

man pages section 4: File Formats

Copyright © 2011, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	9
Introduction	13
Intro(4)	14
File Formats	15
admin(4)	16
alias(4)	22
aliases(4)	23
a.out(4)	28
au(4)	30
audit_class(4)	33
audit_event(4)	35
audit.log(4)	37
auth_attr(4)	46
autofs(4)	49
bart_manifest(4)	51
bart_rules(4)	54
bootparams(4)	58
cardbus(4)	60
compver(4)	64
contents(4)	65
contract(4)	67
copyright(4)	73
core(4)	74
crypt.conf(4)	79
crypto_certs(4)	81

dacf.conf(4)	82
dat.conf(4)	83
defaultdomain(4)	85
default_fs(4)	86
defaultrouter(4)	87
depend(4)	88
device_allocate(4)	90
device_contract(4)	93
device_maps(4)	97
devices(4)	99
dfstab(4)	100
dhcp_inittab(4)	101
dhcp_network(4)	109
dhcpsvc.conf(4)	112
dhcptab(4)	115
dialups(4)	120
dir_ufs(4)	121
d_passwd(4)	122
driver(4)	124
driver.conf(4)	125
ds.log(4)	134
ethers(4)	135
exec_attr(4)	136
fd(4)	140
fdi(4)	141
flash_archive(4)	145
format.dat(4)	154
fspec(4)	158
fstypes(4)	160
ftp(4)	161
ftpusers(4)	162
fx_dptbl(4)	164
gateways(4)	171
geniconvtbl(4)	175
group(4)	195
gsscred.conf(4)	197

hba.conf(4)	198
holidays(4)	200
hosts(4)	202
hosts.equiv(4)	205
ib(4)	208
idnkit.pc(4)	209
ike.config(4)	210
ike.preshared(4)	223
inetd.conf(4)	225
inet_type(4)	228
init.d(4)	229
inittab(4)	231
ipaddrsel.conf(4)	234
ipf(4)	235
ipnat(4)	245
ipnodes(4)	250
ippool(4)	251
issue(4)	254
kadm5.acl(4)	255
kdc.conf(4)	258
keytables(4)	268
krb5.conf(4)	276
label_encodings(4)	296
ldapfilter.conf(4)	301
ldapsearchprefs.conf(4)	303
ldaptemplates.conf(4)	307
llc2(4)	311
logadm.conf(4)	318
logindevperm(4)	319
loginlog(4)	321
magic(4)	322
mddb.cf(4)	325
md.tab(4)	326
mech(4)	332
meddb(4)	334
mnttab(4)	335

mod_ipp(4)	338
mpapi.conf(4)	342
named.conf(4)	343
ncad_addr(4)	354
nca.if(4)	355
ncakmod.conf(4)	357
ncalogd.conf(4)	359
ncaport.conf(4)	361
ndmp(4)	362
ndpd.conf(4)	364
netconfig(4)	371
netgroup(4)	375
netid(4)	377
netmasks(4)	379
netrc(4)	381
networks(4)	383
nfs(4)	384
nfslog.conf(4)	387
nfssec.conf(4)	390
NISLDAPmapping(4)	391
nodename(4)	408
nologin(4)	410
note(4)	411
notrouter(4)	412
nscd.conf(4)	413
nss(4)	416
nsswitch.conf(4)	417
packingrules(4)	424
pam.conf(4)	427
passwd(4)	433
path_to_inst(4)	437
pci(4)	439
phones(4)	443
pkginfo(4)	444
pkgmap(4)	454
plot(4B)	458

policy.conf(4)	459
priv_names(4)	462
proc(4)	463
process(4)	496
prof_attr(4)	502
profile(4)	505
project(4)	507
protocols(4)	510
prototype(4)	512
pseudo(4)	517
publickey(4)	518
queuedefs(4)	519
rcmscript(4)	521
rdc.cf(4)	532
registration_profile(4)	534
remote(4)	536
resolv.conf(4)	540
rmtab(4)	543
rndc.conf(4)	544
rpc(4)	547
rt_dptbl(4)	548
sasl_appname.conf(4)	554
sbus(4)	556
sccsfile(4)	559
scsi(4)	562
securenets(4)	566
sel_config(4)	568
sendmail(4)	570
service_bundle(4)	573
service_provider.conf(4)	575
services(4)	577
shadow(4)	578
sharetab(4)	581
shells(4)	582
slp.conf(4)	583
slpd.reg(4)	591

smb(4)	594
smbautohome(4)	599
smhba.conf(4)	601
sndr(4)	603
sock2path.d(4)	604
space(4)	605
ssh_config(4)	606
sshd_config(4)	619
sulog(4)	635
sysbus(4)	636
syslog.conf(4)	638
system(4)	641
telnetrc(4)	645
term(4)	646
terminfo(4)	649
TIMEZONE(4)	697
timezone(4)	698
TrustedExtensionsPolicy(4)	699
ts_dptbl(4)	701
ttydefs(4)	708
ttysrch(4)	709
ufsdump(4)	711
updaters(4)	717
user_attr(4)	718
utmp(4)	723
utmpx(4)	724
vfstab(4)	725
volume-config(4)	727
volume-request(4)	731
wanboot.conf(4)	737
warn.conf(4)	742
ypfiles(4)	744
yppasswdd(4)	746
ypserv(4)	747
zoneinfo(4)	756

Preface

Both novice users and those familiar 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 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 9E describes the DDI (Device Driver Interface)/DKI (Driver/Kernel Interface), 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	<p>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.</p> <p>The following special characters are used in this section:</p> <ul style="list-style-type: none">[] 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 <code>ioctl</code> and generates its own heading. <code>ioctl</code> calls for a specific device are listed alphabetically (on the man page for that specific device).

	<p><code>ioctl</code> calls are used for a particular class of devices all of which have an <code>io</code> ending, such as <code>mtio(7I)</code>.</p>
OPTIONS	<p>This section 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.</p>
OPERANDS	<p>This section lists the command operands and describes how they affect the actions of the command.</p>
OUTPUT	<p>This section describes the output – standard output, standard error, or output files – generated by the command.</p>
RETURN VALUES	<p>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.</p>
ERRORS	<p>On failure, most functions place an error code in the global variable <code>errno</code> 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.</p>
USAGE	<p>This section lists special rules, features, and commands that require in-depth explanations. The subsections listed here are used to explain built-in functionality:</p> <ul style="list-style-type: none">CommandsModifiersVariablesExpressionsInput Grammar
EXAMPLES	<p>This section provides examples of usage or of how to use a command or function. Wherever possible a complete</p>

	<p>example including command-line entry and machine response is shown. Whenever an example is given, the prompt is shown as <code>example%</code>, or if the user must be superuser, <code>example#</code>. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS, and USAGE sections.</p>
ENVIRONMENT VARIABLES	<p>This section lists any environment variables that the command or function affects, followed by a brief description of the effect.</p>
EXIT STATUS	<p>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.</p>
FILES	<p>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.</p>
ATTRIBUTES	<p>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.</p>
SEE ALSO	<p>This section lists references to other man pages, in-house documentation, and outside publications.</p>
DIAGNOSTICS	<p>This section lists diagnostic messages with a brief explanation of the condition causing the error.</p>
WARNINGS	<p>This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.</p>
NOTES	<p>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.</p>
BUGS	<p>This section describes known bugs and, wherever possible, suggests workarounds.</p>

R E F E R E N C E

Introduction

Name Intro – introduction to file formats

Description This section outlines the formats of various files. The C structure declarations for the file formats are given where applicable. Usually, the headers containing these structure declarations can be found in the directories `/usr/include` or `/usr/include/sys`. For inclusion in C language programs, however, the syntax `#include <filename.h>` or `#include <sys/filename.h>` should be used.

REFERENCE

File Formats

Name admin – installation defaults file

Description admin is a generic name for an ASCII file that defines default installation actions by assigning values to installation parameters. For example, it allows administrators to define how to proceed when the package being installed already exists on the system.

`/var/sadm/install/admin/default` is the default admin file delivered with this release. The default file is not writable, so to assign values different from this file, create a new admin file. There are no naming restrictions for admin files. Name the file when installing a package with the `-a` option of `pkgadd(1M)`. If the `-a` option is not used, the default admin file is used.

Each entry in the admin file is a line that establishes the value of a parameter in the following form:

param=value

All of the parameters listed below can be defined in an admin file, but it is not required to assign values to all of these. If a value is not assigned, `pkgadd(1M)` asks the installer how to proceed.

The valid parameters and their possible values are shown below except as noted. They can be specified in any order. Any of these parameters (except the `mail` and `proxy` parameters) can be assigned the value `ask`, which means that, when the parameter is reached during the installation sequence, the installer is notified and asked to supply instructions (see NOTES).

`basedir`

Indicates the base directory where relocatable packages are to be installed. If there is no `basedir` entry in the file, the installer will be prompted for a path name, as if the file contained the entry `basedir=ask`. This parameter can also be set to `default` (entry is `basedir=default`). In this instance, the package is installed into the base directory specified by the `BASEDIR` parameter in the `pkginfo(4)` file.

`mail`

Defines a list of users to whom mail should be sent following installation of a package. If the list is empty, no mail is sent. If the parameter is not present in the admin file, the default value of `root` is used. The `ask` value cannot be used with this parameter.

