` should be specified. The maps generated are copied to the correct location, `/etc`, `/etc/security` or other source directory, as appropriate.

`-t` — Specifies that traditional NIS maps, without N2L's `LDAP_` prefix, should be converted. By default, maps with the `LDAP_` prefix are converted.

Operands `ypmap2src` supports the following operands:

source-file — Lists the standard source files to convert. If this option is not given, then all the standard source files, plus any custom files pecified by the `-c` option, are converted.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu
Interface Stability	Obsolete

See Also `ypmake(1M)`, `ypserv(1M)`, `NISLDAPmapping(4)`, `attributes(5)`

System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)

Name yppoll – return current version of a NIS map at a NIS server host

Synopsis

```
/usr/sbin/yppoll [-d ypdomain] [-h host] mapname
```

Description The `yppoll` command asks a `ypserv( )` process what the order number is, and which host is the master NIS server for the named map.

Options

`-d` *ypdomain* Use *ypdomain* instead of the default domain.

`-h` *host* Ask the `ypserv` process at *host* about the map parameters. If *host* is not specified, the NIS server for the local host is used. That is, the default host is the one returned by `ypwhich(1)`.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also `ypwhich(1)`, `ypfiles(4)`, `attributes(5)`

Name yppush – force propagation of changed NIS map

Synopsis

```
/usr/lib/netsvc/yp/yppush [-v] [-h host] [-d domain]
     [-p #parallel-xfrs] mapname
```

Description `yppush` copies a new version of a Network Information Service (NIS) map from the master NIS server to the slave NIS servers. It is normally run only on the master NIS server by the `Makefile` in `/var/yp` after the master databases are changed. It first constructs a list of NIS server hosts by reading the NIS `ypservers` map within the *domain*. Keys within the `ypservers` map are the ASCII names of the machines on which the NIS servers run.

A "transfer map" request is sent to the NIS server at each host, along with the information needed by the transfer agent (the program that actually moves the map) to call back the `yppush`. When the attempt has completed (successfully or not), and the transfer agent has sent `yppush` a status message, the results can be printed to `stdout`. Messages are also printed when a transfer is not possible, for instance, when the request message is undeliverable, or when the timeout period on responses has expired.

Refer to `ypfiles(4)` and `ypserv(1M)` for an overview of the NIS service.

Options The following options are supported:

`-d` *domain*	Specifies a *domain*.
`-h` *host*	Propagates only to the named *host*.
`-p` *#parallel-xfrs*	Allows the specified number of map transfers to occur in parallel.
`-v`	Verbose. This prints messages when each server is called, and for each response. If this flag is omitted, only error messages are printed.

Files

`/var/yp`	Directory where NIS configuration files reside.
`/var/yp/`*domain*`/ypservers.` `{`*dir*`,` *pag* `}`	Map containing list of NIS servers to bind to when running in server mode.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu

See Also `ypserv(1M)`, `ypxfr(1M)`, `ypfiles(4)`, `attributes(5)`

Notes The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications PLC, and must not be used without permission.

Bugs In the current implementation (version 2 NIS protocol), the transfer agent is `ypxfr(1M)`, which is started by the `ypserv` program. If `yppush` detects that it is speaking to a version 1 NIS protocol server, it uses the older protocol, sending a version 1 `YPPROC_GET` request and issues a message to that effect. Unfortunately, there is no way of knowing if or when the map transfer is performed for version 1 servers. `yppush` prints a message saying that an "old-style" message has been sent. The system administrator should later check to see that the transfer has actually taken place.

Name ypserv, ypxfrd – NIS server and binder processes

Synopsis

```
/usr/lib/netsvc/yp/ypserv [-dv] [-i | -I] [-r | -R]

/usr/lib/netsvc/yp/ypxfrd
```

Description The Network Information Service (NIS) provides a simple network lookup service consisting of databases and processes. The databases are `ndbm` files in a directory tree rooted at `/var/yp`. See `ndbm(3C)`. These files are described in `ypfiles(4)`. The processes are `/usr/lib/netsvc/yp/ypserv`, the NIS database lookup server, and `/usr/lib/netsvc/yp/ypbind`, the NIS binder. The programmatic interface to the NIS service is described in `ypclnt(3NSL)`. Administrative tools are described in `yppoll(1M)`, `yppush(1M)`, `ypset(1M)`, `ypxfr(1M)`, and `ypwhich(1)`. Tools to see the contents of NIS maps are described in `ypcat(1)`, and `ypmatch(1)`. Database generation and maintenance tools are described in `ypinit(1M)`, `ypmake(1M)`, and `makedbm(1M)`.

The `ypserv` utility is a daemon process typically activated at system startup from `svc:/network/nis/server:default`. Alternatively, you can, as the root user, start NIS services using `ypstart(1M)` from the command-line. `ypserv` runs only on NIS server machines with a complete NIS database. You can halt all NIS services using the `ypstop(1M)` command.

The `ypxfrd` utility transfers entire NIS maps in an efficient manner. For systems that use this daemon, map transfers are 10 to 100 times faster, depending on the map. To use this daemon, be sure `ypxfrd` is running on the master server. See `/usr/lib/netsvc/yp/ypstart`. `ypxfr` attempts to use `ypxfrd` first. If that fails, it prints a warning, then uses the older transfer method.

The `ypserv` daemon's primary function is to look up information in its local database of NIS maps.

The operations performed by `ypserv` are defined for the implementor by the *YP Protocol Specification*, and for the programmer by the header file <`rpcsvc/yp_prot.h`>.

Communication to and from `ypserv` is by means of RPC calls. Lookup functions are described in `ypclnt(3NSL)`, and are supplied as C-callable functions in the `libnsl(3LIB)` library. There are four lookup functions, all of which are performed on a specified map within some NIS domain: `yp_match(3NSL)`, `yp_first(3NSL)`, `yp_next(3NSL)`, and `yp_all(3NSL)`. The `yp_match` operation takes a key, and returns the associated value. The `yp_first` operation returns the first key-value pair from the map, and `yp_next` can be used to enumerate the remainder. `yp_all` ships the entire map to the requester as the response to a single RPC request.

A number of special keys in the DBM files can alter the way in which `ypserv` operates. The keys of interest are:

`YP_INTERDOMAIN` The presence of this key causes `ypserv` to forward to a DNS server host lookups that cannot be satisfied by the DBM files.

`YP_SECURE`	This key causes `ypserv` to answer only questions coming from clients on reserved ports.
`YP_MULTI_`*hostname*	This is a special key in the form, `YP_MULTI_`*hostname addr1,...,addrN*. A client looking for *hostname* receives the closest address.

Two other functions supply information about the map, rather than map entries: `yp_order(3NSL)`, and `yp_master(3NSL)`. In fact, both order number and master name exist in the map as key-value pairs, but the server will not return either through the normal lookup functions. If you examine the map with `makedbm(1M)`, however, they are visible. Other functions are used within the NIS service subsystem itself, and are not of general interest to NIS clients. These functions include `do_you_serve_this_domain?`, `transfer_map`, and `reinitialize_internal_state`.

On start up, `ypserv` checks for the existence of the NIS to LDAP (N2L) configuration file `/var/yp/NISLDAPmapping`. If it is present then a master server starts in N2L mode. If the file is not present it starts in "traditional" (non N2L) mode. Slave servers always start in traditional mode.

In N2L mode, a new set of map files, with an `LDAP_` prefix, are generated, based on the contents of the LDAP DIT. The old map files, NIS source files and `ypmake(1M)` are not used.

It is possible that `ypmake(1M)` can be accidentally run in N2L mode. If the occurs, the old style map files are overwritten. That the map files are overwritten is harmless. However, any resulting `yppush(1M)` operation will push information based on the DIT rather than the source files. The user may not expect information based on the DIT. `ypserv` keeps track of the last modification date of the old style map files. If the map files have been updated, a warning is logged that suggests that the user call `yppush` directly instead of `ypmake`.

If a server attempts to run in N2L mode and a LDAP server cannot be contacted, it behaves as follows:

1. When `ypserv` is started, a warning will be logged.
2. When a NIS read access is made and the TTL entry has expired, a warning is logged.Information that is returned from the cache has not been updated.
3. When a NIS write access is made, a warning is logged. The cache will not be updated, and a NIS failure will be returned.

If `ypxfrd` is running in N2L mode and is asked to transfer a map, `ypxfrd` first checks whether the map is out of date. If the map is out of date, `ypxfrd` initiates an update from the DIT. `ypxfrd` cannot wait for the update to complete. If `ypxfrd` waited, the client end `ypxfr` operation could time out. To prevent `ypxfrd` from timing out, the existing map is transferred from the cache. The most up to date map will be transferred on subsequent `ypxfrd` operations.

Options

ypserv `-d` The NIS service should go to the DNS for more host information. This requires the existence of a correct `/etc/resolv.conf` file pointing to a DNS server. This option turns on DNS forwarding regardless of whether or not the `YP_INTERDOMAIN` flag is set in the `hosts` maps. See makedbm(1M). In the absence of an `/etc/resolv.conf` file, `ypserv` complains, but ignores the `-d` option.

`-i` If in N2L mode, initialize the NIS related parts of the `DIT` based on the current, non `LDAP_` prefixed, map files. The `LDAP_` prefixed maps are not created or updated. If you require that `LDAP_` prefixed maps be updated or created, then use the `-ir` option.

The `-i` option does not attempt to create any NIS domain or container objects. If any NIS domain or container objects have not already been created, then errors will occur, as entries are written to nonexistent containers.

`-I` Identical to `-i`, except that any missing domain and container objects are created.

`-r` If in N2L mode, then refresh the `LDAP_` prefixed map files based on the contents of the `DIT`.

`-ir` If both `-i` and `-r` are specified in N2L mode, then the `DIT` will first be initialized from the current non `LDAP_` prefixed map files. A new set of `LDAP_` prefixed maps will then be generated from the contents of the `DIT`. A new set of `LDAP_` prefixed maps is required when moving from traditional NIS to N2L mode NIS.

`-Ir` Identical to `-ir`, except that any missing domain and container objects are created.

`-v` Operate in the verbose mode, printing diagnostic messages to stderr.

When run with the `-i`, `-r`, `-I`, `-ir` or `-Ir` options, the `ypserv` command runs in the foreground and exits once map initialization has been completed. Once the `ypserv` command exits, the user knows the maps are ready and can restart `ypserv` and the other `yp` daemons by running ypstart(1M).

If there is a requirement to initialize the `DIT` from the NIS source files, which may have been modified since the maps were last remade, run `ypmake` before running `ypserv -i` or `ypserv -ir`. `ypmake` regenerated old style NIS maps. Then `ypserv -ir` dumps them into the `DIT`. When the `-ir` option is used, the `LDAP_` prefixe maps are also generated or updated. Since these maps will be more recent than the old style maps, `ypmake` will not be reported as erroneous when it is run.

Files

`/var/yp/securenets` Defines the hosts and networks that are granted access to information in the served domain. It is read at startup time by both `ypserv` and `ypxfrd`.

`/var/yp/ypserv.log`	If the `/var/yp/ypserv.log` file exists when `ypserv` starts up, log information is written to it when error conditions arise.
`/var/yp/binding/domainname/ypservers`	Lists the NIS server hosts that `ypbind` can bind to.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu

See Also `svcs(1)`, `ypcat(1)`, `ypmatch(1)`, `ypwhich(1)`, `domainname(1M)`, `makedbm(1M)`, `svcadm(1M)`, `ypbind(1M)`, `ypinit(1M)`, `ypmake(1M)`, `yppoll(1M)`, `yppush(1M)`, `ypset(1M)`, `ypstart(1M)`, `ypstop(1M)`, `ypxfr(1M)`, `ndbm(3C)`, `ypclnt(3NSL)`, `libnsl(3LIB)`, `NISLDAPmapping(4)`, `securenets(4)`, `ypfiles(4)`, `ypserv(4)`, `attributes(5)`, `smf(5)`

Network Interface Guide

System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)

Notes `ypserv` supports multiple domains. The `ypserv` process determines the domains it serves by looking for directories of the same name in the directory `/var/yp`. It replies to all broadcasts requesting yp service for that domain.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications PLC, and must not be used without permission.

NIS uses `ndbm()` files to store maps. Therefore, it is subject to the 1024 byte limitations described in the USAGE and NOTES sections of the `ndbm(3C)` man page.

The NIS server service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/network/nis/server:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name ypset – point ypbind at a particular server

Synopsis

```
/usr/sbin/ypset [-d ypdomain] [-h host] server
```

Description In order to run `ypset`, `ypbind` must be initiated with the –`ypset` or –`ypsetme` options. See `ypbind(1M)`. `ypset` tells `ypbind` to get NIS services for the specified *ypdomain* from the `ypserv` process running on *server*. If *server* is down, or is not running `ypserv`, this might not be discovered until an NIS client process tries to obtain a binding for the domain. At this point, the binding set by `ypset` is tested by `ypbind`. If the binding is invalid, `ypbind` attempts to rebind for the same domain.

`ypset` is useful for binding a client node that is not on a broadcast net, or is on a broadcast net that is not running an NIS server host. It is also useful for debugging NIS client applications, for instance, where an NIS map exists only at a single NIS server host.

Where several hosts on the local net are supplying NIS services, `ypbind` can rebind to another host, even while you attempt to find out if the `ypset` operation succeeded. For example, if you enter the `ypset` command below, you might get the subsequent response from `ypwhich`:

```
example% ypset host1
example% ypwhich
host2
```

The sequence shown above is a function of the NIS subsystem's attempt to load-balance among the available NIS servers, and occurs when `host1` does not respond to `ypbind` because it is not running `ypserv` (or is overloaded), and `host2`, running `ypserv`, obtains the binding.

server indicates which NIS server to bind to, and must be specified as a name or an IP address. This works only if the node has a current valid binding for the domain in question and `ypbind` has been set to allow use of `ypset`. In most cases, *server* should be specified as an IP address.

`ypset` tries to bind over a connectionless transport. The NIS library call, `yp_all()`, uses connection-oriented transport and derives the NIS server's address based on the connectionless address supplied by `ypset`.

Refer to `ypfiles(4)` for an overview of the NIS name service.

Options

`-d` *ypdomain* Use *ypdomain*, instead of the default domain.

`-h` *host* Set `ypbind`'s binding on *host*, instead of locally. Specify *host* as a name.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

See Also ypwhich(1), ypfiles(4), attributes(5)

Name ypstart, ypstop – Start and stop NIS services

Synopsis

```
/usr/lib/netsvc/yp/ypstart

/usr/lib/netsvc/yp/ypstop
```

Description The `ypstart` command is used to start the Network Information Service (NIS). After the host has been configured using the `ypinit(1M)` command, `ypstart` automatically determines the NIS status of the machine and starts the appropriate daemons.

The `ypstop` command is used to stop the Network Information Service (NIS).

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu

See Also `ypinit(1M)`, `attributes(5)`

System Administration Guide: Basic Administration

Notes The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two services remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications PLC, and must not be used without permission.

Name ypxfr, ypxfr_1perday, ypxfr_1perhour, ypxfr_2perday – transfer NIS map from a NIS server to host

Synopsis

```
/usr/lib/netsvc/yp/ypxfr [-c] [-f] [-C tid prog server]
     [-d ypdomain] [-h host] [-s ypdomain] mapname
```

Description The `ypxfr` command moves an NIS map in the default domain for the local host to the local host by making use of normal NIS services. It creates a temporary map in the directory `/var/yp/`*ypdomain* (this directory must already exist; *ypdomain* is the default domain for the local host), fills it by enumerating the map's entries, fetches the map parameters (master and order number), and loads them. It then deletes any old versions of the map and moves the temporary map to the real *name*.

If run interactively, `ypxfr` writes its output to the terminal. However, if it is started without a controlling terminal, and if the log file `/var/yp/ypxfr.log` exists, it appends all its output to that file. Since `ypxfr` is most often run from the privileged user's `crontab` file, or by `ypserv`, the log file can retain a record of what was attempted, and what the results were.

For consistency between servers, `ypxfr` should be run periodically for every map in the NIS data base. Different maps change at different rates: a map might not change for months at a time, for instance, and can therefore be checked only once a day. Some maps might change several times per day. In such a case, you might want to check hourly for updates. A `crontab(1)` entry can be used to automatically perform periodic updates. Rather than having a separate `crontab` entry for each map, you can group commands to update several maps in a shell script. Examples (mnemonically named) are in `/usr/sbin/yp`: `ypxfr_1perday`, `ypxfr_2perday`, and `ypxfr_1perhour`.

Refer to `ypfiles(4)` for an overview of the NIS name service.

Options

`-c`	Do not send a “Clear current map” request to the local `ypserv` process. Use this flag if `ypserv` is not running locally at the time you are running `ypxfr`. Otherwise, `ypxfr` complains that it cannot communicate with the local `ypserv`, and the transfer fails.
`-f`	Force the transfer to occur even if the version at the master is not more recent than the local version.
`-C` *tid prog server*	This option is for use *only* by `ypserv`. When `ypserv` starts `ypxfr`, it specifies that `ypxfr` should call back a `yppush` process at the host *server*, registered as program number *prog*, and waiting for a response to transaction *tid*.
`-d` *ypdomain*	Specify a domain other than the default domain.
`-h` *host*	Get the map from *host*, regardless of the master. If *host* is not specified, `ypxfr` asks the NIS service for the name of the master, and tries to get the map from there. *host* must be a valid host name.

`-s` *ypdomain*	Specify a source domain from which to transfer a map that should be the same across domains.

Files

`/var/yp/ypxfr.log`	Log file
`/usr/lib/netsvc/yp/ypxfr_1perday`	Script to run one transfer per day, for use with `cron`(1M)
`/usr/lib/netsvc/yp/ypxfr_2perday`	Script to run two transfer per day, for use with `cron`(1M)
`/usr/lib/netsvc/yp/ypxfr_1perhour`	Script for hourly transfers of volatile maps
`/var/yp/`*ypdomain*	NIS domain
`/usr/spool/cron/crontabs/root`	Privileged user's `crontab` file

Attributes See `attributes`(5) for descriptions of the following attributes:

ypxfr Only

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

ypxfr_1perday, ypxfr_1perhour, and ypxfr_2perday

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWypu

See Also `crontab`(1), `cron`(1M), `ypinit`(1M), `yppush`(1M), `ypserv`(1M), `ypfiles`(4), `attributes`(5)

Name zdb – ZFS debugger

Synopsis `zdb` *pool*

Description The `zdb` command is used by support engineers to diagnose failures and gather statistics. Since the ZFS file system is always consistent on disk and is self-repairing, `zdb` should only be run under the direction by a support engineer.

If no arguments are specified, `zdb`, performs basic consistency checks on the pool and associated datasets, and report any problems detected.

Any options supported by this command are internal to Sun and subject to change at any time.

Exit Status The following exit values are returned:

`0` The pool is consistent.

`1` An error was detected.

`2` Invalid command line options were specified.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWzfsu
Interface Stability	Unstable

See Also `zfs(1M)`, `zpool(1M)`, `attributes(5)`

Name zdump – time zone dumper

Synopsis

```
zdump [--version] [-v] [-c [loyear,]hiyear] [zonename]...
```

Description The `zdump` command prints the current time for each time zone (*zonename*) listed on the command line. Specify *zonename* as the name of the time zone database file relative to `/usr/share/lib/zoneinfo`.

Specifying an invalid time zone (*zonename*) to `zdump` does not return an error, rather `zdump` uses GMTO. This is consistent with the behavior of the library calls; `zdump` reflects the same behavior of the time routines in `libc`. See `ctime(3C)` and `mktime(3C)`.

Options The following options are supported:

`--`*version* Outputs version information and exits.

`-v` Displays the entire contents of the time zone database file for *zonename*. Prints the time at the lowest possible time value; the time one day after the lowest possible time value; the times both one second before and exactly at each time at which the rules for computing local time change; the time at the highest possible time value; and the time at one day less than the highest possible time value. See `mktime(3C)` and `ctime(3C)` for information regarding time value (`time_t`). Each line of output ends with `isdst=1` if the given time is Daylight Saving Time, or `isdst=0` otherwise.

`-c` [*loyear*],*hiyear* Cuts off the verbose output near the start of the given year(s) . By default, the program cuts off verbose output near the start of the years -500 and 2500.

Exit Status The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

Files `/usr/share/lib/zoneinfo` Standard zone information directory

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Stable

See Also `zic(1M)`, `ctime(3C)`, `mktime(3C)`, `attributes(5)`, `environ(5)`

Name zfs – configures ZFS file systems

Synopsis

```
zfs [-?]

zfs create [-p] [-o property=value] ... filesystem

zfs create [-ps] [-b blocksize] [-o property=value] ... -V size volume

zfs destroy [-rRf] filesystem|volume

zfs destroy [-rRd] snapshot

zfs snapshot [-r] [-o property=value]...
      filesystem@snapname|volume@snapname

zfs rollback [-rRf] snapshot

zfs clone [-p] [-o property=value] ... snapshot filesystem|volume

zfs promote clone-filesystem

zfs rename filesystem|volume|snapshot
     filesystem|volume|snapshot

zfs rename [-p] filesystem|volume filesystem|volume

zfs rename -r snapshot snapshot

zfs list [-r|-d depth][-H][-o property[,...]] [-t type[,...]]
     [-s property] ... [-S property] ... [filesystem|volume|snapshot] ...

zfs set property=value filesystem|volume|snapshot ...

zfs get [-r|-d depth][-Hp][-o all | field[,...]] [-s source[,...]]
     all | property[,...] filesystem|volume|snapshot ...

zfs inherit [-rS] property filesystem|volume|snapshot ...

zfs upgrade

zfs upgrade -v

zfs upgrade [-r] [-V version] -a | filesystem

zfs userspace [-niHp] [-o field[,...]] [-sS field] ...
     [-t type [,...]] filesystem|snapshot

zfs groupspace [-niHp] [-o field[,...]] [-sS field] ...
     [-t type [,...]] filesystem|snapshot

zfs mount

zfs mount [-vO] [-o options] -a | filesystem

zfs unmount [-f] -a | filesystem|mountpoint

zfs share -a | filesystem

zfs unshare -a filesystem|mountpoint

zfs send [-DvRp] [-[iI] snapshot] snapshot
```

```
zfs receive [-vnFu] filesystem|volume|snapshot

zfs receive [-vnFu] [-d | -e] filesystem

zfs allow filesystem|volume

zfs allow [-ldug] everyone|user|group[,...] perm|@setname[,...]
     filesystem|volume

zfs allow [-ld] -e perm|@setname[,...] filesystem|volume

zfs allow -c perm|@setname[,...] filesystem|volume

zfs allow -s @setname perm|@setname[,...] filesystem|volume

zfs unallow [-rldug] everyone|user|group[,...] [perm|@setname[,... ]]
     filesystem|volume

zfs unallow [-rld] -e [perm|@setname[,... ]] filesystem|volume

zfs unallow [-r] -c [perm|@setname[ ... ]] filesystem|volume

zfs unallow [-r] -s @setname [perm|@setname[,... ]] filesystem|volume

zfs hold [-r] tag snapshot...

zfs holds [-r] snapshot...

zfs release [-r] tag snapshot...
```

Description The `zfs` command configures ZFS datasets within a ZFS storage pool, as described in `zpool(1M)`. A dataset is identified by a unique path within the ZFS namespace. For example:

```
pool/{filesystem,volume,snapshot}
```

where the maximum length of a dataset name is `MAXNAMELEN` (256 bytes).