`runlevel`

Indicates resolution if the run level is not correct for the installation or removal of a package. Options are:

`nocheck`

Do not check for run level.

`quit`

Abort installation if run level is not met.

`conflict`

Specifies what to do if an installation expects to overwrite a previously installed file, thus creating a conflict between packages. Options are:

`nocheck`

Do not check for conflict; files in conflict will be overwritten.

`quit`

Abort installation if conflict is detected.

`nochange`

Override installation of conflicting files; they will not be installed.

`setuid`

Checks for executables which will have `setuid` or `setgid` bits enabled after installation.

Options are:

`nocheck`

Do not check for `setuid` executables.

`quit`

Abort installation if `setuid` processes are detected.

`nochange`

Override installation of `setuid` processes; processes will be installed without `setuid` bits enabled.

`action`

Determines if action scripts provided by package developers contain possible security impact. Options are:

`nocheck`

Ignore security impact of action scripts.

`quit`

Abort installation if action scripts may have a negative security impact.

`partial`

Checks to see if a version of the package is already partially installed on the system. Options are:

`nocheck`

Do not check for a partially installed package.

`quit`

Abort installation if a partially installed package exists.

`instance`

Determines how to handle installation if a previous version of the package (including a partially installed instance) already exists. Options are:

`quit`

Exit without installing if an instance of the package already exists (does not overwrite existing packages).

overwrite

Overwrite an existing package if only one instance exists. If there is more than one instance, but only one has the same architecture, it overwrites that instance. Otherwise, the installer is prompted with existing instances and asked which to overwrite.

unique

Do not overwrite an existing instance of a package. Instead, a new instance of the package is created. The new instance will be assigned the next available instance identifier.

idepend

Controls resolution if the package to be installed depends on other packages and if other packages depend on the one to be installed. Options are:

nocheck

Do not check package dependencies.

quit

Abort installation if package dependencies are not met.

rdepend

Controls resolution if other packages depend on the package to be removed. Also determines behavior if registered products components to be removed. See [libwsreg\(3LIB\)](#) for a definition of product components. Options are:

nocheck

Do not check package or product dependencies.

quit

Abort removal if package or product dependencies are not met.

space

Controls resolution if disk space requirements for package are not met. Options are:

nocheck

Do not check space requirements (installation fails if it runs out of space).

quit

Abort installation if space requirements are not met.

authentication

Controls resolution when a datastream package with signature is to be installed. Options are:

nocheck

Do not verify package signature. This also disables the use of the Online Certificate Status Protocol (OCSP) to validate the package's signing certificate.

quit

Abort installation if package signature cannot be verified.

networktimeout

Number of seconds to wait before giving up a network connection when downloading a package. This entry must be a positive integer. If not present, the default value of 60 is used.

networkretries

Number of times to retry a failed network connection when downloading a package. This entry must be a positive integer. If not present, the default value of 5 is used.

keystore

Location of trusted certificates used when downloading packages over SSL and when verifying signatures on packages. This is the base directory of the certificate location for trusted certificates used when validating digital signatures on packages. For example, if this setting is `/var/sadm/security`, then `pkgadd` will use `/var/sadm/security/pkgadd/truststore`, then `/var/sadm/security/truststore` when searching for trusted certificates. See **KEYSTORE LOCATIONS** and **KEYSTORE AND CERTIFICATE FORMATS** in [pkgadd\(1M\)](#) for details on certificate store format and usage.

proxy

The default proxy to use when installing packages from the network. Currently, only HTTP or HTTPS proxies are supported. If this field is blank or nonexistent, then no proxy will be used.

rscriptalt=root | noaccess

Determines the user that will run request scripts. This parameter can have either of the values described below. See [pkgadd\(1M\)](#) for details on the conditions under which this parameter is useful.

root

Run request script as user `install`, if such a user exists, with the privileges of that user. Otherwise, run script as user `root`, with UID equal to 0 and with all/zone privileges. (See [zones\(5\)](#).)

noaccess

Run request script as user `install`, if such a user exists, with the privileges of that user. Otherwise, run script as user `noaccess`, with the basic privileges of the unprivileged user `noaccess`.

If this parameter is not present or has a null value, the user `noaccess` is assumed. Likewise, if this parameter is set to anything other than the values described here, a warning is issued, and `noaccess` is assumed. `rscriptalt` is not present in the default `admin` file, `/var/sadm/install/admin/default`. In this case, request scripts are run as the user `noaccess`.

Examples EXAMPLE 1 Default admin File

The default `admin` file, named `default`, is shipped with `user-`, `group-`, and world-read privileges (444). Its contents are as follows:

EXAMPLE 1 Default admin File (Continued)

```
mail=
instance=unique
partial=ask
runlevel=ask
idepend=ask
rdepend=ask
space=ask
setuid=ask
conflict=ask
action=ask
basedir=default
authentication=quit
networktimeout=10
networkretries=3
keystore=/var/sadm/security
proxy=
```

EXAMPLE 2 Sample admin file.

Below is a sample admin file.

```
basedir=default
runlevel=quit
conflict=quit
setuid=quit
action=quit
partial=quit
instance=unique
idepend=quit
rdepend=quit
space=quit
authentication=quit
networktimeout=10
networkretries=5
keystore=/opt/certs
proxy=syrinx.eng.example.com:8080
```

Files The default admin file is consulted during package installation when no other admin file is specified.

```
/var/sadm/install/admin/default
    default admin file
```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	package/svr4
Interface Stability	Committed

See Also [pkgadd\(1M\)](#), [libwsreg\(3LIB\)](#), [pkginfo\(4\)](#), [attributes\(5\)](#), [zones\(5\)](#)

Notes The value `ask` should not be defined in an `admin` file that will be used for non-interactive installation (because, by definition, there is no installer interaction). Doing so causes installation to fail at the point when input is needed.

Name alias – alias table file of encoding names

Synopsis /usr/lib/iconv/alias

Description This file contains the alias table of encoding names for [iconv_open\(3C\)](#).

The format of the alias table is as follows:

```
"%s %s\n", <variant encoding name>, <canonical encoding name>
```

The string specified for the variant encoding name is case-insensitive. A line beginning with '#' is treated as a comment.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Committed

See Also [iconv\(3C\)](#), [iconv_close\(3C\)](#), [iconv_open\(3C\)](#), [attributes\(5\)](#)

Name aliases, addresses, forward – addresses and aliases for sendmail

Synopsis /etc/mail/aliases
 /etc/mail/aliases.db
 /etc/mail/aliases.dir
 /etc/mail/aliases.pag
 ~/.forward

Description These files contain mail addresses or aliases, recognized by `sendmail(1M)` for the local host:

/etc/passwd	Mail addresses (usernames) of local users.
/etc/mail/aliases	Aliases for the local host, in ASCII format. Root can edit this file to add, update, or delete local mail aliases.
/etc/mail/aliases.{dir,pag}	The aliasing information from /etc/mail/aliases, in binary <code>ndbm(3C)</code> format for use by <code>sendmail(1M)</code> . The program <code>newaliases(1M)</code> maintains these files.
/etc/mail/aliases.db	The aliasing information from /etc/mail/aliases, in binary, Berkeley DataBase format for use by <code>sendmail(1M)</code> . The program maintains these files.
	Depending on the configuration of the AliasFile option in /etc/mail/sendmail.cf, either the single file aliases.db or the pair of files aliases.{dir,pag} is generated by <code>newaliases(1M)</code> . As shipped with Solaris, <code>sendmail(1M)</code> supports both formats. If neither is specified, the Berkeley DataBase format which generates the single .db file is used.
~/.forward	Addresses to which a user's mail is forwarded (see Automatic Forwarding).

In addition, the NIS name services aliases map `mail.aliases` contains addresses and aliases available for use across the network.

Addresses As distributed, `sendmail(1M)` supports the following types of addresses:

Local Usernames *username*

Each local *username* is listed in the local host's /etc/passwd file.

Local Filenames *pathname*

Messages addressed to the absolute *pathname* of a file are appended to that file.

Commands | `command`

If the first character of the address is a vertical bar (`|`), `sendmail(1M)` pipes the message to the standard input of the command the bar precedes.

Internet-standard
Addresses `username@domain`

If *domain* does not contain any `'.'` (dots), then it is interpreted as the name of a host in the current domain. Otherwise, the message is passed to a *mailhost* that determines how to get to the specified domain. Domains are divided into subdomains separated by dots, with the top-level domain on the right.

For example, the full address of John Smith could be:

```
js@jsmachine.Podunk-U.EDU
```

if he uses the machine named `jsmachine` at Podunk University.

uucp Addresses `. . . [host!] host!username`

These are sometimes mistakenly referred to as "Usenet" addresses. `uucp(1C)` provides links to numerous sites throughout the world for the remote copying of files.

Other site-specific forms of addressing can be added by customizing the `sendmail.cf` configuration file. See `sendmail(1M)` for details. Standard addresses are recommended.

Aliases

Local Aliases

`/etc/mail/aliases` is formatted as a series of lines of the form

```
aliasname:address[, address]
```

aliasname is the name of the alias or alias group, and *address* is the address of a recipient in the group. Aliases can be nested. That is, an *address* can be the name of another alias group. Because of the way `sendmail(1M)` performs mapping from upper-case to lower-case, an *address* that is the name of another alias group must not contain any upper-case letters.

Lines beginning with white space are treated as continuation lines for the preceding alias. Lines beginning with `#` are comments.

Special Aliases

An alias of the form:

```
owner-aliasname : address
```


sendmail directs error-messages resulting from mail to *aliasname* to *address*, instead of back to the person who sent the message. sendmail rewrites the SMTP envelope sender to match this, so owner-*aliasname* should always point to *alias-request*, and *alias-request* should point to the owner's actual address:

```
owner-aliasname:      aliasname-request
aliasname-request    address
```

An alias of the form:

```
aliasname: :include:pathname
```

with colons as shown, adds the recipients listed in the file *pathname* to the *aliasname* alias. This allows a private list to be maintained separately from the aliases file.

NIS Domain Aliases

The aliases file on the master NIS server is used for the *mail.aliases* NIS map, which can be made available to every NIS client. Thus, the */etc/mail/aliases** files on the various hosts in a network will one day be obsolete. Domain-wide aliases should ultimately be resolved into usernames on specific hosts. For example, if the following were in the domain-wide alias file:

```
jsmith:js@jsmachine
```

then any NIS client could just mail to *jsmith* and not have to remember the machine and username for John Smith.

If an NIS alias does not resolve to an address with a specific host, then the name of the NIS domain is used. There should be an alias of the domain name for a host in this case.

For example, the alias:

```
jsmith:root
```

sends mail on an NIS client to *root@podunk-u* if the name of the NIS domain is *podunk-u*.

Automatic Forwarding When an alias (or address) is resolved to the name of a user on the local host, [sendmail\(1M\)](#) checks for a *~/forward* file, owned by the intended recipient, in that user's home directory, and with universal read access. This file can contain one or more addresses or aliases as described above, each of which is sent a copy of the user's mail.

Care must be taken to avoid creating addressing loops in the *~/forward* file. When forwarding mail between machines, be sure that the destination machine does not return the mail to the sender through the operation of any NIS aliases. Otherwise, copies of the message may "bounce." Usually, the solution is to change the NIS alias to direct mail to the proper destination.

A backslash before a username inhibits further aliasing. For instance, to invoke the *vacation* program, user *js* creates a *~/forward* file that contains the line:

```
\js, "|/usr/ucb/vacation js"
```

so that one copy of the message is sent to the user, and another is piped into the vacation program.

The `~/ .forward` file can be used to specify special “per user” extensions by creating a `.forward+extension` file in the home directory. For example, with an address like `jsmith+jerry@jsmachine`, the `sendmail(1M)` utility recognizes everything before the “+” as the actual username (`jsmith`) and everything after it, up to the “@” symbol, as the extension (`jerry`) which is passed to the mail delivery agent for local use.

The default value of the `ForwardPath` processing option in `sendmail(1M)` is:

```
0 ForwardPath=$z/.forward.$w+$h:$z/.forward+$h:$z/.forward.$w:$z \
/.forward
```

where `$z` is the macro for the user's home directory, `$w` is the macro for the local machine name and `$h` is the extension. For example, for mail using the address, `jsmith+jerry@jsmachine`, the `sendmail(1M)` utility checks each of the four following file names, in the order given, to see if it exists and if it has “safe” permissions, that is, that neither the file nor any of its parent directories are group- or world-writable:

```
~jsmith/.forward.jsmachine+jerry
~jsmith/.forward+jerry
~jsmith/.forward.jsmachine
~jsmith/.forward
```

The first file that meets the conditions is used to forward the mail, that is, all the entries in that file receive a copy of the mail. The search is then stopped.

Files	<code>/etc/passwd</code>	Password file
	<code>/etc/nsswitch.conf</code>	Name service switch configuration file
	<code>/etc/mail/aliases</code>	Mail aliases file (ascii)
	<code>/etc/mail/aliases.db</code>	Database of mail aliases (binary)
	<code>/etc/mail/aliases.dir</code>	Database of mail aliases (binary)
	<code>/etc/mail/aliases.pag</code>	Database of mail aliases (binary)
	<code>/etc/mail/sendmail.cf</code>	sendmail configuration file
	<code>~/ .forward</code>	Forwarding information file

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/smtp/sendmail

See Also [passwd\(1\)](#), [uucp\(1C\)](#), [vacation\(1\)](#), [newaliases\(1M\)](#), [sendmail\(1M\)](#), [ndbm\(3C\)](#), [getusershell\(3C\)](#), [passwd\(4\)](#), [shells\(4\)](#), [attributes\(5\)](#)

Notes Because of restrictions in [ndbm\(3C\)](#), a single alias cannot contain more than about 1000 characters (if this format is used). The Berkeley DataBase format does not have any such restriction. Nested aliases can be used to circumvent this limit.

For aliases which result in piping to a program or concatenating a file, the shell of the controlling user must be allowed. Which shells are and are not allowed are determined by [getusershell\(3C\)](#).

Name a.out – Executable and Linking Format (ELF) files

Synopsis `#include <elf.h>`

Description The file name `a.out` is the default output file name from the link editor, `ld(1)`. The link editor will make an `a.out` executable if there were no errors in linking. The output file of the assembler, `as(1)`, also follows the format of the `a.out` file although its default file name is different.

Programs that manipulate ELF files may use the library that `elf(3ELF)` describes. An overview of the file format follows. For more complete information, see the references given below.

Linking View	Execution View
ELF header	ELF header
Program header table <i>optional</i>	Program header table
Section 1	Segment 1
...	
Section <i>n</i>	Segment 2
...	
...	...
Section header table	Section header table <i>optional</i>

An ELF header resides at the beginning and holds a “road map” describing the file’s organization. Sections hold the bulk of object file information for the linking view: instructions, data, symbol table, relocation information, and so on. Segments hold the object file information for the program execution view. As shown, a segment may contain one or more sections.

A program header table, if present, tells the system how to create a process image. Files used to build a process image (execute a program) must have a program header table; relocatable files do not need one. A section header table contains information describing the file’s sections. Every section has an entry in the table; each entry gives information such as the section name, the section size, etc. Files used during linking must have a section header table; other object files may or may not have one.

Although the figure shows the program header table immediately after the ELF header, and the section header table following the sections, actual files may differ. Moreover, sections and segments have no specified order. Only the ELF header has a fixed position in the file.

When an `a.out` file is loaded into memory for execution, three logical segments are set up: the text segment, the data segment (initialized data followed by uninitialized, the latter actually being initialized to all 0's), and a stack. The text segment is not writable by the program; if other processes are executing the same `a.out` file, the processes will share a single text segment.

The data segment starts at the next maximal page boundary past the last text address. If the system supports more than one page size, the "maximal page" is the largest supported size. When the process image is created, the part of the file holding the end of text and the beginning of data may appear twice. The duplicated chunk of text that appears at the beginning of data is never executed; it is duplicated so that the operating system may bring in pieces of the file in multiples of the actual page size without having to realign the beginning of the data section to a page boundary. Therefore, the first data address is the sum of the next maximal page boundary past the end of text plus the remainder of the last text address divided by the maximal page size. If the last text address is a multiple of the maximal page size, no duplication is necessary. The stack is automatically extended as required. The data segment is extended as requested by the `brk(2)` system call.

See Also `as(1)`, `ld(1)`, `brk(2)`, `elf(3ELF)`

ANSI C Programmer's Guide

Name au – AU audio file format

Synopsis #include <audio/au.h>

Description An AU audio file is composed of three parts: a header, an optional description field, and a contiguous segment of audio data. The header is 24 bytes, and the description field is at least 4 bytes. Therefore, the offset for most AU files is 28 bytes. However, some people store additional data in the AU header.

The AU audio structure members and audio data are stored big endian. That is, it starts with the most significant byte, regardless of the native byte order of the machine architecture on which an application may be running. Therefore, multi-byte audio data may require byte reversal for proper playback on different processor architectures. See the macro section for properly reading and writing the AU audio structure members.

The AU header is defined by the following structure:

```
struct au_filehdr {
    uint32_t au_magic;      /* magic number (.snd) */
    uint32_t au_offset;    /* byte offset to start of audio data */
    uint32_t au_data_size; /* data length in bytes */
    uint32_t au_encoding;  /* data encoding */
    uint32_t au_sample_rate; /* samples per second */
    uint32_t au_channels;  /* number of interleaved channels */
};
typedef struct au_filehdr au_filehdr_t;
```

The `au_magic` field always contains the following constant for an AU audio file:

```
AUDIO_AU_FILE_MAGIC ( 0x2e736e64 ) /* ".snd" */
```

The `au_offset` field contains the length of the audio file header plus the variable length info field. Consequently, it can be interpreted as the offset from the start of the file to the start of the audio data.

The `au_data_size` field contains the length, in bytes, of the audio data segment. If this length is not known when the header is written, it should be set to `AUDIO_AU_UNKNOWN_SIZE`, defined as follows:

```
AUDIO_AU_UNKNOWN_SIZE ( ~0 ) /* (unsigned) -1 */
```

When the `au_data_size` field contains `AUDIO_AU_UNKNOWN_SIZE`, the length of the audio data can be determined by subtracting `au_offset` from the total length of the file.