A dataset can be one of the following:

file system
A ZFS dataset of type `filesystem` can be mounted within the standard system namespace and behaves like other file systems. While ZFS file systems are designed to be POSIX compliant, known issues exist that prevent compliance in some cases. Applications that depend on standards conformance might fail due to nonstandard behavior when checking file system free space.

volume
A logical volume exported as a raw or block device. This type of dataset should only be used under special circumstances. File systems are typically used in most environments.

snapshot
A read-only version of a file system or volume at a given point in time. It is specified as *filesystem@name* or *volume@name*.

ZFS File System Hierarchy

A ZFS storage pool is a logical collection of devices that provide space for datasets. A storage pool is also the root of the ZFS file system hierarchy.

The root of the pool can be accessed as a file system, such as mounting and unmounting, taking snapshots, and setting properties. The physical storage characteristics, however, are managed by the `zpool(1M)` command.

See `zpool(1M)` for more information on creating and administering pools.

Snapshots

A snapshot is a read-only copy of a file system or volume. Snapshots can be created extremely quickly, and initially consume no additional space within the pool. As data within the active dataset changes, the snapshot consumes more data than would otherwise be shared with the active dataset.

Snapshots can have arbitrary names. Snapshots of volumes can be cloned or rolled back, but cannot be accessed independently.

File system snapshots can be accessed under the `.zfs/snapshot` directory in the root of the file system. Snapshots are automatically mounted on demand and may be unmounted at regular intervals. The visibility of the `.zfs` directory can be controlled by the `snapdir` property.

Clones

A clone is a writable volume or file system whose initial contents are the same as the original dataset. As with snapshots, creating a clone is nearly instantaneous, and initially consumes no additional space.

Clones can only be created from a snapshot. When a snapshot is cloned, it creates an implicit dependency between the parent and child. Even though the clone is created somewhere else in the dataset hierarchy, the original snapshot cannot be destroyed as long as a clone exists. The `origin` property exposes this dependency, and the `zfs destroy` command lists any such dependencies, if they exist.

The clone parent-child dependency relationship can be reversed by using the `promote` subcommand. This causes the *origin* file system to become a clone of the specified file system, which makes it possible to destroy the file system that the clone was created from.

Mount Points

Creating a ZFS file system is a simple operation, so the number of file systems per system is likely to be numerous. To cope with this, ZFS automatically manages mounting and unmounting file systems without the need to edit the `/etc/vfstab` file. All automatically managed file systems are mounted by ZFS at boot time.

By default, file systems are mounted under */path*, where *path* is the name of the file system in the ZFS namespace. Directories are created and destroyed as needed.

A file system can also have a mount point set in the `mountpoint` property. This directory is created as needed, and ZFS automatically mounts the file system when the `zfs mount -a` command is invoked (without editing `/etc/vfstab`). The `mountpoint` property can be inherited, so if `pool/home` has a mount point of `/export/stuff`, then `pool/home/user` automatically inherits a mount point of `/export/stuff/user`.

A file system `mountpoint` property of `none` prevents the file system from being mounted.

If needed, ZFS file systems can also be managed with traditional tools (`mount`, `umount`, `/etc/vfstab`). If a file system's mount point is set to `legacy`, ZFS makes no attempt to manage the file system, and the administrator is responsible for mounting and unmounting the file system.

Zones

A ZFS file system can be added to a non-global zone by using the `zonecfg add fs` subcommand. A ZFS file system that is added to a non-global zone must have its `mountpoint` property set to `legacy`.

The physical properties of an added file system are controlled by the global administrator. However, the zone administrator can create, modify, or destroy files within the added file system, depending on how the file system is mounted.

A dataset can also be delegated to a non-global zone by using the `zonecfg add dataset` subcommand. You cannot delegate a dataset to one zone and the children of the same dataset to another zone. The zone administrator can change properties of the dataset or any of its children. However, the `quota` property is controlled by the global administrator.

A ZFS volume can be added as a device to a non-global zone by using the `zonecfg add device` subcommand. However, its physical properties can be modified only by the global administrator.

For more information about `zonecfg` syntax, see `zonecfg(1M)`.

After a dataset is delegated to a non-global zone, the `zoned` property is automatically set. A zoned file system cannot be mounted in the global zone, since the zone administrator might have to set the mount point to an unacceptable value.

The global administrator can forcibly clear the `zoned` property, though this should be done with extreme care. The global administrator should verify that all the mount points are acceptable before clearing the property.

Native Properties

Properties are divided into two types, native properties and user-defined (or *user*) properties. Native properties either export internal statistics or control ZFS behavior. In addition, native properties are either editable or read-only. User properties have no effect on ZFS behavior, but you can use them to annotate datasets in a way that is meaningful in your environment. For more information about user properties, see the "User Properties" section, below.

Every dataset has a set of properties that export statistics about the dataset as well as control various behaviors. Properties are inherited from the parent unless overridden by the child. Some properties apply only to certain types of datasets (file systems, volumes, or snapshots).

The values of numeric properties can be specified using human-readable suffixes (for example, `k`, `KB`, `M`, `Gb`, and so forth, up to `Z` for zettabyte). The following are all valid (and equal) specifications:

```
1536M, 1.5g, 1.50GB
```

The values of non-numeric properties are case sensitive and must be lowercase, except for `mountpoint`, `sharenfs`, and `sharesmb`.

The following native properties consist of read-only statistics about the dataset. These properties can be neither set, nor inherited. Native properties apply to all dataset types unless otherwise noted.

`available`
The amount of space available to the dataset and all its children, assuming no other activity in the pool. Because space is shared within a pool, availability can be limited by any number of factors, including physical pool size, quotas, reservations, or other datasets within the pool.

This property can also be referred to by its shortened column name, `avail`.

`compressratio`
The compression ratio achieved for this dataset, expressed as a multiplier. Compression can be turned on by running: `zfs set compression=on` *dataset*. The default value is `off`.

`creation`
The time this dataset was created.

`defer_destroy`
This property is `on` if the snapshot has been marked for deferred destroy by using the `zfs destroy -d` command. Otherwise, the property is `off`.

`mounted`
For file systems, indicates whether the file system is currently mounted. This property can be either `yes` or `no`.

`origin`
For cloned file systems or volumes, the snapshot from which the clone was created. The origin cannot be destroyed (even with the `-r` or `-f` options) so long as a clone exists.

`referenced`
The amount of data that is accessible by this dataset, which may or may not be shared with other datasets in the pool. When a snapshot or clone is created, it initially references the same amount of space as the file system or snapshot it was created from, since its contents are identical.

This property can also be referred to by its shortened column name, `refer`.

`type`
The type of dataset: `filesystem`, `volume`, or `snapshot`.

`used`
The amount of space consumed by this dataset and all its descendents. This is the value that is checked against this dataset's quota and reservation. The space used does not include this

dataset's reservation, but does take into account the reservations of any descendent datasets. The amount of space that a dataset consumes from its parent, as well as the amount of space that is freed if this dataset is recursively destroyed, is the greater of its space used and its reservation.

When snapshots (see the "Snapshots" section) are created, their space is initially shared between the snapshot and the file system, and possibly with previous snapshots. As the file system changes, space that was previously shared becomes unique to the snapshot, and counted in the snapshot's space used. Additionally, deleting snapshots can increase the amount of space unique to (and used by) other snapshots.

The amount of space used, available, or referenced does not take into account pending changes. Pending changes are generally accounted for within a few seconds. Committing a change to a disk using `fsync(3C)` or `O_SYNC` does not necessarily guarantee that the space usage information is updated immediately.

`usedby*`
The `usedby*` properties decompose the `used` properties into the various reasons that space is used. Specifically, `used` = `usedbychildren` + `usedbydataset` + `usedbyrefreservation` +, `usedbysnapshots`. These properties are only available for datasets created on version 13 pools.

`usedbychildren`
The amount of space used by children of this dataset, which would be freed if all the dataset's children were destroyed.

`usedbydataset`
The amount of space used by this dataset itself, which would be freed if the dataset were destroyed (after first removing any `refreservation` and destroying any necessary snapshots or descendents).

`usedbyrefreservation`
The amount of space used by a `refreservation` set on this dataset, which would be freed if the `refreservation` was removed.

`usedbysnapshots`
The amount of space consumed by snapshots of this dataset. In particular, it is the amount of space that would be freed if all of this dataset's snapshots were destroyed. Note that this is not simply the sum of the snapshots' `used` properties because space can be shared by multiple snapshots.

`userused@`*user*
The amount of space consumed by the specified user in this dataset. Space is charged to the owner of each file, as displayed by `ls -l`. The amount of space charged is displayed by `du` and `ls -s`. See the `zfs userspace` subcommand for more information.

Unprivileged users can access only their own space usage. The root user, or a user who has been granted the `userused` privilege with `zfs allow`, can access everyone's usage.

The `userused@...` properties are not displayed by `zfs get all`. The user's name must be appended after the `@` symbol, using one of the following forms:

- *POSIX name* (for example, `joe`)
- *POSIX numeric ID* (for example, `789`)
- *SID name* (for example, `joe.smith@mydomain`)
- *SID numeric ID* (for example, `S-1-123-456-789`)

`userrefs`
This property is set to the number of user holds on this snapshot. User holds are set by using the `zfs hold` command.

`groupused@`*group*
The amount of space consumed by the specified group in this dataset. Space is charged to the group of each file, as displayed by `ls -l`. See the `userused@`*user* property for more information.

Unprivileged users can only access their own groups' space usage. The root user, or a user who has been granted the `groupused` privilege with `zfs allow`, can access all groups' usage.

`volblocksize=`*blocksize*
For volumes, specifies the block size of the volume. The `blocksize` cannot be changed once the volume has been written, so it should be set at volume creation time. The default `blocksize` for volumes is 8 KB. Any power of 2 from 512 bytes to 128 KB is valid.

This property can also be referred to by its shortened column name, `volblock`.

The following native properties can be used to change the behavior of a ZFS dataset.

`aclinherit=discard` | `noallow` | `restricted` | `passthrough` | `passthrough-x`
Controls how ACL entries are inherited when files and directories are created. A file system with an `aclinherit` property of `discard` does not inherit any ACL entries. A file system with an `aclinherit` property value of `noallow` only inherits inheritable ACL entries that specify deny permissions. The property value `restricted` (the default) removes the `write_acl` and `write_owner` permissions when the ACL entry is inherited. A file system with an `aclinherit` property value of `passthrough` inherits all inheritable ACL entries without any modifications made to the ACL entries when they are inherited. A file system with an `aclinherit` property value of `passthrough-x` has the same meaning as `passthrough`, except that the `owner@`, `group@`, and `everyone@` ACEs inherit the execute permission only if the file creation mode also requests the execute bit.

When the property value is set to `passthrough`, files are created with a mode determined by the inheritable ACEs. If no inheritable ACEs exist that affect the mode, then the mode is set in accordance to the requested mode from the application.

`aclmode=discard` | `groupmask` | `passthrough`
Controls how an ACL is modified during chmod(2). A file system with an `aclmode` property of `discard` deletes all ACL entries that do not represent the mode of the file. An `aclmode` property of `groupmask` (the default) reduces user or group permissions. The permissions

are reduced, such that they are no greater than the group permission bits, unless it is a user entry that has the same UID as the owner of the file or directory. In this case, the ACL permissions are reduced so that they are no greater than owner permission bits. A file system with an `aclmode` property of `passthrough` indicates that no changes are made to the ACL other than generating the necessary ACL entries to represent the new mode of the file or directory.

`atime=on | off`
Controls whether the access time for files is updated when they are read. Turning this property off avoids producing write traffic when reading files and can result in significant performance gains, though it might confuse mailers and other similar utilities. The default value is `on`.

`canmount=on | off | noauto`
If this property is set to `off`, the file system cannot be mounted, and is ignored by `zfs mount -a`. Setting this property to `off` is similar to setting the `mountpoint` property to `none`, except that the dataset still has a normal `mountpoint` property, which can be inherited. Setting this property to `off` allows datasets to be used solely as a mechanism to inherit properties. One example of setting `canmount=off` is to have two datasets with the same `mountpoint`, so that the children of both datasets appear in the same directory, but might have different inherited characteristics.

When the `noauto` option is set, a dataset can only be mounted and unmounted explicitly. The dataset is not mounted automatically when the dataset is created or imported, nor is it mounted by the `zfs mount -a` command or unmounted by the `zfs unmount -a` command.

This property is not inherited.

`checksum=on | off | fletcher2, | fletcher4 | sha256`
Controls the checksum used to verify data integrity. The default value is `on`, which automatically selects the `fletcher4` algorithm. The value `off` disables integrity checking on user data. Disabling checksums is *NOT* a recommended practice.

Changing this property affects only newly-written data.

`compression=on | off | lzjb | gzip | gzip-`*N*` | zle`
Controls the compression algorithm used for this dataset. The `lzjb` compression algorithm is optimized for performance while providing decent data compression. Setting compression to `on` uses the `lzjb` compression algorithm. The `gzip` compression algorithm uses the same compression as the `gzip`(1) command. You can specify the `gzip` level by using the value `gzip-`*N* where *N* is an integer from 1 (fastest) to 9 (best compression ratio). Currently, `gzip` is equivalent to `gzip-6` (which is also the default for `gzip`(1)).

This property can also be referred to by its shortened column name `compress`. Changing this property affects only newly-written data.

`copies=1 | 2 | 3`
Controls the number of copies of data stored for this dataset. These copies are in addition to any redundancy provided by the pool, for example, mirroring or RAID-Z. The copies are stored on different disks, if possible. The space used by multiple copies is charged to the associated file and dataset, changing the `used` property and counting against quotas and reservations.

Changing this property only affects newly-written data. Therefore, set this property at file system creation time by using the `-o copies=`*N* option.

`devices=on | off`
Controls whether device nodes can be opened on this file system. The default value is `on`.

`exec=on | off`
Controls whether processes can be executed from within this file system. The default value is `on`.

`mountpoint=`*path* `| none | legacy`
Controls the mount point used for this file system. See the "Mount Points" section for more information on how this property is used.

When the `mountpoint` property is changed for a file system, the file system and any children that inherit the mount point are unmounted. If the new value is `legacy`, then they remain unmounted. Otherwise, they are automatically remounted in the new location if the property was previously `legacy` or `none`, or if they were mounted before the property was changed. In addition, any shared file systems are unshared and shared in the new location.

`nbmand=on | off`
Controls whether the file system should be mounted with `nbmand` (Non Blocking mandatory locks). This is used for SMB clients. Changes to this property only take effect when the file system is umounted and remounted. See `mount(1M)` for more information on `nbmand` mounts.

This SMB related property is not fully functional in the Oracle Solaris 10 release because the Oracle Solaris SMB server is not supported in the Oracle Solaris 10 release.

`primarycache=all | none | metadata`
Controls what is cached in the primary cache (ARC). If this property is set to `all`, then both user data and metadata is cached. If this property is set to `none`, then neither user data nor metadata is cached. If this property is set to `metadata`, then only metadata is cached. The default value is `all`.

`quota=`*size* `| none`
Limits the amount of space a dataset and its descendents can consume. This property enforces a hard limit on the amount of space used. This includes all space consumed by

descendents, including file systems and snapshots. Setting a quota on a descendent of a dataset that already has a quota does not override the ancestor's quota, but rather imposes an additional limit.

Quotas cannot be set on volumes, as the `volsize` property acts as an implicit quota.

`userquota@`*user*`=`*size*`|none`
Limits the amount of space consumed by the specified user. Similar to the `refquota` property, the `userquota` space calculation does not include space that is used by descendent datasets, such as snapshots and clones. User space consumption is identified by the `userspace@`*user* property.

Enforcement of user quotas may be delayed by several seconds. This delay means that a user might exceed her quota before the system notices that she is over quota. The system would then begin to refuse additional writes and displays the `EDQUOT` error message. See the `zfs userspace` subcommand for more information.

Unprivileged users can only access their own groups' space usage. The root user, or a user who has been granted the `userquota` privilege with `zfs allow`, can get and set everyone's quota.

This property is not available on volumes, on file systems before version 4, or on pools before version 15. The `userquota@...` properties are not displayed by `zfs get all`. The user's name must be appended after the `@` symbol, using one of the following forms:

- *POSIX name* (for example, `joe`)
- *POSIX numeric ID* (for example, `789`)
- *SID name* (for example, `joe.smith@mydomain`)
- *SID numeric ID* (for example, `S-1-123-456-789`)

`groupquota@`*group*`=`*size*`|none`
Limits the amount of space consumed by the specified group. Group space consumption is identified by the `userquota@`*user* property.

Unprivileged users can access only their own groups' space usage. The root user, or a user who has been granted the `groupquota` privilege with `zfs allow`, can get and set all groups' quotas.

`readonly=on|off`
Controls whether this dataset can be modified. The default value is `off`.

This property can also be referred to by its shortened column name, `rdonly`.

`recordsize=`*size*
Specifies a suggested block size for files in the file system. This property is designed solely for use with database workloads that access files in fixed-size records. ZFS automatically tunes block sizes according to internal algorithms optimized for typical access patterns.

For databases that create very large files but access them in small random chunks, these algorithms may be suboptimal. Specifying a `recordsize` greater than or equal to the record size of the database can result in significant performance gains. Use of this property for general purpose file systems is strongly discouraged, and may adversely affect performance.

The size specified must be a power of two greater than or equal to 512 and less than or equal to 128 Kbytes.

Changing the file system's `recordsize` affects only files created afterward; existing files are unaffected.

This property can also be referred to by its shortened column name, `recsize`.

`refquota=`*size* | `none`
Limits the amount of space a dataset can consume. This property enforces a hard limit on the amount of space used. This hard limit does not include space used by descendents, including file systems and snapshots.

`refreservation=`*size* | `none`
The minimum amount of space guaranteed to a dataset, not including its descendents. When the amount of space used is below this value, the dataset is treated as if it were taking up the amount of space specified by `refreservation`. The `refreservation` reservation is accounted for in the parent datasets' space used, and counts against the parent datasets' quotas and reservations.

If `refreservation` is set, a snapshot is only allowed if there is enough free pool space outside of this reservation to accommodate the current number of referenced bytes in the dataset.

This property can also be referred to by its shortened column name, `refreserv`.

`reservation=`*size* | `none`
The minimum amount of space guaranteed to a dataset and its descendents. When the amount of space used is below this value, the dataset is treated as if it were taking up the amount of space specified by its reservation. Reservations are accounted for in the parent datasets' space used, and count against the parent datasets' quotas and reservations.

This property can also be referred to by its shortened column name, `reserv`.

`secondarycache=all` | `none` | `metadata`
Controls what is cached in the secondary cache (L2ARC). If this property is set to `all`, then both user data and metadata is cached. If this property is set to `none`, then neither user data nor metadata is cached. If this property is set to `metadata`, then only metadata is cached. The default value is `all`.

`setuid=on` | `off`
Controls whether the set-UID bit is enabled for the file system. The default value is `on`.

`shareiscsi=on | off`
Like the `sharenfs` property, `shareiscsi` indicates whether a ZFS volume is shared as an iSCSI target. The acceptable values for this property are `on`, `off`, and `type=disk`. The default value is `off`.

You might want to set `shareiscsi=on` for a file system so that all ZFS volumes within the file system are shared by default. However, setting this property on a file system has no direct effect.

`sharesmb=on | off |` *opts*
Controls whether the file system is shared by using the Oracle Solaris SMB service, and what options are to be used. A file system with the `sharesmb` property set to `off` is managed through traditional UNIX tools. Otherwise, the file system is automatically shared and unshared with the `zfs share` and `zfs unshare` commands.

Because SMB shares requires a resource name, a unique resource name is constructed from the dataset name. The constructed name is a copy of the dataset name except that the characters in the dataset name, which would be illegal in the resource name, are replaced with underscore (_) characters. A pseudo property `name` is also supported that allows you to replace the data set name with a specified name. The specified name is then used to replace the prefix dataset in the case of inheritance. For example, if the dataset `data/home/john` is set to `name=john`, then `data/home/john` has a resource name of `john`. If a child dataset of `data/home/john/backups`, it has a resource name of `john_backups`.

When SMB shares are created, the SMB share name appears as an entry in the `.zfs/shares` directory. You can use the `ls` or `chmod` command to display the share-level ACLs on the entries in this directory.

When the `sharesmb` property is changed for a dataset, the dataset and any children inheriting the property are re-shared with the new options, only if the property was previously set to `off`, or if they were shared before the property was changed. If the new property is set to `off`, the file systems are unshared.

Note that the Oracle Solaris SMB service is not supported in the Oracle Solaris 10 release.

`sharenfs=on | off |` *opts*
Controls whether the file system is shared over NFS, and what options are used. A file system with a `sharenfs` property of `off` is managed through traditional tools such as `share(1M)`, `unshare(1M)`, and `dfstab(4)`. Otherwise, the file system is automatically shared and unshared with the `zfs share` and `zfs unshare` commands. If the property is set to `on`, the `share(1M)` command is invoked with no options. Otherwise, the `share(1M)` command is invoked with options equivalent to the contents of this property.

When the `sharenfs` property is changed for a dataset, the dataset and any children inheriting the property are re-shared with the new options, only if the property was previously `off`, or if they were shared before the property was changed. If the new property is `off`, the file systems are unshared.