The encoding field contains one of the following enumerated keys:

```
AUDIO_AU_ENCODING_ULAW      /* 8-bit u-law */
AUDIO_AU_ENCODING_LINEAR_8  /* 8-bit linear PCM */
AUDIO_AU_ENCODING_LINEAR_16 /* 16-bit linear PCM */
AUDIO_AU_ENCODING_LINEAR_24 /* 24-bit linear PCM */
```

```

AUDIO_AU_ENCODING_LINEAR_32    /* 32-bit linear PCM */
AUDIO_AU_ENCODING_FLOAT        /* Floating point */
AUDIO_AU_ENCODING_DOUBLE       /* Double precision float */
AUDIO_AU_ENCODING_FRAGMENTED   /* Fragmented sample data */
AUDIO_AU_ENCODING_DSP          /* DSP program */
AUDIO_AU_ENCODING_FIXED_8      /* 8-bit fixed point */
AUDIO_AU_ENCODING_FIXED_16     /* 16-bit fixed point */
AUDIO_AU_ENCODING_FIXED_24     /* 24-bit fixed point */
AUDIO_AU_ENCODING_FIXED_32     /* 32-bit fixed point */
AUDIO_AU_ENCODING_EMPHASIS     /* 16-bit linear with emphasis */
AUDIO_AU_ENCODING_COMPRESSED   /* 16-bit linear compressed */
AUDIO_AU_ENCODING_EMP_COMP     /* 16-bit linear with emphasis
                                and compression */

AUDIO_AU_ENCODING_MUSIC_KIT    /* Music kit DSP commands */
AUDIO_AU_ENCODING_ADPCM_G721   /* CCITT G.721 ADPCM */
AUDIO_AU_ENCODING_ADPCM_G722   /* CCITT G.722 ADPCM */
AUDIO_AU_ENCODING_ADPCM_G723_3 /* CCITT G.723.3 ADPCM */
AUDIO_AU_ENCODING_ADPCM_G723_5 /* CCITT G.723.5 ADPCM */
AUDIO_AU_ENCODING_ALAW         /* 8-bit A-law G.711 */

```

All of the linear encoding formats are signed integers centered at zero.

The `au_sample_rate` field contains the audio file's sampling rate in samples per second. Some common sample rates include 8000, 11025, 22050, 44100, and 48000 samples per second.

The `au_channels` field contains the number of interleaved data channels. For monaural data, this value is set to one. For stereo data, this value is set to two. More than two data channels can be interleaved, but such formats are currently unsupported by the Solaris audio driver architecture. For a stereo sound file, the first sample is the left track and the second sample is the right track.

The optional `info` field is a variable length annotation field that can be either text or data. If it is a text description of the sound, then it should be NULL terminated. However, some older files might not be terminated properly. The size of the `info` field is set when the structure is created and cannot be enlarged later.

Macros Accessing all of the AU audio structure members should be done through the supplied `AUDIO_AU_FILE2HOST` and `AUDIO_AU_HOST2FILE` macros. By always using these macros, code will be byte-order independent. See the example below.

Examples **EXAMPLE 1** Displaying Header Information for a Sound File

The following program reads and displays the header information for an AU sound file. The `AUDIO_AU_FILE2HOST` macro ensures that this information will always be in the proper byte order.

```

void main(void)
{
    au_filehdr_t    hdr;

```

EXAMPLE 1 Displaying Header Information for a Sound File *(Continued)*

```

au_filehdr_t    local;
int             fd;
char           *name = "bark.au";

if ((fd = open(name, O_RDONLY)) < 0) {
    printf("can't open file %s\n", name);
    exit(1);
}

(void) read(fd, &hdr, sizeof (hdr));

AUDIO_AU_FILE2HOST(&hdr.au_magic, &local.au_magic);
AUDIO_AU_FILE2HOST(&hdr.au_offset, &local.au_offset);
AUDIO_AU_FILE2HOST(&hdr.au_data_size, &local.au_data_size);
AUDIO_AU_FILE2HOST(&hdr.au_encoding, &local.au_encoding);
AUDIO_AU_FILE2HOST(&hdr.au_sample_rate, &local.au_sample_rate);
AUDIO_AU_FILE2HOST(&hdr.au_channels, &local.au_channels);

printf("Magic = %x\n", local.au_magic);
printf("Offset = %d\n", local.au_offset);
printf("Number of data bytes = %d\n", local.au_data_size);
printf("Sound format = %d\n", local.au_encoding);
printf("Sample rate = %d\n", local.au_sample_rate);
printf("Number of channels = %d\n", local.au_channels);

(void) close(fd);
}

```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/header
Interface Stability	Committed

See Also [attributes\(5\)](#)

Notes Some older AU audio files are incorrectly coded with info strings that are not properly NULL-terminated. Thus, applications should always use the `au_offset` value to find the end of the info data and the beginning of the audio data.

Name audit_class – audit class definitions

Synopsis /etc/security/audit_class

Description /etc/security/audit_class is an administrator configurable system file that stores class definitions used in the audit system. Audit events in [audit_event\(4\)](#) are mapped to one or more of the defined audit classes. `audit_event` can be updated in conjunction with changes to `audit_class`. See [auditconfig\(1M\)](#) and [user_attr\(4\)](#) for information about changing the preselection of audit classes in the audit system.

The fields for each class entry are separated by colons. Each class entry is a bitmap and is separated from each other by a NEWLINE.

Each entry in the `audit_class` file has the form:

```
mask:name:description
```

The fields are defined as follows:

```
mask          class mask
```

```
name          class name
```

```
description   class description
```

Each class is represented as a bit in the 64 bit class mask. There are 64 different classes available. Meta-classes can also be defined. Meta-classes are supersets composed of multiple base classes, and have more than 1 bit in `smask`. See [EXAMPLES](#).

Two special meta-classes are pre-defined: `all` and `no`.

`all` Represents a conjunction of all allowed classes, and is provided as a shorthand method of specifying all classes.

`no` Is the invalid class, and any event mapped solely to this class are not audited. Turning auditing on to the `all` meta-class does not cause events mapped solely to the `no` class to be written to the audit trail. This class is also used to map obsolete events which are no longer generated. Obsolete events are retained to process old audit trails files.

The mask positions `0xff00000000000000` are reserved for local site use.

Examples [EXAMPLE 1](#) Using an `audit_class` File

The following is an example of an `audit_class` file:

```
0x0000000000000000:0:no:invalid class
0x0000000000000001:1:fr:file read
0x0000000000000002:2:fw:file write
0x0000000000000004:4:fa:file attribute access
0x0000000000000008:8:fm:file attribute modify
0x0000000000000010:10:fc:file create
```

EXAMPLE 1 Using an audit_class File (Continued)

```

0x0000000000000020:fd:file delete
0x0000000000000040:cl:file close
0x0000000000000080:ft:file transfer
0x0000000000000100:nt:network
0x0000000000000200:ip:ipc
0x0000000000000400:na:non-attribute
0x0000000000001000:lo:login or logout
0x0000000000004000:ap:application
0x000000000000f000:ad:old administrative (meta-class)
0x0000000000070000:am:administrative (meta-class)
0x0000000000010000:ss:change system state
0x0000000000020000:as:system-wide administration
0x0000000000040000:ua:user administration
0x0000000000080000:aa:audit utilization
0x0000000000300000:pc:process (meta-class)
0x0000000000100000:ps:process start/stop
0x0000000000200000:pm:process modify
0x0000000000200000:io:ioctl
0x0000000000400000:ex:exec
0x0000000000800000:ot:other
0xfffffffffffffff:all:all classes (meta-class)

```

Files /etc/security/audit_class

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	See below.

The file format stability is Committed. The file content is Uncommitted.

See Also [auditconfig\(1M\)](#), [audit_event\(4\)](#), [user_attr\(4\)](#), [attributes\(5\)](#)

Part VII, “Auditing in Oracle Solaris,” in *Oracle Solaris Administration: Security Services*

Notes Redefining the no class to have a non - zero value can have undesirable side effects.

Name audit_event – audit event definition and class mapping

Synopsis /etc/security/audit_event

Description /etc/security/audit_event is a user-configurable ASCII system file that stores event definitions used in the audit system. As part of this definition, each event is mapped to one or more of the audit classes defined in [audit_class\(4\)](#). See [auditconfig\(1M\)](#) and [user_attr\(4\)](#) for information about changing the preselection of audit classes in the audit system.

The fields for each event entry are separated by colons. Each event is separated from the next by a NEWLINE. Each entry in the audit_event file has the form:

number:*name*:*description*:*flags*

The fields are defined as follows:

number Event number.

Event number ranges are assigned as follows:

0 Reserved as an invalid event number.

1-2047 Reserved for the Solaris Kernel events.

2048-32767 Reserved for the Solaris TCB programs.

32768-65535 Available for third party TCB applications.

System administrators must *not* add, delete, or modify (except to change the class mapping), events with an event number less than 32768. These events are reserved by the system.

name Event name.

description Event description.

flags Flags specifying classes to which the event is mapped. Classes are comma separated, without spaces.

Obsolete events are commonly assigned to the special class no (invalid) to indicate they are no longer generated. Obsolete events are retained to process old audit trail files. Other events which are not obsolete may also be assigned to the no class.