`logbias` = `latency` | `throughput`
Provides a hint to ZFS about handling of synchronous requests in this dataset. If `logbias` is set to `latency` (the default), ZFS uses the pool's log devices (if configured) to handle the requests at low latency. If `logbias` is set to `throughput`, ZFS does not use the configured pool log devices. Instead, ZFS optimizes synchronous operations for global pool throughput and efficient use of resources.

`snapdir=hidden` | `visible`
Controls whether the `.zfs` directory is hidden or visible in the root of the file system as discussed in the "Snapshots" section. The default value is `hidden`.

`version=`*version* | `current`
The on-disk version of this file system, which is independent of the pool version. This property can only be set to later supported versions. See the `zfs upgrade` command.

`volsize=`*size*
For volumes, specifies the logical size of the volume. By default, creating a volume establishes a reservation of equal size. For storage pools with a version number of 9 or higher, a `refreservation` is set instead. Any changes to `volsize` are reflected in an equivalent change to the reservation (or `refreservation`). The `volsize` can only be set to a multiple of `volblocksize`, and cannot be zero.

The reservation is kept equal to the volume's logical size to prevent unexpected behavior for consumers. The reservation size corresponds to the volume's logical size, increased by ZFS implementation overhead. Without the reservation, the volume could run out of space, resulting in undefined behavior or data corruption, depending on how the volume is used. These effects can also occur when the volume size is changed while it is in use (particularly when shrinking the size). Extreme care should be used when adjusting the volume size.

Though not recommended, a *sparse volume* (also known as *thin provisioning*) can be created by specifying the `-s` option to the `zfs create -V` command, or by changing the reservation after the volume has been created. A sparse volume is a volume where the reservation is less then the volume size. Consequently, writes to a sparse volume can fail with the `ENOSPC` message when the pool is low on space. For a sparse volume, changes to `volsize` are not reflected in the reservation.

`vscan=on` | `off`
Controls whether regular files should be scanned for viruses when a file is opened and closed. In addition to enabling this property, the virus scan service must also be enabled for virus scanning to occur. The default value is `off`.

This SMB related property is not fully functional in the Oracle Solaris 10 release because the Oracle Solaris SMB server is not supported in the Oracle Solaris 10 release.

`xattr=on` | `off`
Controls whether extended attributes are enabled for this file system. The default value is `on`.

`zoned=on | off`
Controls whether the dataset is managed from a non-global zone. See the "Zones" section for more information. The default value is `off`.

The following three properties cannot be changed after the file system is created, and therefore, should be set when the file system is created. If the properties are not set with the `zfs create` or `zpool create` commands, these properties are inherited from the parent dataset. If the parent dataset lacks these properties due to having been created prior to these features being supported, the new file system will have the default values for these properties.

`casesensitivity=sensitive | insensitive | mixed`
Indicates whether the file name matching algorithm used by the file system should be case-sensitive, case-insensitive, or allow a combination of both styles of matching. The default value for the `casesensitivity` property is `sensitive`. Traditionally, UNIX and POSIX file systems have case-sensitive file names.

The `mixed` value for the `casesensitivity` property indicates that the file system can support requests for both case-sensitive and case-insensitive matching behavior. Currently, case-insensitive matching behavior on a file system that supports mixed behavior is limited to the Oracle Solaris SMB server product, which is not supported in the Oracle Solaris 10 release. For more information about the `mixed` value behavior, see the *Oracle Solaris ZFS Administration Guide*.

This SMB related property is not fully functional in the Oracle Solaris 10 release because the Oracle Solaris SMB server is not supported in the Oracle Solaris 10 release.

`normalization = none | formC | formD | formKC | formKD`
Indicates whether the file system should perform a `unicode` normalization of file names whenever two file names are compared, and which normalization algorithm should be used. File names are always stored unmodified, names are normalized as part of any comparison process. If this property is set to a legal value other than `none`, and the `utf8only` property was left unspecified, the `utf8only` property is automatically set to `on`. The default value of the `normalization` property is `none`. This property cannot be changed after the file system is created.

This SMB related property is not fully functional in the Oracle Solaris 10 release because the Oracle Solaris SMB server is not supported in the Oracle Solaris 10 release.

`utf8only=on | off`
Indicates whether the file system should reject file names that include characters that are not present in the UTF-8 character code set. If this property is explicitly set to `off`, the normalization property must either not be explicitly set or be set to `none`. The default value for the `utf8only` property is `off`. This property cannot be changed after the file system is created.

This SMB related property is not fully functional in the Oracle Solaris 10 release because the Oracle Solaris SMB server is not supported in the Oracle Solaris 10 release.

Temporary Mount Point Properties

When a file system is mounted, either through mount(1M) for legacy mounts or the `zfs mount` command for normal file systems, its mount options are set according to its properties. The correlation between properties and mount options is as follows:

PROPERTY	MOUNT OPTION
devices	devices/nodevices
exec	exec/noexec
readonly	ro/rw
setuid	setuid/nosetuid
xattr	xattr/noxattr

In addition, these options can be set on a per-mount basis using the `-o` option, without affecting the property that is stored on disk. The values specified on the command line override the values stored in the dataset. The `-nosuid` option is an alias for `nodevices,nosetuid`. These properties are reported as *temporary* by the `zfs get` command. If the properties are changed while the dataset is mounted, the new setting overrides any temporary settings.

User Properties

In addition to the standard native properties, ZFS supports arbitrary user properties. User properties have no effect on ZFS behavior, but applications or administrators can use them to annotate datasets (file systems, volumes, and snapshots).

User property names must contain a colon (`:`) character to distinguish them from native properties. They may contain lowercase letters, numbers, and the following punctuation characters: colon (`:`), dash (`-`), period (`.`), and underscore (`_`). The expected convention is that the property name is divided into two portions such as *module*`:`*property*, but this namespace is not enforced by ZFS. User property names can be at most 256 characters, and cannot begin with a dash (`-`).

When making programmatic use of user properties, it is strongly suggested to use a reversed DNS domain name for the *module* component of property names to reduce the chance that two independently-developed packages use the same property name for different purposes. Property names beginning with `com.sun.` are reserved for use by Sun Microsystems.

The values of user properties are arbitrary strings, are always inherited, and are never validated. All of the commands that operate on properties (`zfs list`, `zfs get`, `zfs set`, and so forth) can be used to manipulate both native properties and user properties. Use the `zfs inherit` command to clear a user property . If the property is not defined in any parent dataset, it is removed entirely. Property values are limited to 1024 characters.

ZFS Volumes as Swap or Dump Devices

During an initial installation or a live upgrade from a UFS file system, a swap device and dump device are created on ZFS volumes in the ZFS root pool. By default, the swap area size is based on 1/2 the size of physical memory up to 2 GB. The size of the dump device depends on the kernel's requirements at installation time. Separate ZFS volumes must be used for the swap area and dump devices. Do not swap to a file on a ZFS file system. A ZFS swap file configuration is not supported.

If you need to change your swap area or dump device after the system is installed or upgraded, use the swap(1M) and dumpadm(1M) commands. If you need to change the size of your swap area or dump device, see the *Oracle Solaris ZFS Administration Guide*

Subcommands All subcommands that modify state are logged persistently to the pool in their original form.

`zfs ?`
Displays a help message.

`zfs create` [`-p`] [`-o` *property=value*] ... *filesystem*
Creates a new ZFS file system. The file system is automatically mounted according to the `mountpoint` property inherited from the parent.

`-p`
Creates the non-existing parent datasets. Datasets created in this manner are automatically mounted according to the `mountpoint` property inherited from their parent. Any property specified on the command line using the `-o` option is ignored. If the target filesystem already exists, the operation completes successfully.

`-o` *property=value*
Sets the specified property as if the command `zfs set` *property=value* was invoked at the same time the dataset was created. Any editable ZFS property can also be set at creation time. Multiple `-o` options can be specified. An error results if the same property is specified in multiple `-o` options.

`zfs create` [`-ps`] [`-b` *blocksize*] [`-o` *property=value*] ... `-V` *size volume*
Creates a volume of the given size. The volume is exported as a block device in `/dev/zvol/{dsk,rdsk}/`*path*, where *path* is the name of the volume in the ZFS namespace. The size represents the logical size as exported by the device. By default, a reservation of equal size is created.

size is automatically rounded up to the nearest 128 KB to ensure that the volume has an integral number of blocks regardless of *blocksize*.

`-p`
Creates the non-existing parent datasets. Datasets created in this manner are automatically mounted according to the `mountpoint` property inherited from their parent. Any property specified on the command line using the `-o` option is ignored. If the target file system already exists, the operation completes successfully.

`-s`
Creates a sparse volume with no reservation. See `volsize` in the "Native Properties" section for more information about sparse volumes.

`-o` *property=value*
Sets the specified property as if the `zfs set` *property=value* command was invoked at the same time the dataset was created. Any editable ZFS property can also be set at creation time. Multiple `-o` options can be specified. An error results if the same property is specified in multiple `-o` options.

`-b` *blocksize*
Equivalent to `-o volblocksize=`*blocksize*. If this option is specified in conjunction with `-o volblocksize`, the resulting behavior is undefined.

`zfs destroy` [`-rRf`] *filesystem*|*volume*
Destroys the given dataset. By default, the command unshares any file systems that are currently shared, unmounts any file systems that are currently mounted, and refuses to destroy a dataset that has active dependents (children or clones).

`-r`
Recursively destroy all children.

`-R`
Recursively destroy all dependents, including cloned file systems outside the target hierarchy.

`-f`
Force an unmount of any file systems using the `unmount -f` command. This option has no effect on non-file systems or unmounted file systems.

Extreme care should be taken when applying either the `-r` or the `-f` options, as they can destroy large portions of a pool and cause unexpected behavior for mounted file systems in use.

`zfs destroy` [`-rRd`] *snapshot*
The given snapshot is destroyed immediately if and only if the `zfs destroy` command without the `-d` option would have destroyed it. Such immediate destruction would occur, for example, if the snapshot had no clones and the user-initiated reference count were zero.

If the snapshot does not qualify for immediate destruction, it is marked for deferred deletion. In this state, it exists as a usable, visible snapshot until both of the preconditions listed above are met, at which point it is destroyed.

`-d`
Defer snapshot deletion.

`-r`
Destroy (or mark for deferred deletion) all snapshots with this name in descendent file systems.

`-R`
Recursively destroy all dependents.

`zfs snapshot` [`-r`] [`-o` *property=value*] ... *filesystem@snapname*|*volume@snapname*
Creates a snapshot with the given name. All previous modifications by successful system calls to the file system are part of the snapshot. See the "Snapshots" section for details.

`-r`
Recursively create snapshots of all descendent datasets. Snapshots are taken atomically, so that all recursive snapshots correspond to the same moment in time.

`-o` *property=value*
Sets the specified property; see `zfs create` for details.

`zfs rollback` [`-rRf`] *snapshot*
Roll back the given dataset to a previous snapshot. When a dataset is rolled back, all data that has changed since the snapshot is discarded, and the dataset reverts to the state at the time of the snapshot. By default, the command refuses to roll back to a snapshot other than the most recent one. In order to do so, all intermediate snapshots must be destroyed by specifying the `-r` option.

The `-rR` options do not recursively destroy the child snapshots of a recursive snapshot. Only the top-level recursive snapshot is destroyed by either of these options. To completely roll back a recursive snapshot, you must rollback the individual child snapshots.

`-r`
Recursively destroy any snapshots more recent than the one specified.

`-R`
Recursively destroy any more recent snapshots, as well as any clones of those snapshots.

`-f`
Used with the `-R` option to force an unmount of any clone file systems that are to be destroyed.

`zfs clone` [`-p`] [`-o` *property=value*] ... *snapshot filesystem|volume*
Creates a clone of the given snapshot. See the "Clones" section for details. The target dataset can be located anywhere in the ZFS hierarchy, and is created as the same type as the original.

`-p`
Creates the non-existing parent datasets. Datasets created in this manner are automatically mounted according to the `mountpoint` property inherited from their parent. If the target filesystem or volume already exists, the operation completes successfully.

`-o` *property=value*
Sets the specified property; see `zfs create` for details.

`zfs promote` *clone-filesystem*
Promotes a clone file system to no longer be dependent on its "origin" snapshot. This makes it possible to destroy the file system that the clone was created from. The clone parent-child dependency relationship is reversed, so that the origin file system becomes a clone of the specified *filesystem*.

The snapshot that was cloned, and any snapshots previous to this snapshot, are now owned by the promoted clone. The space they use moves from the origin file system to the promoted clone, so enough space must be available to accommodate these snapshots. No new space is consumed by this operation, but the space accounting is adjusted. The

promoted clone must not have any conflicting snapshot names of its own. The `rename` subcommand can be used to rename any conflicting snapshots.

`zfs rename` *filesystem*|*volume*|*snapshot*
filesystem|*volume*|*snapshot*
`zfs rename` [`-p`] *filesystem*|*volume filesystem*|*volume*

Renames the given dataset. The new target can be located anywhere in the ZFS hierarchy, with the exception of snapshots. Snapshots can only be renamed within the parent file system or volume. When renaming a snapshot, the parent file system of the snapshot does not need to be specified as part of the second argument. Renamed file systems can inherit new mount points, in which case they are unmounted and remounted at the new mount point.

`-p`

Creates the nonexistent parent datasets. Datasets created in this manner are automatically mounted according to the `mountpoint` property inherited from their parent.

`zfs rename -r` *snapshot snapshot*

Recursively rename the snapshots of all descendent datasets. Snapshots are the only dataset that can be renamed recursively.

`zfs list` [`-r`|`-d` *depth*] [`-H`] [`-o` *property*[,...]] [`-t` *type*[,...]] [`-s` *property*] ... [`-S` *property*] ... [*filesystem*|*volume*|*snapshot*] ...

Lists the property information for the given datasets in tabular form. If specified, you can list property information by the absolute pathname or the relative pathname. By default, all file systems and volumes are displayed. Snapshots are displayed if the `listsnaps` property is `on` (the default is `off`) . The following fields are displayed, `name,used,available,referenced,mountpoint`.

`-H`

Used for scripting mode. Do not print headers and separate fields by a single tab instead of arbitrary white space.

`-r`

Recursively display any children of the dataset on the command line.

`-d` *depth*

Recursively display any children of the dataset, limiting the recursion to *depth*. A depth of `1` will display only the dataset and its direct children.

`-o` *property*

A comma-separated list of properties to display. The property must be:

- One of the properties described in the "Native Properties" section
- A user property
- The value `name` to display the dataset name

- The value `space` to display space usage properties on file systems and volumes. This is a shortcut for specifying
  `-o name,avail,used,usedsnap,usedds,usedrefreserv,usedchild`
  `-t filesystem,volume` syntax.

`-s` *property*
A property for sorting the output by column in ascending order based on the value of the property. The property must be one of the properties described in the "Properties" section, or the special value `name` to sort by the dataset name. Multiple properties can be specified at one time using multiple `-s` property options. Multiple `-s` options are evaluated from left to right in decreasing order of importance.

The following is a list of sorting criteria:

- Numeric types sort in numeric order.
- String types sort in alphabetical order.
- Types inappropriate for a row sort that row to the literal bottom, regardless of the specified ordering.
- If no sorting options are specified the existing behavior of `zfs list` is preserved.

`-S` *property*
Same as the `-s` option, but sorts by property in descending order.

`-t` *type*
A comma-separated list of types to display, where *type* is one of `filesystem`, `snapshot`, `volume`, or `all`. For example, specifying `-t snapshot` displays only snapshots.

`zfs set` *property=value filesystem*|*volume*|*snapshot* ...
Sets the property to the given value for each dataset. Only some properties can be edited. See the "Properties" section for more information on what properties can be set and acceptable values. Numeric values can be specified as exact values, or in a human-readable form with a suffix of `B`, `K`, `M`, `G`, `T`, `P`, `E`, `Z` (for bytes, kilobytes, megabytes, gigabytes, terabytes, petabytes, exabytes, or zettabytes, respectively). User properties can be set on snapshots. For more information, see the "User Properties" section.

`zfs get` [`-r`|`-d` *depth*] [`-Hp`] [`-o all` | *field*[,...] [`-s` *source*[,...]] `all` | *property*[,...] *filesystem*|*volume*|*snapshot* ...
Displays properties for the given datasets. If no datasets are specified, then the command displays properties for all datasets on the system. For each property, the following columns are displayed:

```
name      Dataset name
property  Property name
value     Property value
source    Property source. Can either be local, default,
          temporary, inherited, or none (-).
```

All columns except the `RECEIVED` column are displayed by default; specify particular or all columns, using the `-o` option. This command takes a comma-separated list of properties as described in the "Native Properties" and "User Properties" sections.

The special value `all` can be used to display all properties that apply to the given dataset's type (file system, volume, or snapshot).

`-r`
Recursively display properties for any children.

`-d` *depth*
Recursively display any children of the dataset, limiting the recursion to *depth*. A depth of `1` will display only the dataset and its direct children.

`-H`
Display output in a form more easily parsed by scripts. Any headers are omitted, and fields are explicitly separated by a single tab instead of an arbitrary amount of space.

`-o` *field*
Set of fields to display. One or more of:

```
name,property,value,received,source
```

Present multiple fields as a comma-separated list. The default value is:

```
name,property,value,source
```

The keyword `all` specifies all sources.

`-s` *source*
A comma-separated list of sources to display. Those properties coming from a source other than those in this list are ignored. Each source must be one of the following:

```
local,default,inherited,temporary,received,none
```

The default value is all sources.

`-p`
Display numbers in parseable (exact) values.

`zfs inherit` [`-rS`] *property filesystem*|*volume*|*snapshot* ...
Clears the specified property, causing it to be inherited from an ancestor. If no ancestor has the property set, then the default value is used. See the "Properties" section for a listing of default values, and details on which properties can be inherited.

`-r`
Recursively inherit the given property for all children.

`-S`
Revert to the received property value, if any. If the property does not have a received value, the behavior of `zfs inherit -S` is the same as `zfs inherit` without `-S`. If the property does have a received value, `zfs inherit` masks the received value with the

inherited value until `zfs inherit -S` reverts to the received value.

`zfs upgrade`
Identifies a file system version, which determines available file system features in the currently running software release. You can continue to use older file system versions, but some features might not be available. A file system can be upgraded by using the `zfs upgrade -a` command. You will not be able to access a file system of a later version on a system that runs an earlier software version.

`zfs upgrade -v`
Displays ZFS file system versions that are supported by the current software. The current ZFS file system version and all previously supported versions are displayed, along with a pithy explanation of the features provided with each version.

`zfs upgrade` [`-r`] [`-V` *version*] [`-a` | *filesystem*]
Upgrades a file system to a new on-disk version. Upgrading a file system means that it will no longer be accessible on a system running an older software version. A `zfs send` stream that is generated from a new file system snapshot cannot be accessed on a system that runs an older software version.

In general, the file system version is independent of the pool version. See `zpool(1M)` for information on the `zpool upgrade` command.

In some cases, the file system version and the pool version are interrelated and the pool version must be upgraded before the file system version can be upgraded.

`-a`
Upgrade all file systems on all imported pools.

filesystem
Upgrade the specified *filesystem*.

`-r`
Upgrade the specified *filesystem* and all descendent file systems.

`-V` *version*
Upgrade to the specified *version*. If the `-V` flag is not specified, this command upgrades to the most recent version. This option can only be used to increase the version number, and only up to the most recent version supported by this software.

`zfs userspace` [`-niHp`] [`-o` *field*[,...]] [`-sS` *field*]... [`-t` *type* [,...]] *filesystem* | *snapshot*
Displays space consumed by, and quotas on, each user in the specified *filesystem* or *snapshot*. This corresponds to the `userused@`*user* and `userquota@`*user* properties.

`-n`
Print numeric ID instead of user/group name.

`-H`
Do not print headers, use tab-delimited output.

`-p`
Use exact (parseable) numeric output.

`-o` *field*[,...]
Display only the specified fields from the following set, `type,name,used,quota`.The default is to display all fields.

`-s` *field*
Sort output by this field. The *s* and *S* flags may be specified multiple times to sort first by one field, then by another. The default is `-s type -s name`.

`-S` *field*
Sort by this field in reverse order. See `-s`.

`-t` *type*[,...]
Print only the specified types from the following set, `all,posixuser,smbuser,posixgroup,smbgroup`.

The default is `-t posixuser,smbuser`

The default can be changed to include group types.

`-i`
Translate SID to POSIX ID. The POSIX ID may be ephemeral if no mapping exists. Normal POSIX interfaces (for example, `stat(2)`, `ls -l`) perform this translation, so the `-i` option allows the output from `zfs userspace` to be compared directly with those utilities. However, `-i` may lead to confusion if some files were created by an SMB user before a SMB-to-POSIX name mapping was established. In such a case, some files are owned by the SMB entity and some by the POSIX entity. However, the `-i` option will report that the POSIX entity has the total usage and quota for both.

`zfs groupspace` [`-niHp`] [`-o` *field*[,...]] [`-sS` *field*]... [`-t` *type* [,...]] *filesystem* | *snapshot*
Displays space consumed by, and quotas on, each group in the specified *filesystem* or *snapshot*. This subcommand is identical to `zfs userspace`, except that the default types to display are `-t posixgroup,smbgroup`.

`-`

`zfs mount`
Displays all ZFS file systems currently mounted.

`zfs mount` [`-vO`] [`-o` *options*] `-a` | *filesystem*
Mounts ZFS file systems. Invoked automatically as part of the boot process.

`-o` *options*
An optional, comma-separated list of mount options to use temporarily for the duration of the mount. See the "Temporary Mount Point Properties" section for details.

`-O`
Perform an overlay mount. See `mount(1M)` for more information.

`-v`
Report mount progress.

`-a`
Mount all available ZFS file systems. Invoked automatically as part of the boot process.

filesystem
Mount the specified *filesystem*.

`zfs unmount` [`-f`] `-a` | *filesystem*|*mountpoint*
Unmounts currently mounted ZFS file systems. Invoked automatically as part of the shutdown process.

`-f`
Forcefully unmount the file system, even if it is currently in use.

`-a`
Unmount all available ZFS file systems. Invoked automatically as part of the boot process.

filesystem|*mountpoint*
Unmount the specified *filesystem*. The command can also be given a path to a ZFS file system mount point on the system.