Examples EXAMPLE 1 Using the audit_event File

The following is an example of some audit_event file entries:

```
7:AUE_EXEC:exec(2):ps,ex
79:AUE_OPEN_WTC:open(2) - write,creat,trunc:fc,fd,fw
6152:AUE_login:login - local:lo
```

EXAMPLE 1 Using the audit_event File (Continued)

```
6153:AUE_logout:logout:lo
6154:AUE_telnet:login - telnet:lo
6155:AUE_rlogin:login - rlogin:lo
```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	See below.

The file format stability is Committed. The file content is Uncommitted.

Files /etc/security/audit_event

See Also [auditconfig\(1M\)](#), [audit_class\(4\)](#), [user_attr\(4\)](#)

Part VII, “Auditing in Oracle Solaris,” in *Oracle Solaris Administration: Security Services*

Notes This functionality is available only if Solaris Auditing has been enabled.

Name audit.log – audit trail file

Synopsis #include <bsm/audit.h>
#include <bsm/audit_record.h>

Description audit.log files are the depository for audit records stored locally or on an NFS-mounted audit server. These files are kept in directories named in the *p_dir* property of the audit service [audit_binfile\(5\)](#) plugin. They are named to reflect the time they are created and are, when possible, renamed to reflect the time they are closed as well. The name takes the form

yyyymmddhhmmss.not_terminated.hostname

when open or if the [auditd\(1M\)](#) terminated ungracefully, and the form

yyyymmddhhmmss.yyyyymmddhhmmss.hostname

when properly closed. *yyyy* is the year, *mm* the month, *dd* day in the month, *hh* hour in the day, *mm* minute in the hour, and *ss* second in the minute. All fields are of fixed width.

Audit data is generated in the binary format described below; the default for Solaris audit is binary format. See [audit_syslog\(5\)](#) for an alternate data format.

The audit.log file begins with a standalone *file* token and typically ends with one also. The beginning *file* token records the pathname of the previous audit file, while the ending *file* token records the pathname of the next audit file. If the file name is NULL the appropriate path was unavailable.

The audit.log files contains audit records. Each audit record is made up of *audit tokens*. Each record contains a header token followed by various data tokens. Depending on the audit policy in place by [auditconfig\(1M\)](#), optional other tokens such as trailers or sequences may be included.

The tokens are defined as follows:

The *file* token consists of:

token ID	1 byte
seconds of time	4 bytes
microseconds of time	4 bytes
file name length	2 bytes
file pathname	N bytes + 1 terminating NULL byte

The header token consists of:

token ID	1 byte
record byte count	4 bytes
version #	1 byte [2]
event type	2 bytes
event modifier	2 bytes

seconds of time	4 bytes/8 bytes (32-bit/64-bit value)
nanoseconds of time	4 bytes/8 bytes (32-bit/64-bit value)

The expanded header token consists of:

token ID	1 byte
record byte count	4 bytes
version #	1 byte [2]
event type	2 bytes
event modifier	2 bytes
address type/length	4 bytes
machine address	4 bytes/16 bytes (IPv4/IPv6 address)
seconds of time	4 bytes/8 bytes (32/64-bits)
nanoseconds of time	4 bytes/8 bytes (32/64-bits)

The trailer token consists of:

token ID	1 byte
trailer magic number	2 bytes
record byte count	4 bytes

The arbitrary data token is defined:

token ID	1 byte
how to print	1 byte
basic unit	1 byte
unit count	1 byte
data items	(depends on basic unit)

The in_addr token consists of:

token ID	1 byte
IP address	4 bytes (IPv4 address)

The expanded in_addr token consists of:

token ID	1 byte
IP address type/length	4 bytes
IP address	16 bytes (IPv6 address)

The ip token consists of:

token ID	1 byte
version and ihl	1 byte
type of service	1 byte
length	2 bytes
id	2 bytes
offset	2 bytes
ttl	1 byte
protocol	1 byte
checksum	2 bytes
source address	4 bytes
destination address	4 bytes

The expanded ip token consists of:

token ID	1 byte
version and ihl	1 byte
type of service	1 byte
length	2 bytes
id	2 bytes
offset	2 bytes
ttl	1 byte
protocol	1 byte
checksum	2 bytes
address type/type	1 byte
source address	4 bytes/16 bytes (IPv4/IPv6 address)
address type/length	1 byte
destination address	4 bytes/16 bytes (IPv4/IPv6 address)

The iport token consists of:

token ID	1 byte
port IP address	2 bytes

The path token consists of:

token ID	1 byte
path length	2 bytes
path	N bytes + 1 terminating NULL byte

The path_attr token consists of:

token ID	1 byte
count	4 bytes
path	<i>count</i> null-terminated string(s)

The process token consists of:

token ID	1 byte
audit ID	4 bytes
effective user ID	4 bytes
effective group ID	4 bytes
real user ID	4 bytes
real group ID	4 bytes
process ID	4 bytes
session ID	4 bytes
terminal ID	
port ID	4 bytes/8 bytes (32-bit/64-bit value)
machine address	4 bytes (IPv4 address)

The expanded process token consists of:

token ID	1 byte
audit ID	4 bytes
effective user ID	4 bytes

effective group ID	4 bytes
real user ID	4 bytes
real group ID	4 bytes
process ID	4 bytes
session ID	4 bytes
terminal ID	
port ID	4 bytes/8 bytes (32-bit/64-bit value)
address type/length	4 bytes
machine address	16 bytes (IPv6 address)

The return token consists of:

token ID	1 byte
error number	1 byte
return value	4 bytes/8 bytes (32-bit/64-bit value)

The subject token consists of:

token ID	1 byte
audit ID	4 bytes
effective user ID	4 bytes
effective group ID	4 bytes
real user ID	4 bytes
real group ID	4 bytes
process ID	4 bytes
session ID	4 bytes
terminal ID	
port ID	4 bytes/8 bytes (32-bit/64-bit value)
machine address	4 bytes (IPv4 address)

The expanded subject token consists of:

token ID	1 byte
audit ID	4 bytes
effective user ID	4 bytes
effective group ID	4 bytes
real user ID	4 bytes
real group ID	4 bytes
process ID	4 bytes
session ID	4 bytes
terminal ID	
port ID	4 bytes/8 bytes (32-bit/64-bit value)
address type/length	4 byte
machine address	16 bytes (IPv6 address)

The System V IPC token consists of:

token ID	1 byte
object ID type	1 byte
object ID	4 bytes

The text token consists of:

token ID	1 byte
text length	2 bytes
text	N bytes + 1 terminating NULL byte

The attribute token consists of:

token ID	1 byte
file access mode	4 bytes
owner user ID	4 bytes
owner group ID	4 bytes
file system ID	4 bytes
node ID	8 bytes
device	4 bytes/8 bytes (32-bit/64-bit)

The groups token consists of:

token ID	1 byte
number groups	2 bytes
group list	N * 4 bytes

The System V IPC permission token consists of:

token ID	1 byte
owner user ID	4 bytes
owner group ID	4 bytes
creator user ID	4 bytes
creator group ID	4 bytes
access mode	4 bytes
slot sequence #	4 bytes
key	4 bytes

The arg token consists of:

token ID	1 byte
argument #	1 byte
argument value	4 bytes/8 bytes (32-bit/64-bit value)
text length	2 bytes
text	N bytes + 1 terminating NULL byte

The exec_args token consists of:

token ID	1 byte
count	4 bytes
text	<i>count</i> null-terminated string(s)

The exec_env token consists of:

token ID	1 byte
count	4 bytes
text	<i>count</i> null-terminated string(s)

The exit token consists of:

token ID	1 byte
status	4 bytes
return value	4 bytes

The socket token consists of:

token ID	1 byte
socket type	2 bytes
remote port	2 bytes
remote Internet address	4 bytes

The expanded socket token consists of:

token ID	1 byte
socket domain	2 bytes
socket type	2 bytes
local port	2 bytes
local Internet address	4 bytes/16 bytes (IPv4/IPv6 address)
remote port	2 bytes
remote Internet address	4 bytes/16 bytes (IPv4/IPv6 address)

The seq token consists of:

token ID	1 byte
sequence number	4 bytes

The privilege token consists of:

token ID	1 byte
text length	2 bytes
privilege set name	N bytes + 1 terminating NULL byte
text length	2 bytes
list of privileges	N bytes + 1 terminating NULL byte

The user token consists of:

token ID	1 byte
user ID	4 bytes
user name length	2 bytes
user name	<user name len> including terminating NULL byte

The use-of-auth token consists of:

token ID	1 byte
text length	2 bytes
authorization(s)	N bytes + 1 terminating NULL byte

The use-of-privilege token consists of:

token ID	1 byte
succ/fail	1 byte

text length	2 bytes
privilege used	N bytes + 1 terminating NULL byte

The command token consists of:

token ID	1 byte
count of args	2 bytes
argument list	(count times)
text length	2 bytes
argument text	N bytes + 1 terminating NULL byte
count of env strings	2 bytes
environment list	(count times)
text length	2 bytes
env. text	N bytes + 1 terminating NULL byte

The ACL token consists of:

token ID	1 byte
type	4 bytes
value	4 bytes
file mode	4 bytes

The ACE token consists of:

token ID	1 byte
who	4 bytes
access_mask	4 bytes
flags	2 bytes
type	2 bytes

The zonename token consists of:

token ID	1 byte
name length	2 bytes
name	<name length> including terminating NULL byte