`zfs share -a` | *filesystem*
Shares ZFS file systems that have the `sharenfs` or `sharesmb` property set. Sharing a file system with the NFS or SMB protocol means that the file system data is available over the network. ZFS file systems that have the `sharenfs` or `sharesmb` property set are automatically shared when a system is booted. For information about sharing a ZFS volume as an iSCSI target, see the `shareiscsi` property description.

Note that the Oracle Solaris SMB service is not supported in the Oracle Solaris 10 release.

`-a`
Shares all ZFS file systems that have the `sharenfs` or `sharesmb` property set and according to the `share` property values.

filesystem
Shares the specified *filesystem* that has the `sharenfs` or `sharesmb` property set and according to the `share` property values.

`zfs unshare -a` | *filesystem*|*mountpoint*
Unshares currently shared ZFS file systems that have the `sharenfs` or `sharesmb` property set. File systems that have the `sharenfs` or `sharesmb` property set are automatically unshared when a system is shutdown.

`-a`
Unshares all ZFS file systems that have the `sharenfs` or `sharesmb` property set.

filesystem|*mountpoint*
Unshares the specified *filesystem*. This command can also be given a path to a ZFS file system shared on the system.

`zfs send` [`-vRp`] [`-`[`iI`] *snapshot*] *snapshot*
Creates a stream representation of the second *snapshot*, which is written to standard output. The output can be redirected to a file or to a different system (for example, using `ssh(1)`. By default, a full stream is generated.

`-i` *snapshot*
Generate an incremental stream from the first *snapshot* to the second *snapshot*. The incremental source (the first *snapshot*) can be specified as the last component of the snapshot name (for example, the part after the `@`), and it is assumed to be from the same file system as the second *snapshot*.

If the destination is a clone, the source may be the origin snapshot, which must be fully specified (for example, `pool/fs@origin`, not just `@origin`).

`-I` *snapshot*
Generate a stream package that sends all intermediary snapshots from the first snapshot to the second snapshot. For example, `-I @a fs@d` is similar to `-i @a fs@b; -i @b fs@c; -i @c fs@d`. The incremental source snapshot may be specified as with the `-i` option.

`-R`
Generate a replication stream package to replicate the specified *filesystem* and all descendent file systems, up to the named snapshot. When received, all properties, snapshots, descendent file systems, and clones are preserved.

If the `-i` or `-I` flags are used in conjunction with the `-R` flag, an incremental replication stream is generated. The current values of properties, and current snapshot and file system names are set when the stream is received. If the `-F` flag is specified when this stream is received, snapshots and file systems that do not exist on the sending side are destroyed.

`-p`
Send properties.

`-v`
Print verbose information about the stream package generated.

The format of the stream is committed. You will be able to receive your streams on future versions of ZFS.

`zfs receive` [`-vnFu`] *filesystem*|*volume*|*snapshot*
`zfs receive` [`-vnFu`] [`-d` | `-e`] *filesystem*

Creates a snapshot whose contents are as specified in the stream provided on standard input. If a full stream is received, then a new file system is created as well. Streams are created using the `zfs send` subcommand, which by default creates a full stream. `zfs recv` can be used as an alias for `zfs receive`.

If an incremental stream is received, then the destination file system must already exist, and its most recent snapshot must match the incremental stream's source. For ZFS volumes, the destination device link is destroyed and recreated, which means the volume cannot be accessed during the `receive` operation.

When a snapshot replication package stream that is generated by using the `zfs send -R` command is received, any snapshots that do not exist on the sending location are destroyed by using the `zfs destroy -d` command.

The name of the snapshot (and file system, if a full stream is received) that this subcommand creates depends on the argument type and the `-d` or `-e` option.

If the argument is a snapshot name, the specified *snapshot* is created. If the argument is a file system or volume name, a snapshot with the same name as the sent snapshot is created within the specified *filesystem* or *volume*. If the `-d` or `-e` option is specified, the snapshot name is determined by appending the sent snapshot's name to the specified filesystem. If the `-d` option is specified, all but the pool name of the sent snapshot path is appended (for example, `b/c@1` appended from sent snapshot `a/b/c@1`), and if the `-e` option is specified, only the tail of the sent snapshot path is appended (for example, `c@1` appended from sent snapshot `a/b/c@1`). In the case of `-d`, any file systems needed to replicate the path of the sent snapshot are created within the specified *filesystem*.

`-d`
: Use all but the first element of the sent snapshot path (all but the pool name) to determine the name of the new snapshot as described in the paragraph above.

`-e`
: Use the last element of the sent snapshot path to determine the name of the new snapshot as described in the paragraph above.

`-u`
: File system that is associated with the received stream is not mounted.

`-v`
: Print verbose information about the stream and the time required to perform the receive operation.

`-n`
: Do not actually receive the stream. This can be useful in conjunction with the `-v` option to verify the name the receive operation would use.

`-F`
: Force a rollback of the file system to the most recent snapshot before performing the receive operation. If receiving an incremental replication stream (for example, one

generated by `zfs send -R -[iI]`), destroy snapshots and file systems that do not exist on the sending side.

`zfs allow` *filesystem* | *volume*
Displays permissions that have been delegated on the specified *filesystem* or *volume*. See the other forms of `zfs allow` for more information.

`zfs allow` [`-ldug`] `everyone`|*user*|*group*[,...] *perm*|*@setname*[,...] *filesystem*| *volume*
`zfs allow` [`-ld`] `-e` *perm*|*@setname*[,...] *filesystem* | *volume*
Delegates ZFS administration permission for the file systems to non-privileged users.

[`-ug`] `everyone`|*user*|*group*[,...]
Specifies to whom the permissions are delegated. Multiple entities can be specified as a comma-separated list. If neither of the `-ug` options are specified, then the argument is interpreted preferentially as the keyword `everyone`, then as a user name, and lastly as a group name. To specify a user or group named `everyone`, use the `-u` or `-g` options. To specify a group with the same name as a user, use the `-g` options.

[`-e`] *perm*|*@setname*[,...]
Specifies that the permissions be delegated to `everyone`. Multiple permissions may be specified as a comma-separated list. Permission names are the same as ZFS subcommand and property names. See the property list below. Property set names, which begin with an at sign (`@`) , may be specified. See the `-s` form below for details.

[`-ld`] *filesystem*|*volume*
Specifies where the permissions are delegated. If neither of the `-ld` options are specified, or both are, then the permissions are allowed for the file system or volume, and all of its descendents. If only the `-l` option is used, then is allowed `locally` only for the specified *filesystem*. If only the `-d` option is used, then is allowed only for the descendent file systems.

Permissions are generally the ability to use a ZFS subcommand or change a ZFS property. The following permissions are available:

```
NAME            TYPE           NOTES
allow           subcommand     Must also have the permission that is
                               being allowed
clone           subcommand     Must also have the 'create' ability and
                               'mount' ability in the origin file system
create          subcommand     Must also have the 'mount' ability
destroy         subcommand     Must also have the 'mount' ability
hold            subcommand     Allows adding a user hold to a snapshot
mount           subcommand     Allows mount/umount of ZFS datasets
promote         subcommand     Must also have the 'mount' and 'promote'
                               ability in the origin file system
receive         subcommand     Must also have the 'mount' and 'create'
                               ability
release         subcommand     Allows releasing a user hold which
                               might destroy the snapshot
```

```
rename          subcommand     Must also have the 'mount' and 'create'
                               ability in the new parent
rollback        subcommand     Must also have the 'mount' ability
send            subcommand
share           subcommand     Allows sharing file systems over NFS or
                               SMB protocols
snapshot        subcommand     Must also have the 'mount' ability
groupquota      other          Allows accessing any groupquota@...
                               property
groupused       other          Allows reading any groupused@... property
userprop        other          Allows changing any user property
userquota       other          Allows accessing any userquota@...
                               property
userused        other          Allows reading any userused@... property
aclinherit      property
aclmode         property
atime           property
canmount        property
casesensitivity property
checksum        property
compression     property
copies          property
devices         property
exec            property
logbias         property
mountpoint      property
nbmand          property
normalization   property
primarycache    property
quota           property
readonly        property
recordsize      property
refquota        property
refreservation  property
reservation     property
secondarycache  property
setuid          property
shareiscsi      property
sharenfs        property
sharesmb        property
snapdir         property
utf8only        property
version         property
volblocksize    property
volsize         property
vscan           property
xattr           property
zoned           property
```

`zfs allow -c` *perm* | *@setname*[,...] *filesystem*|*volume*
Sets create time permissions. These permissions are granted (locally) to the creator of any newly-created descendent file system.

`zfs allow -s` *@setname perm*|*@setname*[,...] *filesystem*|*volume*
Defines or adds permissions to a permission set. The set can be used by other `zfs allow` commands for the specified *filesystem* and its descendents. Sets are evaluated dynamically, so changes to a set are immediately reflected. Permission sets follow the same naming restrictions as ZFS file systems, but the name must begin with an at sign (`@`), and can be no more than 64 characters long.

`zfs unallow` [`-rldug`] `everyone`|*user*|*group*[,...] [*perm*|*@setname*[, ...]] *filesystem*|*volume*
`zfs unallow` [`-rld`] `-e` [*perm*|*@setname* [,...]] *filesystem*|*volume*
`zfs unallow` [`-r`] `-c` [*perm*|*@setname*[,...]]
filesystem|*volume*
Removes permissions that were granted with the `zfs allow` command. No permissions are explicitly denied, so other permissions granted are still in effect. For example, if the permission is granted by an ancestor. If no permissions are specified, then all permissions for the specified *user*, *group*, or `everyone` are removed. Specifying `everyone` (or using the `-e` option) only removes the permissions that were granted to `everyone`, not all permissions for every user and group. See the `zfs allow` command for a description of the `-ldugec` options.

`-r`
Recursively remove the permissions from this file system and all descendents.

`zfs unallow` [`-r`] `-s` *@setname* [*perm*|*@setname*[,...]]
filesystem|*volume*
Removes permissions from a permission set. If no permissions are specified, then all permissions are removed, thus removing the set entirely.

`zfs hold` [`-r`] *tag snapshot*...
Adds a single reference, named with the *tag* argument, to the specified snapshot or snapshots. Each snapshot has its own tag namespace, and tags must be unique within that space.

If a hold exists on a snapshot, attempts to destroy that snapshot by using the `zfs destroy` command return `EBUSY`.

`-r`
Specifies that a hold with the given tag is applied recursively to the snapshots of all descendent file systems.

`zfs holds` [`-r`] *snapshot*...
Lists all existing user references for the given snapshot or snapshots.

`-r`
Lists the holds that are set on the named descendent snapshots, in addition to listing the holds on the named snapshot.

`zfs release` [`-r`] *tag snapshot...*

Removes a single reference, named with the *tag* argument, from the specified snapshot or snapshots. The tag must already exist for each snapshot.

If a hold exists on a snapshot, attempts to destroy that snapshot by using the `zfs destroy` command return `EBUSY`.

`-r`

Recursively releases a hold with the given tag on the snapshots of all descendent file systems.

Examples

EXAMPLE 1 Creating a ZFS File System Hierarchy

The following commands create a file system named `pool/home` and a file system named `pool/home/bob`. The mount point `/export/home` is set for the parent file system, and is automatically inherited by the child file system.

```
# zfs create pool/home
# zfs set mountpoint=/export/home pool/home
# zfs create pool/home/bob
```

EXAMPLE 2 Creating a ZFS Snapshot

The following command creates a snapshot named `yesterday`. This snapshot is mounted on demand in the `.zfs/snapshot` directory at the root of the `pool/home/bob` file system.

```
# zfs snapshot pool/home/bob@yesterday
```

EXAMPLE 3 Creating and Destroying Multiple Snapshots

The following command creates snapshots named `yesterday` of `pool/home` and all of its descendent file systems. Each snapshot is mounted on demand in the `.zfs/snapshot` directory at the root of its file system. The second command destroys the newly created snapshots.

```
# zfs snapshot -r pool/home@yesterday
# zfs destroy -r pool/home@yesterday
```

EXAMPLE 4 Disabling and Enabling File System Compression

The following command disables the `compression` property for all file systems under `pool/home`. The next command explicitly enables `compression` for `pool/home/anne`.

```
# zfs set compression=off pool/home
# zfs set compression=on pool/home/anne
```

EXAMPLE 5 Listing ZFS Datasets

The following command lists all active file systems and volumes in the system. Snapshots are displayed if the pool's `listsnaps` property is `on`. The default is `on`. See `zpool(1M)` for more information on pool properties.

EXAMPLE 5 Listing ZFS Datasets *(Continued)*

```
# zfs list
   NAME                    USED  AVAIL  REFER  MOUNTPOINT
   pool                    450K   457G    18K  /pool
   pool/home               315K   457G    21K  /export/home
   pool/home/anne           18K   457G    18K  /export/home/anne
   pool/home/bob           276K   457G   276K  /export/home/bob
```

EXAMPLE 6 Setting a Quota on a ZFS File System

The following command sets a quota of 50 Gbytes for `pool/home/bob`.

```
# zfs set quota=50G pool/home/bob
```

EXAMPLE 7 Listing ZFS Properties

The following command lists all properties for `pool/home/bob`.

```
# zfs get all pool/home/bob
NAME           PROPERTY              VALUE                  SOURCE
pool/home/bob  type                  filesystem             -
pool/home/bob  creation              Wed May  5 13:09 2010  -
pool/home/bob  used                  21K                    -
pool/home/bob  available             50.0G                  -
pool/home/bob  referenced            21K                    -
pool/home/bob  compressratio         1.00x                  -
pool/home/bob  mounted               yes                    -
pool/home/bob  quota                 50G                    local
pool/home/bob  reservation           none                   default
pool/home/bob  recordsize            128K                   default
pool/home/bob  mountpoint            /pool/home/bob         default
pool/home/bob  sharenfs              off                    default
pool/home/bob  checksum              on                     default
pool/home/bob  compression           off                    default
pool/home/bob  atime                 on                     default
pool/home/bob  devices               on                     default
pool/home/bob  exec                  on                     default
pool/home/bob  setuid                on                     default
pool/home/bob  readonly              off                    default
pool/home/bob  zoned                 off                    default
pool/home/bob  snapdir               hidden                 default
pool/home/bob  aclmode               groupmask              default
pool/home/bob  aclinherit            restricted             default
pool/home/bob  canmount              on                     default
pool/home/bob  shareiscsi            off                    default
pool/home/bob  xattr                 on                     default
pool/home/bob  copies                1                      default
pool/home/bob  version               4                      -
pool/home/bob  utf8only              off                    -
```

EXAMPLE 7 Listing ZFS Properties *(Continued)*

```
pool/home/bob  normalization         none                  -
pool/home/bob  casesensitivity       sensitive             -
pool/home/bob  vscan                 off                   default
pool/home/bob  nbmand                off                   default
pool/home/bob  sharesmb              off                   default
pool/home/bob  refquota              none                  default
pool/home/bob  refreservation        none                  default
pool/home/bob  primarycache          all                   default
pool/home/bob  secondarycache        all                   default
pool/home/bob  usedbysnapshots       0                     -
pool/home/bob  usedbydataset         21K                   -
pool/home/bob  usedbychildren        0                     -
pool/home/bob  usedbyrefreservation  0                     -
pool/home/bob  logbias               latency               default
```

The following command gets a single property value.

```
# zfs get -H -o value compression pool/home/bob
on
```

The following command lists all properties with local settings for `pool/home/bob`.

```
# zfs get -r -s local -o name,property,value all pool/home/bob
NAME           PROPERTY              VALUE
pool/home/bob  quota                 20G
pool/home/bob  compression           on
```

EXAMPLE 8 Rolling Back a ZFS File System

The following command reverts the contents of `pool/home/anne` to the snapshot named `yesterday`, deleting all intermediate snapshots.

```
# zfs rollback -r pool/home/anne@yesterday
```

EXAMPLE 9 Creating a ZFS Clone

The following command creates a writable file system whose initial contents are the same as `pool/home/bob@yesterday`.

```
# zfs clone pool/home/bob@yesterday pool/clone
```

EXAMPLE 10 Promoting a ZFS Clone

The following commands illustrate how to test out changes to a file system, and then replace the original file system with the changed one, using clones, clone promotion, and renaming:

```
# zfs create pool/project/production
  populate /pool/project/production with data
# zfs snapshot pool/project/production@today
```

EXAMPLE 10 Promoting a ZFS Clone *(Continued)*

```
# zfs clone pool/project/production@today pool/project/beta
make changes to /pool/project/beta and test them
# zfs promote pool/project/beta
# zfs rename pool/project/production pool/project/legacy
# zfs rename pool/project/beta pool/project/production
once the legacy version is no longer needed, it can be destroyed
# zfs destroy pool/project/legacy
```

EXAMPLE 11 Inheriting ZFS Properties

The following command causes `pool/home/bob` and `pool/home/anne` to inherit the `checksum` property from their parent.

```
# zfs inherit checksum pool/home/bob pool/home/anne
```

EXAMPLE 12 Remotely Replicating ZFS Data

The following commands send a full stream and then an incremental stream to a remote machine, restoring them into `poolB/received/fs@a`and `poolB/received/fs@b`, respectively. `poolB` must contain the file system `poolB/received`, and must not initially contain `poolB/received/fs`.

```
# zfs send pool/fs@a | \
   ssh host zfs receive poolB/received/fs@a
# zfs send -i a pool/fs@b | ssh host \
   zfs receive poolB/received/fs
```

EXAMPLE 13 Using the `zfs receive -d` Option

The following command sends a full stream of `poolA/fsA/fsB@snap` to a remote machine, receiving it into `poolB/received/fsA/fsB@snap`. The `fsA/fsB@snap` portion of the received snapshot's name is determined from the name of the sent snapshot. `poolB` must contain the file system `poolB/received`. If `poolB/received/fsA` does not exist, it is created as an empty file system.

```
# zfs send poolA/fsA/fsB@snap | \
   ssh host zfs receive -d poolB/received
```

EXAMPLE 14 Setting User Properties

The following example sets the user–defined `com.example:department` property for a dataset.

```
# zfs set com.example:department=12345 tank/accounting
```

EXAMPLE 15 Creating a ZFS Volume as an iSCSI Target Device

The following example shows how to create a ZFS volume as an iSCSI target.

```
# zfs create -V 2g pool/volumes/vol1
# zfs set shareiscsi=on pool/volumes/vol1
```

EXAMPLE 15 Creating a ZFS Volume as an iSCSI Target Device *(Continued)*

```
# iscsitadm list target
Target: pool/volumes/vol1
 iSCSI Name:
 iqn.1986-03.com.sun:02:7b4b02a6-3277-eb1b-e686-a24762c52a8c
 Connections: 0
```

After the iSCSI target is created, set up the iSCSI initiator. For more information about the Solaris iSCSI initiator, see `iscsitadm(1M)`.

EXAMPLE 16 Performing a Rolling Snapshot

The following example shows how to maintain a history of snapshots with a consistent naming scheme. To keep a week's worth of snapshots, the user destroys the oldest snapshot, renames the remaining snapshots, and then creates a new snapshot, as follows:

```
# zfs destroy -r pool/users@7daysago
# zfs rename -r pool/users@6daysago @7daysago
# zfs rename -r pool/users@5daysago @6daysago
# zfs rename -r pool/users@yesterday @5daysago
# zfs rename -r pool/users@yesterday @4daysago
# zfs rename -r pool/users@yesterday @3daysago
# zfs rename -r pool/users@yesterday @2daysago
# zfs rename -r pool/users@today @yesterday
# zfs snapshot -r pool/users@today
```

EXAMPLE 17 Setting `sharenfs` Property Options on a ZFS File System

The following commands show how to set `sharenfs` property options to enable `rw` access for a set of IP addresses and to enable root access for system `neo` on the `tank/home` file system.

```
# zfs set sharenfs='rw=@123.123.0.0/16,root=neo' tank/home
```

If you are using DNS for host name resolution, specify the fully qualified hostname.