The fmri token consists of:

token ID	1 byte
fmri length	2 bytes
fmri	<fmri length> including terminating NULL byte

The label token consists of:

token ID	1 byte
label ID	1 byte
compartment length	1 byte
classification	2 bytes
compartment words	<compartment length> * 4 bytes

The xatom token consists of:

token ID	1 byte
string length	2 bytes
atom string	<i>string length</i> bytes

The xclient token consists of:

token ID	1 byte
client ID	4 bytes

The xcolormap token consists of:

token ID	1 byte
XID	4 bytes
creator UID	4 bytes

The xcursor token consists of:

token ID	1 byte
XID	4 bytes
creator UID	4 bytes

The xfont token consists of:

token ID	1 byte
XID	4 bytes
creator UID	4 bytes

The xgc token consists of:

token ID	1 byte
XID	4 bytes
creator UID	4 bytes

The xpixmap token consists of:

token ID	1 byte
XID	4 bytes
creator UID	4 bytes

The xproperty token consists of:

token ID	1 byte
XID	4 bytes
creator UID	4 bytes
string length	2 bytes
string	<i>string length</i> bytes

The xselect token consists of:

token ID	1 byte
property length	2 bytes
property string	<i>property length</i> bytes
prop. type len.	2 bytes

prop type	<i>prop. type len.</i> bytes
data length	2 bytes
window data	<i>data length</i> bytes

The xwindow token consists of:

token ID	1 byte
XID	4 bytes
creator UID	4 bytes

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	See below.

The binary file format is Committed. The binary file contents is Uncommitted.

See Also [audit\(1M\)](#), [auditconfig\(1M\)](#), [auditd\(1M\)](#), [audit_binfile\(5\)](#), [audit_syslog\(5\)](#)

Part VII, “Auditing in Oracle Solaris,” in *Oracle Solaris Administration: Security Services*

Name auth_attr – authorization description database

Synopsis /etc/security/auth_attr

Description /etc/security/auth_attr is a local source for authorization names and descriptions. The auth_attr file can be used with other authorization sources, including the auth_attr NIS map. Programs use the `getauthattr(3C)` routines to access this information.

The search order for multiple authorization sources is specified in the `/etc/nsswitch.conf` file, as described in the `nsswitch.conf(4)` man page.

An authorization is a right assigned to users that is checked by certain privileged programs to determine whether users can execute restricted functionality. Each entry in the auth_attr database consists of one line of text containing six fields separated by colons (:). Line continuations using the backslash (\) character are permitted. The format of each entry is:

name:res1:res2:short_desc:long_desc:attr

name The name of the authorization. Authorization names are unique strings. Construct authorization names using the following convention:

prefix. or *prefix.suffix*

prefix Everything in the name field up to the final dot (.). Authorizations from Sun Microsystems, Inc. use `solaris` as a prefix. To avoid name conflicts, all other authorizations should use a prefix that begins with the reverse-order Internet domain name of the organization that creates the authorization (for example, `com.xyzcompany`). Prefixes can have additional arbitrary components chosen by the authorization's developer, with components separated by dots.

suffix The final component in the name field. Specifies what is being authorized.

When there is no suffix, the name is defined as a heading. Headings are not assigned to users but are constructed for use by applications in their GUIs.

To assign an authorization, the user needs to have both the authorization itself and the `solaris.auth.delegate` authorization.

res1 The characters `RO` in this field indicate it is read only and not modifiable by the tools that update this database.

res2 Reserved for future use.

short_desc A short description or terse name for the authorization. This name should be suitable for displaying in user interfaces, such as in a scrolling list in a GUI.

<i>long_desc</i>	A long description. This field can explain the precise purpose of the authorization, the applications in which it is used, and the type of user that would be interested in using it. The long description can be displayed in the help text of an application.
<i>attr</i>	An optional list of semicolon-separated (;) key-value pairs that describe the attributes of an authorization. Zero or more keys can be specified. The keyword <code>help</code> identifies a help file in HTML.

Examples EXAMPLE 1 Constructing a Name

In the following example, the name has a prefix (`solaris.admin.usermgr`) followed by a suffix (`read`):

```
solaris.admin.usermgr.read
```

EXAMPLE 2 Defining a Heading

Because the name field ends with a dot, the following entry defines a heading:

```
solaris.user.:::User Accounts::help=AuthUsermgrHeader.html
```

EXAMPLE 3 Assigning Separate Authorizations to Set User Attributes

In this example, a heading entry is followed by other associated authorization entries. The entries below the heading provide separate authorizations for setting user attributes. The *attr* field for each entry, including the heading entry, assigns a help file. The application that uses the `help` key requires the value to equal the name of a file ending in `.htm` or `.html`:

```
solaris.role.:::Role Accounts::help=AuthRoleManageHeader.html
solaris.role.manage:::Manage Role Accounts::help=AuthRoleManage.html
solaris.role.delegate:::Delegate Role Accounts::help=AuthRoleDelegate.html
```

EXAMPLE 4 Assigning a Grant Authorization

This example assigns to an administrator the following authorizations:

```
solaris.auth.delegate
solaris.user.manage
solaris.role.manage
```

With the above authorizations, the administrator can assign to others the `solaris.user.manage` and `solaris.role.manage` authorizations, but not any other authorization. If the administrator has both the `solaris.auth.delegate` and the `solaris.*.manage` authorization, it would be equivalent to the above authorizations. See [user_attr\(4\)](#) for more information about how wildcards can be used to assign multiple authorizations whose names have the same components.

EXAMPLE 5 Authorizing the Ability to Assign Other Authorizations

The following entry defines an authorization that grants the ability to assign any authorization.

```
solaris.auth.assign::Grant All Solaris Authorizations::help=PriAdmin.html
```

EXAMPLE 6 Consulting the Local Authorization File Ahead of the NIS Table

With the following entry from `/etc/nsswitch.conf`, the local `auth_attr` file is consulted before the LDAP name service:

```
auth_attr:files ldap
```

Files `/etc/nsswitch.conf`

`/etc/user_attr`

`/etc/security/auth_attr` Locally added entries. Make sure that the shipped header remains intact.

`/etc/security/auth_attr.d/*` Entries added by package installation.

See Also [auths\(1\)](#), [getauthattr\(3C\)](#), [getexecattr\(3C\)](#), [getprofattr\(3C\)](#), [getuserattr\(3C\)](#), [exec_attr\(4\)](#), [nsswitch.conf\(4\)](#), [user_attr\(4\)](#)

Notes Because the list of legal keys is likely to expand, any code that parses this database must be written to ignore unknown key-value pairs without error. When any new keywords are created, the names should be prefixed with a unique string, such as the company's stock symbol, to avoid potential naming conflicts.

Each application has its own requirements for whether the help value must be a relative pathname ending with a filename or the name of a file. The only known requirement is for the name of a file.

The following characters are used in describing the database format and must be escaped with a backslash if used as data: colon (:), semicolon (;), equals (=), and backslash (\).

The authorization required to add/modify/delete authorizations is `solaris.auth.manage`. The `solaris.auth.assign` authorization allows an authorized user to grant any authorization to a user. The `solaris.auth.delegate` allows an authorized user to grant only the user's authorizations to another user.

- Name** autofs – file containing parameter values for automountd daemon and automount command
- Synopsis** /etc/default/autofs
- Description** The settings formerly managed by the autofs file have been moved to SMF properties and are now managed by [sharectl\(1M\)](#). The functionality described here is provided for backward compatibility only.

The autofs file resides in directory /etc/default and supplies default parameters for the [automountd\(1M\)](#) daemon and the [automount\(1M\)](#) command.

The autofs file format is ASCII; comment lines begin with the crosshatch (#) character. Parameters consist of a keyword followed by an equal sign followed by the parameter value, of the form:

keyword=value

As shipped, the parameters in the autofs file are commented out. As root, you must uncomment a keyword-value line to make the value for that parameter take effect.

Administrators can make changes to the startup parameters for automountd by logging in as root and editing the autofs file. Changes made to autofs values on an automount or automountd command line override values in /etc/default/autofs. The /etc/default/autofs file is preserved across operating system upgrades.

The following parameters are currently supported in the autofs file:

AUTOMOUNT_TIMEOUT=<num>	Specifies a duration, in seconds, that a file system is to remain mounted when not in use. The default value is 600 (10 minutes). Equivalent to the -t option in automount.
AUTOMOUNT_VERBOSE=TRUE FALSE	Verbose mode. Causes you to be notified of non-critical events, such as autofs mounts and unmounts. The default value is FALSE. Equivalent to the -v option in automount.
AUTOMOUNTD_VERBOSE=TRUE FALSE	Verbose mode. Causes status messages to be logged to /var/svc/log/system-filesystem-autofs:default.log. (See smf(5) .) The default value is FALSE. Equivalent to the -v option in automountd.
AUTOMOUNTD_NOBROWSE=<num>	Turn on or off browsing for all autofs mount points. The default value is FALSE. Equivalent to the -n option in automountd.
AUTOMOUNTD_TRACE=<num>	Expands each RPC call and logs it to /var/svc/log/system-filesystem-autofs:default.log.

(See [smf\(5\)](#).) The default value, 0, turns off such tracing. Starting with 1, with each higher value, the verbosity of trace output increases.

`AUTOMOUNTD_ENV=<name>=<value>`

Environment variables. Each environment variable-value pairing must be on its own line. You can specify multiple such pairings. There are no environment variable settings supplied. For example: `AUTOMOUNTD_ENV=DAY=TUES`

See Also [automount\(1M\)](#), [automountd\(1M\)](#), [sharectl\(1M\)](#), [smf\(5\)](#)

Name bart_manifest – system audit manifest file

Description The `bart(1M)` command generates a manifest that describes the contents of a managed host. A manifest consists of a header and entries. Each entry represents a single file. Entries are sorted in ascending order by file name. Any nonstandard file names, such as those that contain embedded newline or tab characters, have the special characters quoted prior to being sorted. See [Quoting Syntax](#).

Lines that begin with `!` supply metadata about the manifest. The manifest version line indicates the manifest specification version. The date line shows the date on which the manifest was created, in [date\(1\)](#) form.

Some lines are ignored by the manifest comparison tool. Ignored lines include blank lines, lines that consist only of white space, and comments that begin with `#`.

In addition to metadata lines, the header contains the format comment block. This comment block lists the attributes reported for each file type.

To see the format of a manifest file, see [EXAMPLES](#).

Manifest File Entries Each manifest file entry is a single line of one of the following forms, depending on the file type:

```
fname D size mode acl dirmtime uid gid
fname P size mode acl mtime uid gid
fname S size mode acl mtime uid gid
fname F size mode acl mtime uid gid contents
fname L size mode acl lnmtime uid gid dest
fname B size mode acl mtime uid gid devnode
fname C size mode acl mtime uid gid devnode
```

The fields of the manifest file entries are described as follows:

fname Name of the file. To prevent parsing problems that are caused by special characters embedded in file names, file names are encoded as described in [Quoting Syntax](#).

type Type of file.

Possible values for *type* are as follows:

- B Block device node
- C Character device node
- D Directory
- F File
- L Symbolic link

	P	Pipe
	S	Socket
<i>size</i>		File size in bytes.
<i>mode</i>		Octal number that represents the permissions of the file.
<i>acl</i>		ACL attributes for the file. For a file with ACL attributes, this field contains the output from <code>acltotext()</code> .
<i>uid</i>		Numerical user ID of the owner of this entry.
<i>gid</i>		Numerical group ID of the owner of this entry.
<i>dirmtime</i>		Modification time in seconds since 00:00:00 UTC, January 1, 1970 for directories.
<i>lnmtime</i>		Creation time for links.
<i>mtime</i>		Modification time in seconds since 00:00:00 UTC, January 1, 1970 for files.
<i>contents</i>		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 <code>-</code> .
<i>dest</i>		Destination of a symbolic link.
<i>devnode</i>		Value of the device node. This attribute is for character device files and block device files only.

Quoting Syntax The rules file supports a quoting syntax for representing nonstandard file names.

When generating a manifest for file names that embed TAB, SPACE, or NEWLINE characters, the special characters are encoded in their octal forms.

Input Character	Quoted Character
SPACE	\SPACE
TAB	\TAB
NEWLINE	\NEWLINE
?	\?
[\[
*	*

Examples EXAMPLE 1 Sample Manifest File

The following is a sample system manifest file. The file entries are sorted by the encoded versions of the file names to correctly handle special characters.

```
! Version 1.0
! Mon Feb 11 10:55:30 2002
# Format:
# fname D size mode acl dirmtime uid gid
# fname P size mode acl mtime uid gid
# fname S size mode acl mtime uid gid
# fname F size mode acl mtime uid gid contents
# fname L size mode acl lnmtime uid gid dest
# fname B size mode acl mtime uid gid devnode
# fname C size mode acl mtime uid gid devnode
/etc D 3584 40755 user::rwx,group::r-x,mask::r-x,other::r-x,
    3c6803d7 0 3
/etc/.login F 524 100644 user::rw-,group::r--,mask::r--,other::r--,
    3c165878 0 3 27b53d5c3e844af3306f1f12b330b318
/etc/.pwd.lock F 0 100600 user::rw-,group::---,mask::---,other::---,
    3c166121 0 0 d41d8cd98f00b204e9800998ecf8427e
/etc/.syslog_door L 20 120777 user::rw-,group::r--,mask::
    rwx,other::r--,3c6803d5 0 0 /var/run/syslog_door
/etc/cron.d/FIFO P 0 10600 user::rw-,group::---,mask::---,other::---,
    3c6803d5 0 0
```

See Also [date\(1\)](#), [bart\(1M\)](#), [bart_rules\(4\)](#), [attributes\(5\)](#)

Name bart_rules – bart rules file

Description The `bart_rules` file is a text file that is used by the `bart(1M)` command. The rules file determines which files to validate and which file attributes of those files to ignore.

Some lines are ignored by the manifest comparison tool. Ignored lines include blank lines, lines that consist only of white space, and comments that begin with `#`.

The rules file supports three directives: `CHECK`, `IGNORE`, and a *subtree* directive, which is an absolute path name and optional pattern matching modifiers. Each `CHECK`, `IGNORE`, and *subtree* directive must be on a separate line. Bart supports continuation of long lines using a backslash (`\`). The rules file uses the directives to create logical blocks.

Syntax The syntax for the rules file is as follows:

```
[IGNORE attribute...]*
[CHECK] [attribute...]*

subtree1 [pattern...]*
[IGNORE attribute...]*
[CHECK] [attribute...]*

subtree2 [pattern...]*
subtree3 [pattern...]*
subtree4 [pattern...]*
[IGNORE attribute...]*
[CHECK] [attribute...]*
...
```

Rule Blocks Rule blocks are composed of statements that are created by using directives and arguments.

There are three types of blocks:

Global Block The first block in the file. The block is considered “global” if it specifies `CHECK` and `IGNORE` statements, but no previous *subtree* statement. A global block pertains to all subsequent blocks.

Local block A block that specifies `CHECK` and `IGNORE` statements as well as a *subtree* directive. The rules in this block pertain to files and directories found in the specified subtree.

Heir block A block that contains a null `CHECK` statement, no arguments. This block inherits the global `CHECK` statements and `IGNORE` statements.

The order in which `CHECK` and `IGNORE` statements appear in blocks is important. The `bart` command processes `CHECK` and `IGNORE` statements in the order in which they are read, with later statements overriding earlier statements.

Subtree specifications must appear one per line. Each specification must begin with an absolute path name. Optionally, each specification can be followed by pattern-matching arguments.

When a file system being tracked belongs to more than one subtree directive, `bart` performs the following resolution steps:

- Applies the CHECK and IGNORE statements set in the global block. Note that all CHECK and IGNORE statements are processed in order.
- Finds the last subtree directive that matches the file.
- Processes the CHECK and IGNORE statements that belong to the last matching subtree directive. These statements are processed in the order in which they are read, overriding global settings.

Pattern Matching Statements

There are two types of pattern matching statements

- AND For a given subtree directive, all pattern matching statements are logically ANDed with the subtree. Patterns have the following syntax:
- Wildcards are permitted for both the subtree and pattern matching statements.
 - The exclamation point (!) character represents logical NOT.
 - A pattern that terminates with a slash is a subtree. The absence of a slash indicates that the pattern is not a directory. The subtree itself does not require an end slash.

For example, the following subtree example includes the contents of `/home/nickiso/src` except for object files, core files, and all of the SCCS subtrees. Note that directory names that terminate with `.o` and directories named `core` are *not* excluded because the patterns specified do not terminate with `.`

```
/home/nickiso/src !*.o !core !SCCS/
CHECK all
```

- OR Group multiple subtree directives together. Such subtree directives are logically ORed together.

```
/home/nickiso/src !*.o !core
/home/nickiso/Mail
/home/nickiso/docs *.sdw
CHECK all
IGNORE mtime lnmtime dirmtime
```

The files included in the previous example are as follows:

- Everything under `/home/nickiso/src` except for `*.o` and `core` files
- Everything under `/home/nickiso/Mail`
- All files under `/home/nickiso/docs` that end in `*.sdw`

For these files, all attributes are checked except for modification times.

File Attributes The `bart` command uses `CHECK` and `IGNORE` statements to define which attributes to track or ignore. Each attribute has an associated keyword.

The attribute keywords are as follows:

<code>acl</code>	ACL attributes for the file. For a file with ACL attributes, this field contains the output from <code>acltotext()</code> .
<code>all</code>	All attributes.
<code>contents</code>	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 <code>-</code> .
<code>dest</code>	Destination of a symbolic link.
<code>devnode</code>	Value of the device node. This attribute is for character device files and block device files only.
<code>dirmtime</code>	Modification time in seconds since 00:00:00 UTC, January 1, 1970 for directories.
<code>gid</code>	Numerical group ID of the owner of this entry.
<code>lnmtime</code>	Creation time for links.
<code>mode</code>	Octal number that represents the permissions of the file.
<code>mtime</code>	Modification time in seconds since 00:00:00 UTC, January 1, 1970 for files.
<code>size</code>	File size in bytes.
<code>type</code>	Type of file.
<code>uid</code>	Numerical user ID of the owner of this entry.