EXAMPLE 18 Delegating ZFS Administration Permissions on a ZFS Dataset

The following example shows how to set permissions so that user `cindys` can create, destroy, mount, and take snapshots on `tank/cindys`. The permissions on `tank/cindys` are also displayed.

```
# zfs allow cindys create,destroy,mount,snapshot tank/cindys
# zfs allow tank/cindys
-------------------------------------------------------------
Local+Descendent permissions on (tank/cindys)
          user cindys create,destroy,mount,snapshot
-------------------------------------------------------------
```

Because the `tank/cindys` mount point permission is set to 755 by default, user `cindys` will be unable to mount file systems under `tank/cindys`. Set an ACL similar to the following syntax to provide mount point access:

```
# chmod A+user:cindys:add_subdirectory:allow /tank/cindys
```

EXAMPLE 19 Delegating Create Time Permissions on a ZFS Dataset

The following example shows how to grant anyone in the group `staff` to create file systems in `tank/users`. This syntax also allows staff members to destroy their own file systems, but not destroy anyone else's file system. The permissions on `tank/users` are also displayed.

```
# zfs allow staff create,mount tank/users
# zfs allow -c destroy tank/users
# zfs allow tank/users
-------------------------------------------------------------
Create time permissions on (tank/users)
          create,destroy
Local+Descendent permissions on (tank/users)
          group staff create,mount
-------------------------------------------------------------
```

EXAMPLE 20 Defining and Granting a Permission Set on a ZFS Dataset

The following example shows how to define and grant a permission set on the `tank/users` file system. The permissions on `tank/users` are also displayed.

```
# zfs allow -s @pset create,destroy,snapshot,mount tank/users
# zfs allow staff @pset tank/users
# zfs allow tank/users
-------------------------------------------------------------
Permission sets on (tank/users)
        @pset create,destroy,mount,snapshot
Create time permissions on (tank/users)
        create,destroy
Local+Descendent permissions on (tank/users)
        group staff @pset,create,mount
-------------------------------------------------------------
```

EXAMPLE 21 Delegating Property Permissions on a ZFS Dataset

The following example shows to grant the ability to set quotas and reservations on the `users/home` file system. The permissions on `users/home` are also displayed.

```
# zfs allow cindys quota,reservation users/home
# zfs allow users/home
-------------------------------------------------------------
Local+Descendent permissions on (users/home)
        user cindys quota,reservation
-------------------------------------------------------------
cindys% zfs set quota=10G users/home/marks
```

EXAMPLE 21 Delegating Property Permissions on a ZFS Dataset *(Continued)*

```
cindys% zfs get quota users/home/marks
NAME              PROPERTY  VALUE             SOURCE
users/home/marks  quota     10G               local
```

EXAMPLE 22 Removing ZFS Delegated Permissions on a ZFS Dataset

The following example shows how to remove the snapshot permission from the `staff` group on the `tank/users` file system. The permissions on `tank/users` are also displayed.

```
# zfs unallow staff snapshot tank/users
# zfs allow tank/users
-------------------------------------------------------------
Permission sets on (tank/users)
        @pset create,destroy,mount,snapshot
Create time permissions on (tank/users)
        create,destroy
Local+Descendent permissions on (tank/users)
        group staff @pset,create,mount
-------------------------------------------------------------
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`1`
An error occurred.

`2`
Invalid command line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Committed

See Also

`ssh(1)`, `iscsitadm(1M)`, `mount(1M)`, `share(1M)`, `unshare(1M)`, `zonecfg(1M)`, `zpool(1M)`, `chmod(2)`, `stat(2)`, `write(2)`, `fsync(3C)`, `dfstab(4)`, `attributes(5)`

See the `gzip(1)` man page, which is not part of the SunOS man page collection.

For information about using the ZFS web-based management tool and other ZFS features, see the *Oracle Solaris ZFS Administration Guide*.

Notes The ZFS deduplication feature is not available in the Oracle Solaris 10 release.

Name zic – time zone compiler

Synopsis

```
zic [--version] [-s] [-v] [-l localtime] [-p posixrules]
    [-d directory] [-y yearistype] [filename]...
```

Description `zic` reads text from the file(s) named on the command line and creates the time conversion information files specified in this input. If a *filename* is '–', the standard input is read.

Input lines are made up of fields. Fields are separated by any number of white space characters. Leading and trailing white space on input lines is ignored. A pound sign (#) indicates a comment that extends to the end of the line. White space characters and pound signs can be enclosed within double quotes (" ") if they are to be used as part of a field. Any line that is blank (after comment stripping) is ignored. Non-blank lines are expected to be of one of three types: rule lines, zone lines, or link lines.

Rule A rule line has the form:

For example:

```
Rule   NAME  FROM  TO  TYPE  IN   ON     AT   SAVE  LETTER/S
```

The fields that make up a rule line are:

```
Rule   USA   1969  1973   -  Apr lastSun 2:00  1:00   D
```

`NAME` Gives the (arbitrary) name of the set of rules this rule is part of.

`FROM` Gives the first year in which the rule applies. The word `minimum` (or an abbreviation) means the minimum year with a representable time value. The word `maximum` (or an abbreviation) means the maximum year with a representable time value.

`TO` Gives the final year in which the rule applies. In addition to `minimum` and `maximum` (as above), the word `only` (or an abbreviation) can be used to repeat the value of the `FROM` field.

`TYPE` Gives the type of year in which the rule applies. If `TYPE` is:

- '–' The rule applies in all years between `FROM` and `TO`, inclusive.
- `uspres` The rule applies in U.S. Presidential election years.
- `nonpres` The rule applies in years other than U.S. Presidential election years.
- `even` The rule applies to even-numbered years.
- `odd` The rule applies to odd-numbered years.

If `TYPE` is something else, then `zic` will attempt to execute the command

`yearistype` *year* `type`

to check the type of a year: an exit status of `0` means that the year is of the given type; an exit status of `1` means that the year is not of the given type. The `yearistype` command is not currently provided in the Solaris environment.

`IN` Names the month in the rule takes effect. Month names can be abbreviated.

`ON` Gives the day on which the rule takes effect. Recognized forms include:

`5`	the fifth day of the month
`lastSun`	The last Sunday in the month
`lastMon`	The last Monday in the month
`Sun>=8`	First Sunday on or after the eighth
`Sun<=25`	Last Sunday on or before the 25th

Names of days of the week can be abbreviated or spelled out in full. Note: There cannot be spaces within the `ON` field.

`AT` Gives the time of day at which the rule takes effect. Recognized forms include:

`2`	Time in hours
`2:00`	Time in hours and minutes
`15:00`	24-hour format time (for times after noon)
`1:28:14`	Time in hours, minutes, and seconds, where hour 0 is midnight at the start of the day and hour 24 is midnight at the end of the day.

Any of these forms can be followed by the letter `w` if the given time is local "wall clock" time; `s` if the given time is local "standard" time; or `u` (or `g` or `z`) if the given time is universal time. In the absence of an indicator, wall clock time is assumed.

`SAVE` Gives the amount of time to be added to local standard time when the rule is in effect. This field has the same format as the `AT` field (without the `w` and `s` suffixes).

`LETTER/S` Gives the "variable part" (for example, the "S" or "D" in "EST" or "EDT" of time zone abbreviations to be used when this rule is in effect. If this field is '–', the variable part is null.

`Zone` A zone line has the form:

```
Zone  NAME                GMTOFF  RULES/SAVE   FORMAT  [UNTIL]
```

For example:

```
Zone Australia/SouthWest   9:30        -         CST    1992 Mar 15 12:00
                           8:30      Aus         CST
```

The fields that make up a zone line are:

NAME — The name of the time zone. This is the name used in creating the time conversion information file for the zone.

GMTOFF — The amount of time to add to UTC to get standard time in this zone. This field has the same format as the AT and SAVE fields of rule lines; begin the field with a minus sign to subtract time from UTC.

RULES/SAVE — The name of the rule(s) that apply in the time zone or, alternately, an amount of time to add to local standard time. If this field is '–', then standard time always applies in the time zone.

FORMAT — The format for time zone abbreviations in this time zone. The pair of characters %s is used to show where the "variable part" of the time zone abbreviation goes. Alternately, a slash (/) separates standard and daylight abbreviations.

UNTIL — The time at which the UTC offset or the rule(s) change for a location. It is specified as a year, a month, a day, and a time of day. The time of day has the same format as the AT field of rule lines. If this is specified, the time zone information is generated from the given UTC offset and rule change until the time specified.

The month, day, and time of day have the same format as the IN, ON, and AT columns of a rule; trailing columns can be omitted, and default to the earliest possible value for the missing columns.

The next line must be a "continuation" line. This line has the same form as a zone line except that the string "Zone" and the name are omitted. The continuation line places information starting at the time specified as the UNTIL field in the previous line in the file used by the previous line. Continuation lines can contain an UNTIL field, just as zone lines do, indicating that the next line is a further continuation.

Link — A link line has the form:

```
Link   LINK-FROM   LINK-TO
```

For example:

```
Link   Europe/Istanbul Asia/Istanbul
```

The LINK-FROM field should appear as the NAME field in some zone line; the LINK-TO field is used as an alternate name for that zone.

Except for continuation lines, lines can appear in any order in the input.

Options

`--`*version* Outputs version information and exits.

`-d` *directory* Creates time conversion information files in the directory *directory* rather than in the standard directory `/usr/share/lib/zoneinfo`.

`-l` *localtime* Uses the given time zone as local time *localtime*. `zic` acts as if the file contained a link line of the form:

```
Link localtime localtime
```

`-p` *posixrules* Uses the rules of the given time zone *posixrules* when handling POSIX-format time zone environment variables. `zic` acts as if the input contained a link line of the form:

```
Link    posixrules posixrules
```

This option is not used by `ctime(3C)` and `mktime(3C)` in the Solaris environment.

`-s` Limits time values stored in output files to values that are the same whether they are taken to be signed or unsigned. You can use this option to generate SVVS-compatible files.

This option is obsolete and may be removed in a future release.

`-v` Complains if a year that appears in a data file is outside the range of years representable by system time values (`0:00:00 a.m. UTC, January 1, 1970`, to `3:14:07 a.m. UTC, January 19, 2038`). This option also complains if a time of 24:00 (which cannot be handled by pre-1998 versions of `zic`) appears in the input.

`-y` *yearistype* Uses the given command *yearistype* rather than `yearistype` when checking year types (see `Rules` under `DESCRIPTION`).

Operands

filename A file containing input lines that specify the time conversion information files to be created. If a *filename* is '–', the standard input is read.

Files

`/usr/share/lib/zoneinfo` Standard directory used for created files

`/usr/share/lib/zoneinfo/src` Directory containing source files

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
Interface Stability	Committed*

* The `-s` option is obsolete.

See Also `time(1)`, `zdump(1M)`, `ctime(3C)`, `mktime(3C)`, `attributes(5)`

Notes For areas with more than two types of local time, you might need to use local standard time in the `AT` field of the earliest transition time's rule to ensure that the earliest transition time recorded in the compiled file is correct.

If the current *timezone* file is edited and compiled using the "`zic`" command, the changes will only be reflected in any new processes that are running. The most accurate way to reflect the changes for the whole system would be a reboot.

Name zoneadm – administer zones

Synopsis

```
zoneadm -z zonename [-u uuid-match] subcommand
     [subcommand_options]

zoneadm [-R root] [-z zonename] [-u uuid-match] list
     [list_options]

zoneadm [-R root] -z zonename [-u uuid-match] mark incomplete
```

Description The `zoneadm` utility is used to administer system zones. A zone is an application container that is maintained by the operating system runtime.

Security Once a process has been placed in a zone other than zone `0`, the process or any of its children cannot change zones.

Options The following options are supported:

`-R` *root*
Specify an alternate root (boot environment). This option can only be used in conjunction with the "`list`" and "`mark`" subcommands.

`-u` *uuid-match*
Unique identifier for a zone, as assigned by `libuuid(3LIB)`. If this option is present and the argument is a non-empty string, then the zone matching the UUID is selected instead of the one named by the `-z` option, if such a zone is present.

`-z` *zonename*
String identifier for a zone.

Subcommands Subcommands which can result in destructive actions or loss of work have a `-F` flag to force the action. If input is from a terminal device, the user is prompted if such a command is given without the `-F` flag; otherwise, if such a command is given without the `-F` flag, the action is disallowed, with a diagnostic message written to standard error. If a zone installation or uninstallation is interrupted, the zone is left in the incomplete state. Use uninstall to reset such a zone back to the configured state.

The following subcommands are supported:

`attach` [`-u` | `-U`] [`-b` *patchid*]... [`-F`] [`-n` *path*] [*brand-specific options*]
The `attach` subcommand takes a zone that has been detached from one system and attaches the zone onto a new system. Therefore, it is advised (though not required) that the `detach` subcommand should be run before the "attach" takes place. Once you have the new zone in the configured state, use the `attach` subcommand to set up the zone root instead of installing the zone as a new zone.

For native zones, `zoneadm` checks package and patch levels on the machine to which the zone is to be attached. If the packages/patches that the zone depends on from the global zone are different (have different revision numbers) from the dependent packages/patches on the source machine, `zoneadm` reports these conflicts and does not perform the attach. If

the destination system has only newer dependent packages/patches (higher revision numbers) than those on the source system, you can use the `-u` or `-U` options. The `-u` option updates the minimum number of packages within the attached zone to match the higher-revision packages and patches that exist on the new system. The `-U` option updates all packages in the attached zone that are also installed in the global zone. With `-u` or `-U`, as in the default behavior, `zoneadm` does not perform an attach if outdated packages/patches are found on the target system.

For native zones, one or more `-b` options can be used to specify a patch ID for a patch installed in the zone. These patches will be backed out before the zone is attached or, if `-u` was also specified, updated.

The `-F` option can be used to force the zone into the "installed" state with no validation. This option should be used with care since it can leave the zone in an unsupportable state if it was moved from a source system to a target system that is unable to properly host the zone. The `-n` option can be used to perform a "dry run" of the `attach` subcommand. It uses the output of the "`detach -n`" subcommand as input and is useful to identify any conflicting issues, such as the network device being incompatible, and can also determine whether the host is capable of supporting the zone. The path can be "`-`", to read the input from standard input.

The zone's brand may include additional options that govern how the zone will be attached. See `brands(5)` for specific brand information.

The zone being attached must first be configured using the `zonecfg` (see `zonecfg(1M)`) command. This does not apply when running "`attach -n`".

Use the following command to attach a zone:

```
# zoneadm -z my-zone attach
```

Use the following command to attach and update a zone:

```
# zoneadm -z my-zone attach -u (or -U)
```

In the absence of `-n` (as above), the source zone must be halted before this subcommand can be used.

`-n` *path*
: Read the zone manifest and verify that the target machine has the correct configuration to host the zone without actually performing an attach. The zone on the target system does not have to be configured on the new host before doing a trial-run attach.

`-u` or `-U`
: Update the attached zone as described above.

`boot` [`--` *boot_options*]
: Boot (or activate) the specified zones.

The following *boot_options* are supported:

`-i` *altinit*
Select an alternative executable to be the primordial Process. *altinit* is a valid path to an executable. The default primordial process is `init(1M)`.

`-m` *smf_options*
The *smf_options* include two categories of options to control booting behavior of the service management facility: recovery options and messages options.

Message options determine the type and amount of messages that `smf(5)` displays during boot. Service options determine the services which are used to boot the system. See `kernel(1M)` for a listing of the `-m` suboptions.

`-s`
Boots only to milestone `svc:/milestone/single-user:default`. This milestone is equivalent to init level `s`. See `svc.startd(1M)` and `init(1M)`.

`clone` [`-m` *copy*] [`-s` *zfs_snapshot*] *source_zone*
Install a zone by copying an existing installed zone. This subcommand is an alternative way to install the zone.

`-m` *copy*
Force the clone to be a copy, even if a "ZFS clone" is possible.

`-s` *zfs_snapshot*
Specify the name of a ZFS snapshot to use as the source of the clone. The *snapshot* must be a *snapshot* of the source zone taken from a previous "`zoneadm` clone" installation.

The source zone must be halted before this subcommand can be used.

`detach` [`-n`]
Detach the specified zone. Detaching a zone is the first step in moving a zone from one system to another. The full procedure to migrate a zone is that the zone is detached, the *zonepath* directory is moved to the new host, and then the zone is attached on the new host. Once the zone is detached, it is left in the configured state. If you try to install or clone to a configured zone that has been detached, you will receive an error message and the `install` or `clone` subcommand will not be allowed to proceed. The `-n` option can be used to perform a "dry run" of the `detach` subcommand. This generates the information needed for running the "`attach -n`" subcommand, which is useful to identify any conflicting issues, such as the network device being incompatible or if the host is capable of supporting the zone. The information is sent to standard output and can be saved to a file or piped to the "`attach -n`" subcommand.

Use the following command to detach a zone:

```
# zoneadm -z my-zone detach
```

Unless the `-n` option is used, the source zone must be halted before this subcommand can be used.

`-n`
Generate a zone manifest on a running zone without actually detaching the zone. The state of the zone on the originating system is not changed. The zone manifest is sent to `stdout`. The global administrator can direct this output to a file or pipe it to a remote command to be immediately validated on the target host.

`halt`
Halt the specified zones. `halt` bypasses running the shutdown scripts inside the zone. It also removes run time resources of the zone.

Use:

```
zlogin zone shutdown
```

to cleanly shutdown the zone by running the shutdown scripts.

`help` [*subcommand*]
Display general help. If you specify *subcommand*, displays help on *subcommand*.

`install` [`-x` *nodataset*] [*brand-specific options*]
Install the specified zone on the system. This subcommand automatically attempts to verify first. It refuses to install if the verify step fails. See the `verify` subcommand.

The zone's brand may include additional options that govern how the software will be installed in the zone. See `brands(5)` for specific brand information.

`-x` *nodataset*
Do not create a ZFS file system.

`list` [*list_options*]
Display the name of the current zones, or the specified zone if indicated.

By default, all running zones are listed. If you use this subcommand with the `zoneadm -z` *zonename* option, it lists only the specified zone, regardless of its state. In this case, the `-i` and `-c` options are disallowed.

If neither the `-i` or `-c` options are given, all running zones are listed.

The following *list_options* are supported:

`-c`
Display all configured zones. This option overides the `-i` option.

`-i`
Expand the display to all installed zones.

`-p`
Request machine parsable output. The output format is a list of lines, one per zone, with colon- delimited fields. These fields are:

```
zoneid:zonename:state:zonepath:uuid:brand:ip-type
```

If the `zonepath` contains embedded colons, they can be escaped by a backslash ("\:"), which is parsable by using the shell `read(1)` function with the environmental variable IFS. The *uuid* value is assigned by `libuuid(3LIB)` when the zone is installed, and is useful for identifying the same zone when present (or renamed) on alternate boot environments. Any software that parses the output of the "`zoneadm list -p`" command must be able to handle any fields that may be added in the future.

The `-v` and `-p` options are mutually exclusive. If neither `-v` nor `-p` is used, just the zone name is listed.

`-v`

Display verbose information, including zone name, id, current state, root directory, brand type, ip-type, and options.

The `-v` and `-p` options are mutually exclusive. If neither `-v` nor `-p` is used, just the zone name is listed.

`mark incomplete`

Change the state of an installed zone to "incomplete." This command may be useful in cases where administrative changes on the system have rendered a zone unusable or inconsistent. This change cannot be undone (except by uninstalling the zone).

`move` *new_zonepath*

Move the *zonepath* to *new_zonepath*. The zone must be halted before this subcommand can be used. The *new_zonepath* must be a local file system and normal restrictions for *zonepath* apply.

`ready`

Prepares a zone for running applications but does not start any user processes in the zone.

`reboot`

Restart the zones. This is equivalent to a `halt boot` sequence. This subcommand fails if the specified zones are not active.

`uninstall [-F]`

Uninstall the specified zone from the system. Use this subcommand with caution. It removes all of the files under the *zonepath* of the zone in question. You can use the `-F` flag to force the action.

`verify`

Check to make sure the configuration of the specified zone can safely be installed on the machine. Following is a break-down of the checks by `resource/property` type:

`zonepath`

`zonepath` and its parent directory exist and are owned by root with appropriate modes . The appropriate modes are that `zonepath` is `700`, its parent is not `group` or `world-writable` and so forth. `zonepath` is not over an NFS mount. A sub-directory of the `zonepath` named "root" does not exist.

If `zonepath` does not exist, the `verify` does not fail, but merely warns that a subsequent install will attempt to create it with proper permissions. A `verify` subsequent to that might fail should anything go wrong.

`zonepath` cannot be a symbolic link.

`fs`
Any `fs` resources have their *type* value checked. An error is reported if the value is one of `proc`, `mntfs`, `autofs`, `cachefs`, or `nfs` or the filesystem does not have an associated mount binary at `/usr/lib/fs/`<*fstype*>`/mount`.

It is an error for the *directory* to be a relative path.

It is an error for the path specified by `raw` to be a relative path or if there is no `fsck` binary for a given filesystem type at `/usr/lib/fs/`<*fstype*>`/fsck`. It is also an error if a corresponding `fsck` binary exists but a `raw` path is not specified.

`net`
All physical network interfaces exist. All network address resources are one of:

- a valid IPv4 address, optionally followed by "/" and a prefix length;
- a valid IPv6 address, which must be followed by "/" and a prefix length;
- a host name which resolves to an IPv4 address.

Note that hostnames that resolve to IPv6 addresses are not supported.

The physical interface name is the network interface name.

A zone can be configured to be either exclusive-IP or shared-IP. For a shared-IP zone, both the physical and address properties must be set. For an exclusive-IP zone, the physical property must be set and the address property cannot be set.

`rctl`
It also verifies that any defined resource control values are valid on the current machine. This means that the privilege level is `privileged`, the limit is lower than the currently defined system value, and that the defined action agrees with the actions that are valid for the given resource control.

Examples

EXAMPLE 1 Using the `-m` Option

The following command illustrates the use of the `-m` option.

```
# zoneadm boot -- -m verbose
```

EXAMPLE 2 Using the `-i` Option

The following command illustrates the use of the `-i` option.

```
# zoneadm boot -- -i /sbin/init
```

EXAMPLE 3 Using the -s Option

The following command illustrates the use of the -s option.

```
# zoneadm boot -- -s
```

Exit Status

The following exit values are returned:

`0`
Successful completion.

`1`
An error occurred.

`2`
Invalid usage.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/zones
Interface Stability	Committed

See Also

`read(1)`, `svcs(1)`, `zlogin(1)`, `zonename(1)`, `init(1M)`, `kernel(1M)`, `svcadm(1M)`, `svc.startd(1M)`, `svc.startd(1M)`, `zonecfg(1M)`, `libuuid(3LIB)`, `attributes(5)`, `brands(5)`, `smf(5)`, `zones(5)`

Notes

The `zones(5)` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/zones:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

The act of installing a new non-global zone is a fresh installation of the Solaris operating system. A new installation of Solaris must not require interaction with the user (that is, it must be “hands off”). Because of this, packages installed in the global zone and all non-global zones cannot contain request scripts (see `pkgask(1M)`). If a package did have a request script, then the creation of a non-global zone could not be done without user intervention. Any package that contains a request script is added to the global zone only. See `pkgadd(1M)`.

Name zoneadmd – zone administration daemon

Synopsis

```
/usr/lib/zones/zoneadmd
```

Description `zoneadmd` is a system daemon that is started when the system needs to manage a particular zone. Because each instance of the `zoneadmd` daemon manages a particular zone, it is not unexpected to see multiple `zoneadmd` daemons running.

This daemon is started automatically by the zone management software and should not be invoked directly. The daemon shuts down automatically when no longer in use. It does not constitute a programming interface, but is classified as a private interface.

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWzoneu
Interface Stability	Private

See Also `svcs(1)`, `zlogin(1)`, `svcadm(1M)`, `zoneadm(1M)`, `attributes(5)`, `smf(5)`, `zones(5)`

Notes The `zones(5)` service is managed by the service management facility, `smf(5)`, under the service identifier:

```
svc:/system/zones:default
```

Administrative actions on this service, such as enabling, disabling, or requesting restart, can be performed using `svcadm(1M)`. The service's status can be queried using the `svcs(1)` command.

Name zonecfg – set up zone configuration

Synopsis

```
zonecfg -z zonename
zonecfg -z zonename subcommand
zonecfg -z zonename -f command_file
zonecfg help
```

Description The `zonecfg` utility creates and modifies the configuration of a zone. Zone configuration consists of a number of resources and properties.

To simplify the user interface, `zonecfg` uses the concept of a scope. The default scope is global.

The following synopsis of the `zonecfg` command is for interactive usage:

```
zonecfg -z zonename subcommand
```

Parameters changed through `zonecfg` do not affect a running zone. The zone must be rebooted for the changes to take effect.

In addition to creating and modifying a zone, the `zonecfg` utility can also be used to persistently specify the resource management settings for the global zone.

In the following text, "rctl" is used as an abbreviation for "resource control". See `resource_controls(5)`.

Every zone is configured with an associated brand. The brand determines the user-level environment used within the zone, as well as various behaviors for the zone when it is installed, boots, or is shutdown. Once a zone has been installed the brand cannot be changed. The default brand is determined by the installed distribution in the global zone. Some brands do not support all of the `zonecfg` properties and resources. See the brand-specific man page for more details on each brand. For an overview of brands, see the `brands(5)` man page.

Resources The following resource types are supported:

`attr`
Generic attribute.

`capped-cpu`
Limits for CPU usage.

`capped-memory`
Limits for physical, swap, and locked memory.

`dataset`
ZFS dataset.

`dedicated-cpu`
Subset of the system's processors dedicated to this zone while it is running.

`device`
Device.

`fs`
file-system

`inherit-pkg-dir`
Directory inherited from the global zone. Used for sparse root zones (see the discussion of "Sparse and Whole Root Non-Global Zones," below). Software packages whose contents have been transferred into that directory are inherited in read-only mode by the non-global zone and the non-global zone's packaging database is updated to reflect those packages. Such resources are not modifiable or removable once a zone has been installed with `zoneadm`.

`net`
Network interface.

`rctl`
Resource control.

Sparse and Whole Root Non-Global Zones

In the administration of zones, it is useful to distinguish between the global zone and non-global zones. Within non-global zones, there are two zone root file system models: sparse and whole root. The sparse root zone model optimizes the sharing of objects. The whole root zone model provides the maximum configurability. Note that not all brands support the sparse zone model.

Sparse Root Zones

Non-global zones that have `inherit-pkg-dir` resources are called sparse root zones.

The sparse root zone model optimizes the sharing of objects in the following ways:

- Only a subset of the packages installed in the global zone are installed directly into the non-global zone.
- Read-only loopback file systems, identified as `inherit-pkg-dir` resources, are used to gain access to other files.

In this model, all packages appear to be installed in the non-global zone. Packages that do not deliver content into read-only loopback mount file systems are fully installed. There is no need to install content delivered into read-only loopback mounted file systems since that content is inherited (and visible) from the global zone.

- As a general guideline, a zone requires about 100 megabytes of free disk space per zone when the global zone has been installed with all of the standard Solaris packages.
- By default, any additional packages installed in the global zone also populate the non-global zones. The amount of disk space required might be increased accordingly, depending on whether the additional packages deliver files that reside in the `inherit-pkg-dir` resource space.

An additional 40 megabytes of RAM per zone are suggested, but not required on a machine with sufficient swap space.

A sparse zone inherits the following directories:

```
/lib
/platform
/sbin
/usr
```

Although `zonecfg` allows you to remove one of these as an inherited directory, you should not do so. You should either follow the whole-root model or the sparse model; a subset of the sparse model is not tested and you might encounter unexpected problems.

Adding an additional `inherit-pkg-dir` directory, such as `/opt`, to a sparse root zone is acceptable.

Whole Root Zones

The whole root zone model provides the maximum configurability. All of the required and any selected optional Solaris packages are installed into the private file systems of the zone. The advantages of this model include the capability for global administrators to customize their zones file system layout. This would be done, for example, to add arbitrary unbundled or third-party packages.

The disk requirements for this model are determined by the disk space used by the packages currently installed in the global zone.

Note – If you create a sparse root zone that contains the following `inherit-pkg-dir` directories, you must remove these directories from the non-global zone's configuration before the zone is installed to have a whole root zone:

- `/lib`
- `/platform`
- `/sbin`
- `/usr`

Properties Each resource type has one or more properties. There are also some global properties, that is, properties of the configuration as a whole, rather than of some particular resource.

The following properties are supported:

(global)
`zonename`

(global)
`zonepath`

(global)
`autoboot`

(global)
bootargs

(global)
pool

(global)
limitpriv

(global)
brand

(global)
cpu-shares

(global)
hostid

(global)
max-lwps

(global)
max-msg-ids

(global)
max-sem-ids

(global)
max-shm-ids

(global)
max-shm-memory

(global)
scheduling-class

fs
dir, special, raw, type, options

inherit-pkg-dir
dir

net
address, physical, defrouter

device
match

rctl
name, value

attr
name, type, value

```
dataset
   name

dedicated-cpu
   ncpus, importance

capped-memory
   physical, swap, locked

capped-cpu
   ncpus
```

As for the property values which are paired with these names, they are either simple, complex, or lists. The type allowed is property-specific. Simple values are strings, optionally enclosed within quotation marks. Complex values have the syntax:

(*<name>*=*<value>*,*<name>*=*<value>*, . . .)

where each *<value>* is simple, and the *<name>* strings are unique within a given property. Lists have the syntax:

[*<value>*, . . .]

where each *<value>* is either simple or complex. A list of a single value (either simple or complex) is equivalent to specifying that value without the list syntax. That is, "foo" is equivalent to "[foo]". A list can be empty (denoted by "[]").

In interpreting property values, `zonecfg` accepts regular expressions as specified in `fnmatch(5)`. See `EXAMPLES`.

The property types are described as follows:

global: `zonename`
The name of the zone.

global: `zonepath`
Path to zone's file system.

global: `autoboot`
Boolean indicating that a zone should be booted automatically at system boot. Note that if the zones service is disabled, the zone will not autoboot, regardless of the setting of this property. You enable the zones service with a `svcadm` command, such as:

```
# svcadm enable svc:/system/zones:default
```

Replace `enable` with `disable` to disable the zones service. See `svcadm(1M)`.

global: `bootargs`
Arguments (options) to be passed to the zone bootup, unless options are supplied to the "`zoneadm boot`" command, in which case those take precedence. The valid arguments are described in `zoneadm(1M)`.

global: `pool`
Name of the resource pool that this zone must be bound to when booted. This property is incompatible with the `dedicated-cpu` resource.

global: `limitpriv`
The maximum set of privileges any process in this zone can obtain. The property should consist of a comma-separated privilege set specification as described in `priv_str_to_set(3C)`. Privileges can be excluded from the resulting set by preceding their names with a dash (-) or an exclamation point (!). The special privilege string "zone" is not supported in this context. If the special string "default" occurs as the first token in the property, it expands into a safe set of privileges that preserve the resource and security isolation described in `zones(5)`. A missing or empty property is equivalent to this same set of safe privileges.

The system administrator must take extreme care when configuring privileges for a zone. Some privileges cannot be excluded through this mechanism as they are required in order to boot a zone. In addition, there are certain privileges which cannot be given to a zone as doing so would allow processes inside a zone to unduly affect processes in other zones. `zoneadm(1M)` indicates when an invalid privilege has been added or removed from a zone's privilege set when an attempt is made to either "boot" or "ready" the zone.

See `privileges(5)` for a description of privileges. The command "`ppriv -l`" (see `ppriv(1)`) produces a list of all Solaris privileges. You can specify privileges as they are displayed by `ppriv`. In `privileges(5)`, privileges are listed in the form PRIV_*privilege_name*. For example, the privilege *sys_time*, as you would specify it in this property, is listed in `privileges(5)` as `PRIV_SYS_TIME`.

global: `brand`
The zone's brand type.

global: `ip-type`
A zone can either share the IP instance with the global zone, which is the default, or have its own exclusive instance of IP.

This property takes the values `shared` and `exclusive`.

global: `hostid`
A zone can emulate a 32-bit host identifier to ease system consolidation. A zone's `hostid` property is empty by default, meaning that the zone does not emulate a host identifier. Zone host identifiers must be hexadecimal values between 0 and FFFFFFFE. A `0x` or `0X` prefix is optional. Both uppercase and lowercase hexadecimal digits are acceptable.

`fs`: dir, special, raw, type, options
Values needed to determine how, where, and so forth to mount file systems. See `mount(1M)`, `mount(2)`, `fsck(1M)`, and `vfstab(4)`.

`inherit-pkg-dir`: dir
The directory path.

`net`: address, physical, defrouter
The network address and physical interface name of the network interface. The network address is one of:

- a valid IPv4 address, optionally followed by "`/`" and a prefix length;
- a valid IPv6 address, which must be followed by "`/`" and a prefix length;
- a host name which resolves to an IPv4 address.

Note that host names that resolve to IPv6 addresses are not supported.

The physical interface name is the network interface name.

The value for the optional default router is specified similarly to the network address except that it must not be followed by a `/` (slash) and a network prefix length. To enable correct use of the `defrouter` functionality, the zones that use the property must be on a different subnet from the subnet on which the global zone resides. Also, each zone (or set of zones) that uses a different `defrouter` setting must be on a different subnet.

A zone can be configured to be either exclusive-IP or shared-IP. For a shared-IP zone, you must set both the physical and address properties; setting the default router is optional. The interface specified in the physical property must be plumbed in the global zone prior to booting the non-global zone. However, if the interface is not used by the global zone, it should be configured `down` in the global zone, and the default router for the interface should be specified here.

For an exclusive-IP zone, the physical property must be set and the address and default router properties cannot be set. Note that a single datalink cannot be shared among multiple exclusive-IP zones.

`device`: match
Device name to match.

`rctl`: name, value
The name and *priv*/*limit*/*action* triple of a resource control. See `prctl(1)` and `rctladm(1M)`. The preferred way to set rctl values is to use the global property name associated with a specific rctl.

`attr`: name, type, value
The name, type and value of a generic attribute. The `type` must be one of `int`, `uint`, `boolean` or `string`, and the value must be of that type. `uint` means unsigned , that is, a non-negative integer.

`dataset`: name
The name of a ZFS dataset to be accessed from within the zone. See `zfs(1M)`.

global: `cpu-shares`
The number of Fair Share Scheduler (FSS) shares to allocate to this zone. This property is incompatible with the `dedicated-cpu` resource. This property is the preferred way to set the `zone.cpu-shares` rctl.

global: `max-lwps`
The maximum number of LWPs simultaneously available to this zone. This property is the preferred way to set the `zone.max-lwps` rctl.

global: `max-msg-ids`
The maximum number of message queue IDs allowed for this zone. This property is the preferred way to set the `zone.max-msg-ids` rctl.

global: `max-sem-ids`
The maximum number of semaphore IDs allowed for this zone. This property is the preferred way to set the `zone.max-sem-ids` rctl.

global: `max-shm-ids`
The maximum number of shared memory IDs allowed for this zone. This property is the preferred way to set the `zone.max-shm-ids` rctl.

global: `max-shm-memory`
The maximum amount of shared memory allowed for this zone. This property is the preferred way to set the `zone.max-shm-memory` rctl. A scale (K, M, G, T) can be applied to the value for this number (for example, 1M is one megabyte).

global: `scheduling-class`
Specifies the scheduling class used for processes running in a zone. When this property is not specified, the scheduling class is established as follows:

- If the `cpu-shares` property or equivalent rctl is set, the scheduling class FSS is used.
- If neither `cpu-shares` nor the equivalent rctl is set and the zone's pool property references a pool that has a default scheduling class, that class is used.
- Under any other conditions, the system default scheduling class is used.

`dedicated-cpu`: ncpus, importance
The number of CPUs that should be assigned for this zone's exclusive use. The zone will create a pool and processor set when it boots. See `pooladm(1M)` and `poolcfg(1M)` for more information on resource pools. The `ncpu` property can specify a single value or a range (for example, 1-4) of processors. The `importance` property is optional; if set, it will specify the `pset.importance` value for use by `poold(1M)`. If this resource is used, there must be enough free processors to allocate to this zone when it boots or the zone will not boot. The processors assigned to this zone will not be available for the use of the global zone or other zones. This resource is incompatible with both the `pool` and `cpu-shares` properties. Only a single instance of this resource can be added to the zone.

`capped-memory`: physical, swap, locked
The caps on the memory that can be used by this zone. A scale (K, M, G, T) can be applied to the value for each of these numbers (for example, 1M is one megabyte). Each of these properties is optional but at least one property must be set when adding this resource. Only a single instance of this resource can be added to the zone. The `physical` property sets the `max-rss` for this zone. This will be enforced by `rcapd(1M)` running in the global zone. The

swap property is the preferred way to set the zone.max-swap rctl. The locked property is the preferred way to set the zone.max-locked-memory rctl.

capped-cpu: ncpus
Sets a limit on the amount of CPU time that can be used by a zone. The unit used translates to the percentage of a single CPU that can be used by all user threads in a zone, expressed as a fraction (for example, .75) or a mixed number (whole number and fraction, for example, 1.25). An ncpu value of 1 means 100% of a CPU, a value of 1.25 means 125%, .75 mean 75%, and so forth. When projects within a capped zone have their own caps, the minimum value takes precedence.

The capped-cpu property is an alias for zone.cpu-cap resource control and is related to the zone.cpu-cap resource control. See resource_controls(5).

The following table summarizes resources, property-names, and types:

```
resource          property-name   type
(global)          zonename        simple
(global)          zonepath        simple
(global)          autoboot        simple
(global)          bootargs        simple
(global)          pool            simple
(global)          limitpriv       simple
(global)          brand           simple
(global)          ip-type         simple
(global)          hostid          simple
(global)          cpu-shares      simple
(global)          max-lwps        simple
(global)          max-msg-ids     simple
(global)          max-sem-ids     simple
(global)          max-shm-ids     simple
(global)          max-shm-memory  simple
(global)          scheduling-class simple
fs                dir             simple
                   special         simple
                   raw             simple
                   type            simple
                   options         list of simple
inherit-pkg-dir   dir             simple
net               address         simple
                   physical        simple
device            match           simple
rctl              name            simple
                   value           list of complex
attr              name            simple
                   type            simple
                   value           simple
dataset           name            simple
dedicated-cpu     ncpus           simple or range
```

```
                    importance      simple

capped-memory      physical        simple with scale
                    swap            simple with scale
                    locked          simple with scale

capped-cpu         ncpus           simple
```

To further specify things, the breakdown of the complex property "value" of the "rctl" resource type, it consists of three name/value pairs, the names being "priv", "limit" and "action", each of which takes a simple value. The "name" property of an "attr" resource is syntactically restricted in a fashion similar but not identical to zone names: it must begin with an alphanumeric, and can contain alphanumerics plus the hyphen (-), underscore (_), and dot (.) characters. Attribute names beginning with "zone" are reserved for use by the system. Finally, the "autoboot" global property must have a value of "true" or "false".

Using Kernel Statistics to Monitor CPU Caps

Using the kernel statistics (kstat(3KSTAT)) module caps, the system maintains information for all capped projects and zones. You can access this information by reading kernel statistics (kstat(3KSTAT)), specifying caps as the kstat module name. The following command displays kernel statistics for all active CPU caps:

```
# kstat caps::'/cpucaps/'
```

A kstat(1M) command running in a zone displays only CPU caps relevant for that zone and for projects in that zone. See EXAMPLES.

The following are cap-related arguments for use with kstat(1M):

caps
: The kstat module.

project_caps or zone_caps
: kstat class, for use with the kstat -c option.

cpucaps_project_*id* or cpucaps_zone_*id*
: kstat name, for use with the kstat -n option. *id* is the project or zone identifier.

The following fields are displayed in response to a kstat(1M) command requesting statistics for all CPU caps.

module
: In this usage of kstat, this field will have the value caps.

name
: As described above, cpucaps_project_*id* or cpucaps_zone_*id*

above_sec
: Total time, in seconds, spent above the cap.

below_sec
: Total time, in seconds, spent below the cap.

`maxusage`
Maximum observed CPU usage.

`nwait`
Number of threads on cap wait queue.

`usage`
Current aggregated CPU usage for all threads belonging to a capped project or zone, in terms of a percentage of a single CPU.

`value`
The cap value, in terms of a percentage of a single CPU.

`zonename`
Name of the zone for which statistics are displayed.

See `EXAMPLES` for sample output from a `kstat` command.

Options The following options are supported:

`-f` *command_file*
Specify the name of `zonecfg` command file. *command_file* is a text file of `zonecfg` subcommands, one per line.

`-z` *zonename*
Specify the name of a zone. Zone names are case sensitive. Zone names must begin with an alphanumeric character and can contain alphanumeric characters, the underscore (`_`) the hyphen (`-`), and the dot (`.`). The name `global` and all names beginning with `SUNW` are reserved and cannot be used.

Subcommands You can use the `add` and `select` subcommands to select a specific resource, at which point the scope changes to that resource. The `end` and `cancel` subcommands are used to complete the resource specification, at which time the scope is reverted back to global. Certain subcommands, such as `add`, `remove` and `set`, have different semantics in each scope.

`zonecfg` supports a semicolon-separated list of subcommands. For example:

```
# zonecfg -z myzone "add net; set physical=myvnic; end"
```

Subcommands which can result in destructive actions or loss of work have an `-F` option to force the action. If input is from a terminal device, the user is prompted when appropriate if such a command is given without the `-F` option otherwise, if such a command is given without the `-F` option, the action is disallowed, with a diagnostic message written to standard error.

The following subcommands are supported:

`add` *resource-type* (global scope)
`add` *property-name property-value* (resource scope)
In the global scope, begin the specification for a given resource type. The scope is changed to that resource type.

In the resource scope, add a property of the given name with the given value. The syntax for property values varies with different property types. In general, it is a simple value or a list of simple values enclosed in square brackets, separated by commas (`[foo,bar,baz]`). See `PROPERTIES`.

`cancel`
End the resource specification and reset scope to global. Abandons any partially specified resources. `cancel` is only applicable in the resource scope.

`clear` *property-name*
Clear the value for the property.

`commit`
Commit the current configuration from memory to stable storage. The configuration must be committed to be used by `zoneadm`. Until the in-memory configuration is committed, you can remove changes with the `revert` subcommand. The `commit` operation is attempted automatically upon completion of a `zonecfg` session. Since a configuration must be correct to be committed, this operation automatically does a verify.

`create [-F] [ -a` *path* `|-b | -t` *template*`]`
Create an in-memory configuration for the specified zone. Use `create` to begin to configure a new zone. See `commit` for saving this to stable storage.

If you are overwriting an existing configuration, specify the `-F` option to force the action. Specify the `-t` *template* option to create a configuration identical to *template*, where *template* is the name of a configured zone.

Use the `-a` *path* option to facilitate configuring a detached zone on a new host. The *path* parameter is the zonepath location of a detached zone that has been moved on to this new host. Once the detached zone is configured, it should be installed using the "`zoneadm attach`" command (see `zoneadm(1M)`). All validation of the new zone happens during the `attach` process, not during zone configuration.

Use the `-b` option to create a blank configuration. Without arguments, `create` applies the Sun default settings.

`delete [-F]`
Delete the specified configuration from memory and stable storage. This action is instantaneous, no commit is necessary. A deleted configuration cannot be reverted.

Specify the `-F` option to force the action.

`end`
End the resource specification. This subcommand is only applicable in the resource scope. `zonecfg` checks to make sure the current resource is completely specified. If so, it is added to the in-memory configuration (see `commit` for saving this to stable storage) and the scope reverts to global. If the specification is incomplete, it issues an appropriate error message.

`export [-f` *output-file*`]`
Print configuration to standard output. Use the `-f` option to print the configuration to *output-file*. This option produces output in a form suitable for use in a command file.

`help [usage] [`*subcommand*`] [syntax] [`*command-name*`]`
Print general help or help about given topic.

`info zonename | zonepath | autoboot | brand | pool | limitpriv`
`info [`*resource-type* `[`*property-name*`=`*property-value*`]*]`
Display information about the current configuration. If *resource-type* is specified, displays only information about resources of the relevant type. If any *property-name* value pairs are specified, displays only information about resources meeting the given criteria. In the resource scope, any arguments are ignored, and `info` displays information about the resource which is currently being added or modified.

`remove` *resource-type*{*property-name*=*property-value*}(global scope)
In the global scope, removes the specified resource. The `[]` syntax means 0 or more of whatever is inside the square braces. If you want only to remove a single instance of the resource, you must specify enough property name-value pairs for the resource to be uniquely identified. If no property name-value pairs are specified, all instances will be removed. If there is more than one pair is specified, a confirmation is required, unless you use the `-F` option.

`select` *resource-type* {*property-name*=*property-value*}
Select the resource of the given type which matches the given *property-name property-value* pair criteria, for modification. This subcommand is applicable only in the global scope. The scope is changed to that resource type. The `{}` syntax means 1 or more of whatever is inside the curly braces. You must specify enough *property -name property-value* pairs for the resource to be uniquely identified.

`set` *property-name*=*property- value*
Set a given property name to the given value. Some properties (for example, `zonename` and `zonepath`) are global while others are resource-specific. This subcommand is applicable in both the global and resource scopes.

`verify`
Verify the current configuration for correctness:

- All resources have all of their required properties specified.
- A `zonepath` is specified.

`revert [-F]`
Revert the configuration back to the last committed state. The `-F` option can be used to force the action.

`exit [-F]`
Exit the `zonecfg` session. A commit is automatically attempted if needed. You can also use an EOF character to exit `zonecfg`. The `-F` option can be used to force the action.

Examples

EXAMPLE 1 Creating the Environment for a New Zone

In the following example, `zonecfg` creates the environment for a new zone. `/usr/local` is loopback mounted from the global zone into `/opt/local`. `/opt/sfw` is loopback mounted from the global zone, three logical network interfaces are added, and a limit on the number of fair-share scheduler (FSS) CPU shares for a zone is set using the `rctl` resource type. The example also shows how to select a given resource for modification.

```
example# zonecfg -z myzone3
my-zone3: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:myzone3> create
zonecfg:myzone3> set zonepath=/export/home/my-zone3
zonecfg:myzone3> set autoboot=true
zonecfg:myzone3> add fs
zonecfg:myzone3:fs> set dir=/usr/local
zonecfg:myzone3:fs> set special=/opt/local
zonecfg:myzone3:fs> set type=lofs
zonecfg:myzone3:fs> add options [ro,nodevices]
zonecfg:myzone3:fs> end
zonecfg:myzone3> add fs
zonecfg:myzone3:fs> set dir=/mnt
zonecfg:myzone3:fs> set special=/dev/dsk/c0t0d0s7
zonecfg:myzone3:fs> set raw=/dev/rdsk/c0t0d0s7
zonecfg:myzone3:fs> set type=ufs
zonecfg:myzone3:fs> end
zonecfg:myzone3> add inherit-pkg-dir
zonecfg:myzone3:inherit-pkg-dir> set dir=/opt/sfw
zonecfg:myzone3:inherit-pkg-dir> end
zonecfg:myzone3> add net
zonecfg:myzone3:net> set address=192.168.0.1/24
zonecfg:myzone3:net> set physical=eri0
zonecfg:myzone3:net> end
zonecfg:myzone3> add net
zonecfg:myzone3:net> set address=192.168.1.2/24
zonecfg:myzone3:net> set physical=eri0
zonecfg:myzone3:net> end
zonecfg:myzone3> add net
zonecfg:myzone3:net> set address=192.168.2.3/24
zonecfg:myzone3:net> set physical=eri0
zonecfg:myzone3:net> end
zonecfg:my-zone3> set cpu-shares=5
zonecfg:my-zone3> add capped-memory
zonecfg:my-zone3:capped-memory> set physical=50m
zonecfg:my-zone3:capped-memory> set swap=100m
zonecfg:my-zone3:capped-memory> end
zonecfg:myzone3> exit
```

EXAMPLE 2 Creating a Non-Native Zone

The following example creates a new Linux zone:

```
example# zonecfg -z lxzone
lxzone: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:lxzone> create -t SUNWlx
zonecfg:lxzone> set zonepath=/export/zones/lxzone
zonecfg:lxzone> set autoboot=true
zonecfg:lxzone> exit
```

EXAMPLE 3 Creating an Exclusive-IP Zone

The following example creates a zone that is granted exclusive access to `bge1` and `bge33000` and that is isolated at the IP layer from the other zones configured on the system.

The IP addresses and routing is configured inside the new zone using `sysidtool(1M)`.

```
example# zonecfg -z excl
excl: No such zone configured
Use 'create' to begin configuring a new zone
zonecfg:excl> create
zonecfg:excl> set zonepath=/export/zones/excl
zonecfg:excl> set ip-type=exclusive
zonecfg:excl> add net
zonecfg:excl:net> set physical=bge1
zonecfg:excl:net> end
zonecfg:excl> add net
zonecfg:excl:net> set physical=bge33000
zonecfg:excl:net> end
zonecfg:excl> exit
```

EXAMPLE 4 Associating a Zone with a Resource Pool

The following example shows how to associate an existing zone with an existing resource pool:

```
example# zonecfg -z myzone
zonecfg:myzone> set pool=mypool
zonecfg:myzone> exit
```

For more information about resource pools, see `pooladm(1M)` and `poolcfg(1M)`.

EXAMPLE 5 Changing the Name of a Zone

The following example shows how to change the name of an existing zone:

```
example# zonecfg -z myzone
zonecfg:myzone> set zonename=myzone2
zonecfg:myzone2> exit
```

EXAMPLE 6 Changing the Privilege Set of a Zone

The following example shows how to change the set of privileges an existing zone's processes will be limited to the next time the zone is booted. In this particular case, the privilege set will be the standard safe set of privileges a zone normally has along with the privilege to change the system date and time:

```
example# zonecfg -z myzone
zonecfg:myzone> set limitpriv="default,sys_time"
zonecfg:myzone2> exit
```

EXAMPLE 7 Setting the `zone.cpu-shares` Property for the Global Zone

The following command sets the `zone.cpu-shares` property for the global zone:

```
example# zonecfg -z global
zonecfg:global> set cpu-shares=5
zonecfg:global> exit
```

EXAMPLE 8 Using Pattern Matching

The following commands illustrate `zonecfg` support for pattern matching. In the zone `flexlm`, enter:

```
zonecfg:flexlm> add device
zonecfg:flexlm:device> set match="/dev/cua/a00[2-5]"
zonecfg:flexlm:device> end
```

In the global zone, enter:

```
global# ls /dev/cua
a     a000  a001  a002  a003  a004  a005  a006  a007  b
```

In the zone `flexlm`, enter:

```
flexlm# ls /dev/cua
a002  a003  a004  a005
```

EXAMPLE 9 Setting a Cap for a Zone to Three CPUs

The following sequence uses the `zonecfg` command to set the CPU cap for a zone to three CPUs.

```
zonecfg:myzone> add capped-cpu
zonecfg:myzone>capped-cpu> set ncpus=3
zonecfg:myzone>capped-cpu>capped-cpu> end
```

The preceding sequence, which uses the capped-cpu property, is equivalent to the following sequence, which makes use of the `zone.cpu-cap` resource control.

```
zonecfg:myzone> add rctl
zonecfg:myzone:rctl> set name=zone.cpu-cap
zonecfg:myzone:rctl> add value (priv=privileged,limit=300,action=none)
```

EXAMPLE 9 Setting a Cap for a Zone to Three CPUs *(Continued)*

```
zonecfg:myzone:rctl> end
```

EXAMPLE 10 Using `kstat` to Monitor CPU Caps

The following command displays information about all CPU caps.

```
# kstat -n /cpucaps/
module: caps                            instance: 0
name:   cpucaps_project_0               class:    project_caps
        above_sec                       0
        below_sec                       2157
        crtime                          821.048183159
        maxusage                        2
        nwait                           0
        snaptime                        235885.637253027
        usage                           0
        value                           18446743151372347932
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_1               class:    project_caps
        above_sec                       0
        below_sec                       0
        crtime                          225339.192787265
        maxusage                        5
        nwait                           0
        snaptime                        235885.637591677
        usage                           5
        value                           18446743151372347932
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_201             class:    project_caps
        above_sec                       0
        below_sec                       235105
        crtime                          780.37961782
        maxusage                        100
        nwait                           0
        snaptime                        235885.637789687
        usage                           43
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_202             class:    project_caps
        above_sec                       0
        below_sec                       235094
```

EXAMPLE 10 Using `kstat` to Monitor CPU Caps *(Continued)*

```
        crtime                          791.72983782
        maxusage                        100
        nwait                           0
        snaptime                        235885.637967512
        usage                           48
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_203             class:    project_caps
        above_sec                       0
        below_sec                       235034
        crtime                          852.104401481
        maxusage                        75
        nwait                           0
        snaptime                        235885.638144304
        usage                           47
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_project_86710           class:    project_caps
        above_sec                       22
        below_sec                       235166
        crtime                          698.441717859
        maxusage                        101
        nwait                           0
        snaptime                        235885.638319871
        usage                           54
        value                           100
        zonename                        global

module: caps                            instance: 0
name:   cpucaps_zone_0                  class:    zone_caps
        above_sec                       100733
        below_sec                       134332
        crtime                          821.048177123
        maxusage                        207
        nwait                           2
        snaptime                        235885.638497731
        usage                           199
        value                           200
        zonename                        global

module: caps                            instance: 1
name:   cpucaps_project_0               class:    project_caps
```

EXAMPLE 10 Using `kstat` to Monitor CPU Caps *(Continued)*

```
        above_sec                       0
        below_sec                       0
        crtime                          225360.256448422
        maxusage                        7
        nwait                           0
        snaptime                        235885.638714404
        usage                           7
        value                           18446743151372347932
        zonename                        test_001

module: caps                            instance: 1
name:   cpucaps_zone_1                  class:    zone_caps
        above_sec                       2
        below_sec                       10524
        crtime                          225360.256440278
        maxusage                        106
        nwait                           0
        snaptime                        235885.638896443
        usage                           7
        value                           100
        zonename                        test_001
```

EXAMPLE 11 Displaying CPU Caps for a Specific Zone or Project

Using the `kstat -c` and `-i` options, you can display CPU caps for a specific zone or project, as below. The first command produces a display for a specific project, the second for the same project within zone 1.

```
# kstat -c project_caps

# kstat -c project_caps -i 1
```

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`1`
: An error occurred.

`2`
: Invalid usage.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/zones
Interface Stability	Volatile

See Also ppriv(1), prctl(1), zlogin(1), kstat(1M), mount(1M), pooladm(1M), poolcfg(1M), poold(1M), rcapd(1M), rctladm(1M), svcadm(1M), sysidtool(1M), zfs(1M), zoneadm(1M), priv_str_to_set(3C), kstat(3KSTAT), vfstab(4), attributes(5), brands(5), fnmatch(5), lx(5), privileges(5), resource_controls(5), zones(5)

System Administration Guide: Solaris Containers-Resource Management and Solaris Zones

Notes All character data used by `zonecfg` must be in US-ASCII encoding.

Name zpool – configures ZFS storage pools

Synopsis

```
zpool [-?]

zpool add [-fn] pool vdev ...

zpool attach [-f] pool device new_device

zpool clear [-F [-n]] pool [device]

zpool create [-fn] [-o property=value] ... [-O file-system-property=value]
     ... [-m mountpoint] [-R root] pool vdev ...

zpool destroy [-f] pool

zpool detach pool device

zpool export [-f] pool ...

zpool get all | property[,...] pool ...

zpool history [-il] [pool] ...

zpool import [-d dir] [-D]

zpool import [-o mntopts] [-o property=value] ... [-d dir | -c cachefile]
     [-D] [-f] [-R root] [-F [-n]] -a

zpool import [-o mntopts] [-o property=value] ... [-d dir | -c cachefile]
     [-D] [-f] [-R root] [-F [-n]] pool |id [newpool]

zpool iostat [-v] [pool] ... [interval[count]]

zpool list [-H] [-o property[,...]] [pool] ...

zpool offline [-t] pool device ...

zpool online pool device ...

zpool remove pool device ...

zpool replace [-f] pool device [new_device]

zpool scrub [-s] pool ...

zpool set property=value pool

zpool split [-R altroot] [-n] [-o mntopts] [-o property=value] pool
     newpool [device ...]

zpool status [-xv] [pool] ...

zpool upgrade

zpool upgrade -v

zpool upgrade [-V version] -a | pool ...
```

Description The `zpool` command configures ZFS storage pools. A storage pool is a collection of devices that provides physical storage and data replication for ZFS datasets.

All datasets within a storage pool share the same space. See `zfs(1M)` for information on managing datasets.

Virtual Devices (`vdevs`) A *virtual device* describes a single device or a collection of devices organized according to certain performance and fault characteristics. The following virtual devices are supported:

`disk`
A block device, typically located under `/dev/dsk`. ZFS can use individual slices or partitions, though the recommended mode of operation is to use whole disks. A disk can be specified by a full path, or it can be a shorthand name (the relative portion of the path under `/dev/dsk`). A whole disk can be specified by omitting the slice or partition designation. When given a whole disk, ZFS automatically labels the disk, if necessary.

`file`
A regular file. The use of files as a backing store is strongly discouraged. It is designed primarily for experimental purposes, as the fault tolerance of a file is only as good as the file system of which it is a part. A file must be specified by a full path.

`mirror`
A mirror of two or more devices. Data is replicated in an identical fashion across all components of a mirror. A mirror with *N* disks of size *X* can hold *X* bytes and can withstand (*N-1*) devices failing before data integrity is compromised.

`raidz`
`raidz1`
`raidz2`
`raidz3`
A variation on RAID-5 that allows for better distribution of parity and eliminates the "RAID-5 write hole" (in which data and parity become inconsistent after a power loss). Data and parity is striped across all disks within a raidz group.

A raidz group can have single-, double- , or triple parity, meaning that the raidz group can sustain one, two, or three failures, respectively, without losing any data. The raidz1 `vdev` type specifies a single-parity raidz group; the raidz2 `vdev` type specifies a double-parity raidz group; and the raidz3 `vdev` type specifies a triple-parity raidz group. The raidz `vdev` type is an alias for raidz1.

A raidz group with *N* disks of size *X* with *P* parity disks can hold approximately (*N-P*)***X* bytes and can withstand *P* device(s) failing before data integrity is compromised. The minimum number of devices in a raidz group is one more than the number of parity disks. The recommended number of devices is between 3 and 9, to help increase performance.

`spare`
A special pseudo-`vdev` that identifies an available hot spare for a pool. For more information, see the "Hot Spares" section.

`log`
A separate-intent log device. If more than one log device is specified, then writes are load-balanced between devices. Log devices can be mirrored. However, raidz `vdev` types are not supported for the intent log. For more information, see the "Intent Log" section.

`cache`
A device used to cache storage pool data. A cache device cannot be configured as a mirror or `raidz` group. For more information, see the "Cache Devices" section.

Virtual devices cannot be nested, so a mirror or raidz virtual device can only contain files or disks. Mirrors of mirrors (or other combinations) are not allowed.

A pool can have any number of virtual devices at the top of the configuration (known as *top-level vdevs*). Data is dynamically distributed across all top-level devices to balance data among devices. As new virtual devices are added, ZFS automatically places data on the newly available devices.

Virtual devices are specified one at a time on the command line, separated by whitespace. The keywords `mirror` and `raidz` are used to distinguish where a group ends and another begins. For example, the following creates two top-level vdevs, each a mirror of two disks:

```
# zpool create mypool mirror c0t0d0 c0t1d0 mirror c1t0d0 c1t1d0
```

Device Failure and Recovery

ZFS supports a rich set of mechanisms for handling device failure and data corruption. All metadata and data is checksummed, and ZFS automatically repairs bad data from a good copy when corruption is detected.

In order to take advantage of these features, a pool must make use of some form of redundancy, using either mirrored or raidz groups. While ZFS supports running in a non-redundant configuration, where each top-level vdev is simply a disk or file, this is strongly discouraged. A single case of bit corruption can render some or all of your data unavailable.

A pool's health status is described by one of three states: online, degraded, or faulted. An online pool has all devices operating normally. A degraded pool is one in which one or more devices have failed, but the data is still available due to a redundant configuration. A faulted pool has corrupted metadata, or one or more faulted devices, and insufficient replicas to continue functioning.

The health of the top-level vdev, such as mirror or raidz device, is potentially impacted by the state of its associated vdevs, or component devices. A top-level vdev or component device is in one of the following states:

`DEGRADED`
One or more top-level vdevs is in the degraded state because one or more component devices are offline. Sufficient replicas exist to continue functioning.

One or more component devices is in the degraded or faulted state, but sufficient replicas exist to continue functioning. The underlying conditions are as follows:

- The number of checksum errors exceeds acceptable levels and the device is degraded as an indication that something may be wrong. ZFS continues to use the device as necessary.
- The number of I/O errors exceeds acceptable levels. The device could not be marked as faulted because there are insufficient replicas to continue functioning.

`FAULTED`
One or more top-level vdevs is in the faulted state because one or more component devices are offline. Insufficient replicas exist to continue functioning.

One or more component devices is in the faulted state, and insufficient replicas exist to continue functioning. The underlying conditions are as follows:

- The device could be opened, but the contents did not match expected values.
- The number of I/O errors exceeds acceptable levels and the device is faulted to prevent further use of the device.

`OFFLINE`
The device was explicitly taken offline by the "`zpool offline`" command.

`ONLINE`
The device is online and functioning.

`REMOVED`
The device was physically removed while the system was running. Device removal detection is hardware-dependent and may not be supported on all platforms.

`UNAVAIL`
The device could not be opened. If a pool is imported when a device was unavailable, then the device will be identified by a unique identifier instead of its path since the path was never correct in the first place.

If a device is removed and later reattached to the system, ZFS attempts to put the device online automatically. Device attachment detection is hardware-dependent and might not be supported on all platforms.

Hot Spares ZFS allows devices to be associated with pools as *hot spares*. These devices are not actively used in the pool, but when an active device fails, it is automatically replaced by a hot spare. To create a pool with hot spares, specify a `spare` vdev with any number of devices. For example,

```
# zpool create pool mirror c0d0 c1d0 spare c2d0 c3d0
```

Spares can be shared across multiple pools, and can be added with the `zpool add` command and removed with the `zpool remove` command. When a spare replacement is initiated, a new `spare` vdev is created within the configuration that will remain there until the original device is replaced. At this point, the hot spare becomes available again if another device fails.

If a pool has a shared spare that is currently being used, the pool can not be exported since other pools may use this shared spare, which may lead to potential data corruption.

An in-progress spare replacement can be cancelled by detaching the hot spare. If the original faulted device is detached, then the hot spare assumes its place in the configuration, and is removed from the spare list of all active pools.

Spares cannot replace log devices.

Intent Log

The ZFS Intent Log (ZIL) satisfies POSIX requirements for synchronous transactions. For instance, databases often require their transactions to be on stable storage devices when returning from a system call. NFS and other applications can also use `fsync()` to ensure data stability. By default, the intent log is allocated from blocks within the main pool. However, it might be possible to get better performance using separate intent log devices such as NVRAM or a dedicated disk. For example:

```
# zpool create pool c0d0 c1d0 log c2d0
```

Multiple log devices can also be specified, and they can be mirrored. See the "EXAMPLES" section for an example of mirroring multiple log devices.

Log devices can be added, replaced, attached, detached, and imported and exported as part of the larger pool. Mirrored log devices can be removed by specifying the top-level mirror for the log.

Cache Devices

Devices can be added to a storage pool as *cache devices*. These devices provide an additional layer of caching between main memory and disk. For read-heavy workloads, where the working set size is much larger than what can be cached in main memory, using cache devices allow much more of this working set to be served from low latency media. Using cache devices provides the greatest performance improvement for random read-workloads of mostly static content.

To create a pool with cache devices, specify a `cache` vdev with any number of devices. For example:

```
# zpool create pool c0d0 c1d0 cache c2d0 c3d0
```

Cache devices cannot be mirrored or part of a RAID-2 configuration. If a read error is encountered on a cache device, that read I/O is reissued to the original storage pool device, which might be part of a mirrored or RAID-2 configuration.

The content of the cache devices is considered volatile, as is the case with other system caches.

Processes

Each imported pool has an associated process, named `zpool-`*poolname*. The threads in this process are the pool's I/O processing threads, which handle the compression, checksumming, and other tasks for all I/O associated with the pool. This process exists to provides visibility into the CPU utilization of the system's storage pools. The existence of this process is an unstable interface.

Properties Each pool has several properties associated with it. Some properties are read-only statistics while others are configurable and change the behavior of the pool. The following are read-only properties:

`allocated`
Amount of storage space within the pool that has been physically allocated. This property can also be referred to by its shortened column name, `alloc`.

`capacity`
Percentage of pool space used. This property can also be referred to by its shortened column name, `cap`.

`free`
Number of blocks within the pool that are not allocated.

`guid`
A unique identifier for the pool.

`health`
The current health of the pool. Health can be `ONLINE`, `DEGRADED`, `FAULTED`, `OFFLINE`, `REMOVED`, or `UNAVAIL`.

`size`
Total size of the storage pool.

These space usage properties report actual physical space available to the storage pool. The physical space can be different from the total amount of space that any contained datasets can actually use. The amount of space used in a raidz configuration depends on the characteristics of the data being written. In addition, ZFS reserves some space for internal accounting that the `zfs(1M)` command takes into account, but the `zpool` command does not. For non-full pools of a reasonable size, these effects should be invisible. For small pools, or pools that are close to being completely full, these discrepancies may become more noticeable.

The following property can be set at creation time and import time:

`altroot`
Alternate root directory. If set, this directory is prepended to any mount points within the pool. This can be used when examining an unknown pool where the mount points cannot be trusted, or in an alternate boot environment, where the typical paths are not valid. `altroot` is not a persistent property. It is valid only while the system is up. Setting `altroot` defaults to using `cachefile=none`, though this value may be overridden using an explicit setting.

The following properties can be set at creation time and import time, and later changed with the `zpool set` command:

`autoexpand=on | off`
Controls automatic pool expansion when the underlying LUN is grown. If set to `on`, the pool is resized according to the size of the expanded device. If the device is part of a `mirror`

or `raidz` configuration, then all devices within that `mirror` or`raidz` group must be expanded before the new space is made available to the pool. The default behavior is `off`. This property can also be referred to by its shortened column name, `expand`.

`autoreplace=on | off`
Controls automatic device replacement. If set to `off`, device replacement must be initiated by the administrator by using the `zpool replace` command. If set to `on`, any new device, found in the same physical location as a device that previously belonged to the pool, is automatically formatted and replaced. The default behavior is `off`. This property can also be referred to by its shortened column name, `replace`.

`bootfs=`*pool*`/`*dataset*
Identifies the default bootable dataset for the root pool. This property is expected to be set mainly by the installation and upgrade programs.

`cachefile=`*path* | `none`
Controls the location of where the pool configuration is cached. Discovering all pools on system startup requires a cached copy of the configuration data that is stored on the root file system. All pools in this cache are automatically imported when the system boots. Some environments, such as install and clustering, need to cache this information in a different location so that pools are not automatically imported. Setting this property caches the pool configuration in a different location that can later be imported with `zpool import -c`. Setting it to the special value `none` creates a temporary pool that is never cached, and the special value `''` (empty string) uses the default location.

Multiple pools can share the same cache file. Because the kernel destroys and recreates this file when pools are added and removed, care should be taken when attempting to access this file. When the last pool using a `cachefile` is exported or destroyed, the file is removed.

`delegation=on | off`
Controls whether a non-privileged user is granted access based on the dataset permissions defined on the dataset. See `zfs(1M)` for more information on ZFS delegated administration.

`failmode=wait | continue | panic`
Controls the system behavior in the event of catastrophic pool failure. This condition is typically a result of a loss of connectivity to the underlying storage device(s) or a failure of all devices within the pool. The behavior of such an event is determined as follows:

`wait`
Blocks all I/O access to the pool until the device connectivity is recovered and the errors are cleared. A pool remains in the wait state until the device issue is resolved. This is the default behavior.

`continue`
Returns `EIO` to any new write I/O requests but allows reads to any of the remaining healthy devices. Any write requests that have yet to be committed to disk would be blocked.

`panic`
Prints out a message to the console and generates a system crash dump.

`listsnaps`=on | off
Controls whether information about snapshots associated with this pool is output when `zfs list` is run without the `-t` option. The default value is `on`.

`version`=*version*
The current on-disk version of the pool. The pool version can be increased, but never decreased. The preferred method of updating pools is with the `zpool upgrade` command, though this property can be used when a specific version is needed for backwards compatibility. This property can be any number between 1 and the current version reported by `zpool upgrade -v`.

Subcommands

All subcommands that modify state are logged persistently to the pool in their original form.

The `zpool` command provides subcommands to create and destroy storage pools, add capacity to storage pools, and provide information about the storage pools. The following subcommands are supported:

`zpool -?`
Displays a help message.

`zpool add` [`-fn`] *pool vdev ...*
Adds the specified virtual devices to the given pool. The *vdev* specification is described in the "Virtual Devices" section. The behavior of the `-f` option, and the device checks performed are described in the zpool create subcommand.

`-f`
Forces use of *vdevs*, even if they appear in use or specify a conflicting replication level. Not all devices can be overridden in this manner.

`-n`
Displays the configuration that would be used without actually adding the *vdev*s. The actual pool creation can still fail due to insufficient privileges or device sharing.

Do not add a disk that is currently configured as a quorum device to a storage pool. After a disk is in the pool, that disk can then be configured as a quorum device.

`zpool attach` [`-f`] *pool device new_device*
Attaches *new_device* to an existing `zpool` device. The existing device cannot be part of a raidz configuration. If *device* is not currently part of a mirrored configuration, *device* automatically transforms into a two-way mirror of *device* and *new_device*. If *device* is part of a two-way mirror, attaching *new_device* creates a three-way mirror, and so on. In either case, *new_device* begins to resilver immediately.

`-f`
Forces use of *new_device*, even if its appears to be in use. Not all devices can be overridden in this manner.

`zpool clear` [`-F` [`-n`]] *pool* [*device*] ...
Clears device errors in a pool. If no arguments are specified, all device errors within the pool are cleared. If one or more devices is specified, only those errors associated with the specified device or devices are cleared.

`-F`
Initiates recovery mode for an unopenable pool. Attempts to discard the last few transactions in the pool to return it to an openable state. Not all damaged pools can be recovered by using this option. If successful, the data from the discarded transactions is irretrievably lost.

`-n`
Used in combination with the `-F` flag. Check whether discarding transactions would make the pool openable, but do not actually discard any transactions.

`zpool create` [`-fn`] [`-o` *property=value*] ... [`-O` *file-system-property=value*] ... [`-m` *mountpoint*] [`-R` *root*] *pool vdev* ...
Creates a new storage pool containing the virtual devices specified on the command line. The pool name must begin with a letter, and can only contain alphanumeric characters as well as underscore (_), dash (-), and period (“.”). The pool names `mirror`, `raidz`, `raidz1`, `raidz2`, `raidz3`, `spare`, and `log` are reserved, as are names beginning with the pattern `c[0-9]`. The *vdev* specification is described in the “Virtual Devices” section.

The command verifies that each device specified is accessible and not currently in use by another subsystem. There are some uses, such as being currently mounted, or specified as the dedicated dump device, that prevents a device from ever being used by ZFS. Other uses, such as having a preexisting UFS file system, can be overridden with the `-f` option.

The command also checks that the replication strategy for the pool is consistent. An attempt to combine redundant and non-redundant storage in a single pool, or to mix disks and files, results in an error unless `-f` is specified. The use of differently sized devices within a single `raidz` or `mirror` group is also flagged as an error unless `-f` is specified.

Unless the `-R` option is specified, the default mount point is */pool*. The mount point must not exist or must be empty, or else the root dataset cannot be mounted. This can be overridden with the `-m` option.

`-f`
Forces use of *vdev*s, even if they appear in use or specify a conflicting replication level. Not all devices can be overridden in this manner.

`-n`
Displays the configuration that would be used without actually creating the pool. The actual pool creation can still fail due to insufficient privileges or device sharing.

`-o` *property=value* [`-o` *property=value*] ...
Sets the given pool properties. See the “Properties” section for a list of valid properties that can be set.

`-O` *file-system-property=value*
[`-O` *file-system-property=value*] ...
Sets the given file system properties in the root file system of the pool. See the "Properties" section of `zfs(1M)` for a list of valid properties that can be set.

`-R` *root*
Equivalent to `-o cachefile=none,altroot=`*root*

`-m` *mountpoint*
Sets the mount point for the root dataset. The default mount point is /*pool* or `altroot`/*pool* if `altroot` is specified. The mount point must be an absolute path, `legacy`, or `none`. For more information on dataset mount points, see `zfs(1M)`.

`zpool destroy` [`-f`] *pool*
Destroys the given pool, freeing up any devices for other use. This command tries to unmount any active datasets before destroying the pool.

`-f`
Forces any active datasets contained within the pool to be unmounted.

`zpool detach` *pool device*
Detaches *device* from a mirror. The operation is refused if there are no other valid replicas of the data.

`zpool export` [`-f`] *pool* ...
Exports the given pools from the system. All devices are marked as exported, but are still considered in use by other subsystems. The devices can be moved between systems (even those of different endianness) and imported as long as a sufficient number of devices are present.

Before exporting the pool, all datasets within the pool are unmounted. A pool can not be exported if it has a shared spare that is currently being used.

For pools to be portable, they must be created with whole disks, not just slices, so that ZFS can label the disks with portable EFI labels. Otherwise, disk drivers on platforms of different endianness will not recognize the disks.

`-f`
Forcefully unmount all datasets, using the `unmount -f` command.

This command will forcefully export the pool even if it has a shared spare that is currently being used. This may lead to potential data corruption.

`zpool get all` | *property*[,...] *pool* ...
Retrieves the given list of properties (or all properties if `all` is used) for the specified storage pool(s). These properties are displayed with the following fields:

```
name          Name of storage pool
 property      Property name
 value         Property value
```

```
source        Property source, either 'default' or 'local'.
```

See the "Properties" section for more information on available pool properties.

`zpool history` [`-il`] [*pool*] ...
Displays the command history of the specified pools or all pools if no pool is specified.

`-i`
Displays internally logged ZFS events in addition to user initiated events.

`-l`
Displays log records in long format, which in addition to standard format includes, the user name, the hostname, and the zone in which the operation was performed.

`zpool import` [`-d` *dir* | `-c` *cachefile*] [`-D`]
Lists pools available to import. If the `-d` option is not specified, this command searches for devices in `/dev/dsk`. The `-d` option can be specified multiple times, and all directories are searched. If the device appears to be part of an exported pool, this command displays a summary of the pool with the name of the pool, a numeric identifier, as well as the *vdev* layout and current health of the device for each device or file. Destroyed pools, pools that were previously destroyed with the `zpool destroy` command, are not listed unless the `-D` option is specified.

The numeric identifier is unique, and can be used instead of the pool name when multiple exported pools of the same name are available.

`-c` *cachefile*
Reads the pool configuration information from the given `cachefile` that was created with the `cachefile` pool property. This `cachefile` is used instead of searching for devices.

`-d` *dir*
Searches for devices or files in *dir*. The `-d` option can be specified multiple times.

`-D`
Lists destroyed pools only.

`zpool import` [`-o` *mntopts*] [`-o` *property*=*value*] ... [`-d` *dir* | `-c` *cachefile*] [`-D`] [`-f`] [`-R` *root*] [`-F` [`-n`]] `-a`
Imports all pools found in the search directories. Identical to the previous command, except that all pools with a sufficient number of devices available are imported. Destroyed pools, pools that were previously destroyed with the `zpool destroy` command, will not be imported unless the `-D` option is specified.

`-o` *mntopts*
Comma-separated list of mount options to use when mounting datasets within the pool. See `zfs(1M)` for a description of dataset properties and mount options.

`-o` *property=value*
Sets the specified property on the imported pool. See the "Properties" section for more information on available pool properties.

`-c` *cachefile*
Reads the pool configuration information from the given `cachefile` that was created with the "`cachefile`" pool property. This `cachefile` is used instead of searching for devices.

`-d` *dir*
Searches for devices or files in *dir*. The `-d` option can be specified multiple times. This option is incompatible with the `-c` option.

`-D`
Imports destroyed pools only.

`-f`
Forces import, even if the pool appears to be potentially active.

`-F`
Recovery mode for a non-importable pool. Attempts to return the pool to an importable state by discarding the last few transactions. Not all damaged pools can be recovered by using this option. If successful, the data from the discarded transactions is irretrievably lost. This option is ignored if the pool is importable or already imported.

`-a`
Searches for and imports all pools found.

`-R` *root*
Sets the `cachefile` property to `none` and the *altroot* property to *root*.

`-n`
Used with the `-F` recovery option. Determines whether a non-importable pool can be made importable again, but does not actually perform the pool recovery. For more details about pool recovery mode, see the `-F` option, above.

`zpool import` [`-o` *mntopts*] [`-o` *property=value*] ... [`-d` *dir* | `-c` *cachefile*] [`-D`] [`-f`] [`-R` *root*] [`-F` [`-n`]] *pool* | *id* [*newpool*]
Imports a specific pool. A pool can be identified by its name or by the numeric identifier. If *newpool* is specified, the pool is imported using the name *newpool*. Otherwise, it is imported with the same name as its exported name.

If a device is removed from a system without running `zpool export` first, the device appears as potentially active. It cannot be determined if this was a failed export, or whether the device is really in use from another host. To import a pool in this state, the `-f` option is required.

`-o` *mntopts*
Comma-separated list of mount options to use when mounting datasets within the pool. See `zfs(1M)` for a description of dataset properties and mount options.

`-o` *property=value*
Sets the specified property on the imported pool. See the "Properties" section for more information on available pool properties.

`-c` *cachefile*
Reads configuration from the given `cachefile` that was created with the `cachefile` pool property. This `cachefile` is used instead of searching for devices.

`-d` *dir*
Searches for devices or files in *dir*. The `-d` option can be specified multiple times. This option is incompatible with the `-c` option.

`-D`
Imports destroyed pool. The `-f` option is also required.

`-f`
Forces import, even if the pool appears to be potentially active.

`-F`
Recovery mode for a non-importable pool. Attempts to return the pool to an importable state by discarding the last few transactions. Not all damaged pools can be recovered by using this option. If successful, the data from the discarded transactions is irretrievably lost. This option is ignored if the pool is importable or already imported.

`-R` *root*
Sets the `cachefile` property to `none` and the *altroot* property to *root*.

`-n`
Used with the `-F` recovery option. Determines whether a non-importable pool can be made importable again, but does not actually perform the pool recovery. For more details about pool recovery mode, see the `-F` option, above.

`zpool iostat` [`-v`] [*pool*] ... [*interval*[*count*]]
Displays I/O statistics for the given pools. When given an interval, the statistics are printed every *interval* seconds until `Ctrl-C` is pressed. If no *pools* are specified, statistics for every pool in the system is shown. If *count* is specified, the command exits after *count* reports are printed.

`-v`
Verbose statistics. Reports usage statistics for individual *vdevs* within the pool, in addition to the pool-wide statistics.

`zpool list` [`-H`] [`-o` *props*[,...]] [*pool*] ...
Lists pool health status and space usage. The default list of properties displayed is `name`, `size`, `allocated`, `free`, `capacity`, `health`, and `altroot`. With no arguments, all pools in the system are listed.

`-H`
Scripted mode. Do not display headers, and separate fields by a single tab instead of arbitrary space.

`-o` *props*
Comma-separated list of properties to display. See the "Properties" section for a list of valid properties.

`zpool offline` [`-t`] *pool device* ...
Takes the specified physical device offline. While the *device* is offline, no attempt is made to read or write to the device.

This command is not applicable to spares or cache devices.

`-t`
Temporary. Upon reboot, the specified physical device reverts to its previous state.

`zpool online` [`-e`] *pool device*...
Brings the specified physical device online.

This command is not applicable to spares or cache devices.

`-e`
Expand the device to use all available space. If the device is part of a mirrored RAID-2 configuration, then all devices must be expanded before the new space is made available to the pool.

`zpool remove` *pool device* ...
Removes the specified device from the pool. This command currently only supports removing hot spares, cache, and log devices. A mirrored log device can be removed by specifying the top-level mirror for the log. Non-log devices that are part of a mirrored configuration can be removed using the `zpool detach` command. Non-redundant and raidz devices cannot be removed from a pool.

`zpool replace` [`-f`] *pool old_device* [*new_device*]
Replaces *old_device* with *new_device*. This is equivalent to attaching *new_device*, waiting for it to resilver, and then detaching *old_device*.

The size of *new_device* must be greater than or equal to the minimum size of all the devices in a mirror or raidz configuration.

new_device is required if the pool is not redundant. If *new_device* is not specified, it defaults to *old_device*. This form of replacement is useful after an existing disk has failed and has been physically replaced. In this case, the new disk may have the same `/dev/dsk` path as the old device, even though it is actually a different disk. ZFS recognizes this.

`-f`
Forces use of *new_device*, even if its appears to be in use. Not all devices can be overridden in this manner.

`zpool scrub` [`-s`] *pool* ...
Begins a scrub. The scrub examines all data in the specified pools to verify that it checksums correctly. For replicated (`mirror` or `raidz`) devices, ZFS automatically repairs

any damage discovered during the scrub. The `zpool status` command reports the progress of the scrub and summarizes the results of the scrub upon completion.

Scrubbing and resilvering are very similar operations. The difference is that resilvering only examines data that ZFS knows to be out of date (for example, when attaching a new device to a mirror or replacing an existing device), whereas scrubbing examines all data to discover silent errors due to hardware faults or disk failure.

Because scrubbing and resilvering are I/O-intensive operations, ZFS only allows one at a time. If a scrub is already in progress, the "`zpool scrub`" command terminates it and starts a new scrub. If a resilver is in progress, ZFS does not allow a scrub to be started until the resilver completes.

`-s`
Stop scrubbing.

`zpool set` *property=value pool*
Sets the given property on the specified pool. See the "Properties" section for more information on what properties can be set and acceptable values.

`zpool split` [`-R` *altroot*] [`-n`] [`-o` *mntopts*] [`-o` *property=value*] *pool newpool* [*device* ...]
Splits off one disk from each mirrored top-level vdev in a pool and creates a new pool from the split-off disks. The original pool must be made up of one or more mirrors and must not be in the process of resilvering. The `split` subcommand chooses the last device in each mirror vdev unless overridden by a device specification on the command line.

When using a *device* argument, `split` includes the specified device(s) in a new pool and, should any devices remain unspecified, assigns the last device in each mirror vdev to that pool, as it does normally. If you are uncertain about the outcome of a `split` command, use the `-n` (*dry-run*) option to confirm the intended command syntax.

`-R` *altroot*
Automatically import the newly created pool after splitting, using the specified *altroot* parameter for the new pool's alternate root. See the `altroot` description in the "Properties" section, above.

`-n`
Displays the configuration that would be created without actually splitting the pool. The actual pool split could still fail due to insufficient privileges or device status.

`-o` *mntopts*
Comma-separated list of mount options to use when mounting datasets within the pool. See `zfs(1M)` for a description of dataset properties and mount options. Valid only in conjunction with the `-R` option.

`-o` *property=value*
Sets the specified property on the new pool. See the "Properties" section, above, for more information on available pool properties.

`zpool status` [`-xv`] [*pool*] ...
Displays the detailed health status for the given pools. If no *pool* is specified, then the status of each pool in the system is displayed. For more information on pool and device health, see the "Device Failure and Recovery" section.

If a scrub or resilver is in progress, this command reports the percentage done and the estimated time to completion. Both of these are only approximate, because the amount of data in the pool and the other workloads on the system can change.

`-x`
Only display status for pools that are exhibiting errors or are otherwise unavailable.

`-v`
Displays verbose data error information, printing out a complete list of all data errors since the last complete pool scrub.

`zpool upgrade`
Identifies a pool's on-disk version, which determines available pool features in the currently running software release. You can continue to use older pool versions, but some features may not be available. A pool can be upgraded by using the `zpool upgrade -a` command. You will not be able to access a pool of a later version on a system that runs an earlier software version.

`zpool upgrade -v`
Displays ZFS versions supported by the current software. The current ZFS pool versions and all previous supported versions are displayed, along with an explanation of the features provided with each version.

`zpool upgrade` [`-V` *version*] `-a` | *pool* ...
Upgrades the given pool to the latest on-disk version. If this command identifies that a pool is out-of-date, then it can be upgraded by using the `zpool upgrade -a` command. A pool that is upgraded will not be accessible on a system that runs an earlier software release.

`-a`
Upgrades all pools.

`-V` *version*
Upgrade to the specified version. If the `-V` flag is not specified, the pool is upgraded to the most recent version. This option can only be used to increase the version number, and only up to the most recent version supported by this software.

Examples

EXAMPLE 1 Creating a RAID-Z Storage Pool

The following command creates a pool with a single raidz root *vdev* that consists of six disks.

```
# zpool create tank raidz c0t0d0 c0t1d0 c0t2d0 c0t3d0 c0t4d0 c0t5d0
```

EXAMPLE 2 Creating a Mirrored Storage Pool

The following command creates a pool with two mirrors, where each mirror contains two disks.

```
# zpool create tank mirror c0t0d0 c0t1d0 mirror c0t2d0 c0t3d0
```

EXAMPLE 3 Adding a Mirror to a ZFS Storage Pool

The following command adds two mirrored disks to the pool *tank*, assuming the pool is already made up of two-way mirrors. The additional space is immediately available to any datasets within the pool.

```
# zpool add tank mirror c1t0d0 c1t1d0
```

EXAMPLE 4 Listing Available ZFS Storage Pools

The following command lists all available pools on the system.

```
# zpool list
NAME    SIZE  ALLOC   FREE    CAP  HEALTH  ALTROOT
pool    136G  95.5K   136G     0%  ONLINE  -
rpool    93G   111K  93.0G     0%  ONLINE  -
```

EXAMPLE 5 Listing All Properties for a Pool

The following command lists all the properties for a pool.

```
% zpool get all pool
NAME  PROPERTY       VALUE       SOURCE
pool  size           136G        -
pool  capacity       0%          -
pool  altroot        -           default
pool  health         ONLINE      -
pool  guid           4898854009727303489  default
pool  version        22          default
pool  bootfs         -           default
pool  delegation     on          default
pool  autoreplace    off         default
pool  cachefile      -           default
pool  failmode       wait        default
pool  listsnapshots  off         default
pool  autoexpand     off         default
pool  free           136G        -
pool  allocated      76.5K       -
```

EXAMPLE 6 Destroying a ZFS Storage Pool

The following command destroys the pool `tank` and any datasets contained within.

```
# zpool destroy -f tank
```

EXAMPLE 7 Exporting a ZFS Storage Pool

The following command exports the devices in pool `tank` so that they can be relocated or later imported.

```
# zpool export tank
```

EXAMPLE 8 Importing a ZFS Storage Pool

The following command displays available pools, and then imports the pool `tank` for use on the system.

The results from this command are similar to the following:

```
# zpool import
  pool: tank
    id: 7678868315469843843
 state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        tank        ONLINE
          mirror-0  ONLINE
            c1t2d0  ONLINE
            c1t3d0  ONLINE

# zpool import tank
```

EXAMPLE 9 Upgrading All ZFS Storage Pools to the Current Version

The following command upgrades all ZFS Storage pools to the current version of the software.

```
# zpool upgrade -a
This system is currently running ZFS pool version 22.

All pools are formatted using this version.
```

EXAMPLE 10 Managing Hot Spares

The following command creates a new pool with an available hot spare:

```
# zpool create tank mirror c0t0d0 c0t1d0 spare c0t2d0
```

If one of the disks were to fail, the pool would be reduced to the degraded state. The failed device can be replaced using the following command:

```
# zpool replace tank c0t0d0 c0t3d0
```

After the device has been resilvered, the spare is automatically detached and is made available should another device fail. The hot spare can be permanently removed from the pool using the following command:

```
# zpool remove tank c0t2d0
```

EXAMPLE 11 Creating a ZFS Pool with Mirrored Separate Intent Logs

The following command creates a ZFS storage pool consisting of two, two-way mirrors and mirrored log devices:

```
# zpool create pool mirror c0d0 c1d0 mirror c2d0 c3d0 log mirror \
   c4d0 c5d0
```

EXAMPLE 12 Adding Cache Devices to a ZFS Pool

The following command adds two disks for use as cache devices to a ZFS storage pool:

```
# zpool add pool cache c2d0 c3d0
```

Once added, the cache devices gradually fill with content from main memory. Depending on the size of your cache devices, it could take over an hour for them to fill. Capacity and reads can be monitored using the `iostat` option as follows:

```
# zpool iostat -v pool 5
```

EXAMPLE 13 Removing a Mirrored Log Device

The following command removes the mirrored log device `mirror-2`.

Given this configuration:

```
   pool: tank
  state: ONLINE
  scrub: none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tank        ONLINE       0     0     0
          mirror-0  ONLINE       0     0     0
            c6t0d0  ONLINE       0     0     0
            c6t1d0  ONLINE       0     0     0
          mirror-1  ONLINE       0     0     0
            c6t2d0  ONLINE       0     0     0
            c6t3d0  ONLINE       0     0     0
        logs
          mirror-2  ONLINE       0     0     0
            c4t0d0  ONLINE       0     0     0
            c4t1d0  ONLINE       0     0     0
```

The command to remove the mirrored log `mirror-2` is:

```
# zpool remove tank mirror-2
```

EXAMPLE 14 Recovering a Faulted ZFS Pool

If a pool is faulted but recoverable, a message indicating this state is provided by `zpool status` if the pool was cached (see `cachefile` above), or as part of the error output from a failed `zpool import` of the pool.

EXAMPLE 14 Recovering a Faulted ZFS Pool *(Continued)*

Recover a cached pool with the `zpool clear` command:

```
# zpool clear -F data
Pool data returned to its state as of Tue Sep 08 13:23:35 2009.
Discarded approximately 29 seconds of transactions.
```

If the pool configuration was not cached, use `zpool import` with the recovery mode flag:

```
# zpool import -F data
Pool data returned to its state as of Tue Sep 08 13:23:35 2009.
Discarded approximately 29 seconds of transactions.
```

Exit Status

The following exit values are returned:

`0`
: Successful completion.

`1`
: An error occurred.

`2`
: Invalid command line options were specified.

Attributes

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/file-system/zfs
Interface Stability	Committed

See Also

`zfs(1M)`, `attributes(5)`

Name zuludaemon – load microcode for Sun XVR-4000 Graphics Accelerator device

Synopsis

```
/usr/sbin/zuludaemon [-dev zulu_device]
```

Description The `zuludaemon` is a daemon, started and stopped from a script in `/etc/init.d`, that loads the microcode and provides other support functions for the Sun XVR-4000 Graphics Accelerator device. Do not kill this process. This deamon is essential to the `zulu(7d)` driver.

The `zuludaemon` process is not configurable.

Options

`-dev` *zulu_device* name of the Sun XVR-4000 Graphics Accelerator device

Files

`/usr/sbin/zuludaemon` daemon executable

`/usr/lib/zulu.ucode` file containing microcode used by `zuludaemon`

`/etc/init.d/zuluinit` startup/kill script for `zuludaemon`

Attributes See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWzuluc

See Also `SUNWzulu_config(1M)`, `attributes(5)`, `zulu(7d)`