Examples **EXAMPLE 1** Sample Rules File

The following is a sample rules file:

```
# Global rules, track everything except dirmtime.
CHECK  all
IGNORE dirmtime

# The files in /data* are expected to change, so don't bother
# tracking the attributes expected to change.
# Furthermore, by specifying "IGNORE contents," you save
# time and resources.
/data*
IGNORE contents mtime size
```


EXAMPLE 1 Sample Rules File (Continued)

```
/home/nickiso f* bar/
IGNORE acl

# For /usr, apply the global rules.
/usr
CHECK

# Note: Since /usr/tmp follows the /usr block, the /usr/tmp
# subtree is subjected to the "IGNORE all."
/usr/tmp
/home/nickiso *.o
/home/nickiso core
/home/nickiso/proto
IGNORE all
```

The following files are cataloged based on the sample rules file:

- All attributes, except for `dirmtime`, `mtime`, `size`, and `contents`, are tracked for files under the `/data*` subtrees.
- Files under the `/usr` subtree, except for `/usr/tmp`, are cataloged by using the global rules.
- If the `/home/nickiso/foo.c` file exists, its attributes, except for `acl` and `dirmtime`, are cataloged.
- All `.o` and `core` files under `/home/nickiso`, as well as the `/home/nickiso/proto` and `/usr/tmp` subtrees, are ignored.
- If the `/home/nickiso/bar/foo.o` file exists, it is ignored because it is subject to the last block.

See Also [bart\(1M\)](#), [bart_manifest\(4\)](#), [attributes\(5\)](#)

Name bootparams – boot parameter data base

Synopsis /etc/bootparams

Description The bootparams file contains a list of client entries that diskless clients use for booting. Diskless booting clients retrieve this information by issuing requests to a server running the `rpc.bootparamd(1M)` program. The bootparams file may be used in conjunction with or in place of other sources for the bootparams information. See `nsswitch.conf(4)`.

For each client the file contains an entry with the client's name and a list of boot parameter values for that client. Each entry has the form:

```
clientname keyword=value...
```

The first item of each entry is the host name of the diskless client. You can use the asterisk (*) character as a wildcard in place of the client name in a single entry. A wildcard entry applies to all clients for which there is not an entry that specifically names them.

In a given entry, the host name or asterisk is followed by one or more whitespace characters and a series of keyword—value pairs separated by whitespace characters. There must not be any whitespace within a keyword—value pair.

Each keyword—value pair has the syntax:

```
keyword=value
```

The preceding form breaks out further as:

```
keyword=server:value
```

Where *server* can be null and *value* can be a pathname.

An example that includes a server is:

```
client1 root=server1:/export/client1/root
```

An example where *server* is null is:

```
client1 rootopts=:vers2
```

A minor variation of the *keyword=value* syntax is used for the domain keyword. Unlike the forms shown above, this syntax does not use a colon. For example:

```
client1 domain=bldg1.workco.com
```

Entries can span multiple lines. Use the backslash (\) character as the last character of a line to continue the entry to the following line. For multiple-line entries, you can split a line only in places where whitespace is allowed. For example, you can use a backslash to split the following entry between the end of the path (`root`) and the keyword `domain`:

```
client1 root=server1:/export/client1/root domain=bldg1.workco.com
```

In entries that specify a server, *server* is the name of the server that will provide the file or filesystem to the diskless client and *value* is the pathname of the exported file or filesystem on that server.

In entries that use the `domain` keyword, the domain name specified must be the client's domain name. The algorithm for determining a client's domain name is to first check for a `domain` keyword in the client-specific entry and then in wildcard entry. If none is found, the server's domain name is used.

An entry with the `ns` keyword associates a server (a name server) with, instead of a pathname, a specific name service (NIS, LDAP, or none) and, if that server is not on a local subnet, the netmask needed to reach it. For example:

```
ns=hoot:nis(255.255.255.0)
```

An `ns` entry forces installation software to use the specified name service. By default, NIS is used in preference to LDAP if it can find an NIS server for the system's domain on the subnet. An `ns` entry might be necessary if you are trying to set up a hands-off installation.

If an `ns` keyword is not used, installation software uses broadcast to attempt to bind to either an NIS or LDAP server. If a name server is not on the local subnet, which is possible for LDAP, the bind will fail, automatic configuration of the name service will fail, and an interactive screen is displayed, prompting the user to specify the name service.

The `ns` keyword can be set by Host Manager.

Examples **EXAMPLE 1** Sample bootparams Entry

Here is an example of an entry in the bootparams file:

```
client1 root=server1:/export/client1/root rootopts=:vers=2 \
    domain=bldg1.workco.com
client2 root=server2:/export/client2/root ns=:nis
client3 root=server2:/export/client3/root ns=watson:
client4 root=server2:/export/client4/root \
    ns=mach:nis(255.255.255.0)
```

Files /etc/bootparams

See Also [rpc.bootparamd\(1M\)](#), [nsswitch.conf\(4\)](#)

Notes Solaris diskless clients use the keywords `root` and `rootopts` to look up the pathname for the root filesystem and the mount options for the root filesystem, respectively. These are the only keywords meaningful for diskless booting clients. See [mount_ufs\(1M\)](#).

Name cardbus – configuration files for cardbus device drivers

Description The CardBus bus share the same configuration parameters with the PCI bus. CardBus devices are self-identifying, which means that these devices provide configuration parameters to the system that allow the system to identify the device and its driver. The configuration parameters are represented in the form of name-value pairs that can be retrieved using the DDI property interfaces. See [ddi_prop_lookup\(9F\)](#) for details.

The CardBus bus properties of CardBus devices are derived from PCI configuration space. Therefore, driver configuration files are not necessary for these devices.

On some occasions, drivers for CardBus devices can use driver configuration files to provide driver private properties through the global property mechanism. See [driver.conf\(4\)](#) for further details. Driver configuration files can also be used to augment or override properties for a specific instance of a driver.

The CardBus nexus driver recognizes the following properties:

reg An arbitrary length array where each element of the array consists of a 5-tuple of 32-bit values. Each array element describes a logically contiguous mappable resource on the PCI bus.

The first three values in the 5-tuple describe the PCI address of the mappable resource. The first tuple contains the following information:

Bits 0 - 7	8-bit register number
Bits 8 - 10	3-bit function number
Bits 11 - 15	5-bit device number
Bits 16 - 23	8-bit bus number
Bits 24 - 25	2-bit address space type identifier
Bits 31 - 28	Register number extended bits 8:11 for extended config space. Zero for conventional configuration space.

The address space type identifier can be interpreted as follows:

0x0	configuration space
0x1	I/O space

0x2	32-bit memory space address
-----	--------------------------------

The bus number is a unique identifying number assigned to each bus within the PCI or PCIe domain.

The device number is a unique identifying number assigned to each device on a PCI bus, PCIe logical bus, or CardBus bus. A device number is unique only within the set of device numbers for a particular bus or logical bus.

Each CardBus device can have one to eight logically independent functions, each with its own independent set of configuration registers. Each function on a device is assigned a function number. For a device with only one function, the function number must be 0.

The register number fields select a particular register within the set of configuration registers corresponding to the selected function. When the address space type identifier indicates configuration space, non-zero register number extended bits select registers in extended configuration space.

The second and third values in the `reg` property 5-tuple specify the 64-bit address of the mappable resource within the PCI or PCIe address domain. Since the CardBus is a 32-bit bus, the second 32-bit tuple is not used. The third 32-bit tuple corresponds to the 32-bit address.

The fourth and fifth 32-bit values in the 5-tuple `reg` property specify the size of the mappable resource. The size is a 64-bit value. Since it's a 32-bit bus, only the fifth tuple is used.

The driver can refer to the elements of this array by index, and construct kernel mappings to these addresses using `ddi_regs_map_setup(9F)`. The index into the array is passed as the *rnumber* argument of `ddi_regs_map_setup(9F)`.

At a high-level interrupt context, you can use the `ddi_get*` and `ddi_put*` family of functions to access I/O and memory space. However, access to configuration space is not allowed when running at a high-interrupt level.

interrupts

This property consists of a single-integer element array. Valid interrupt property values are 1, 2, 3, and 4. This value is derived directly from the contents of the device's configuration-interrupt-pin register.

A driver should use an index value of 0 when registering its interrupt handler with the DDI interrupt interfaces.

All CardBus devices support the `reg` property. The device number and function number as derived from the `reg` property are used to construct the address part of the device name under `/devices`.

Only devices that generate interrupts support an `interrupts` property.

Occasionally it might be necessary to override or augment the configuration information supplied by a CardBus device. This change can be achieved by writing a driver configuration file that describes a prototype device node specification containing the additional properties required.

For the system to merge the prototype node specification into an actual device node, certain conditions must be met.

- First, the `name` property must be identical. The value of the `name` property needs to match the binding name of the device. The binding name is the name chosen by the system to bind a driver to a device and is either an alias associated with the driver or the hardware node name of the device.
- Second, the parent property must identify the PCI bus or PCIe logical bus.
- Third, the `unit-address` property must identify the card. The format of the `unit-address` property is:

`DD[,F]`

where `DD` is the device number and `F` is the function number. If the function number is 0, only `DD` is specified.

Examples **EXAMPLE 1** Sample Configuration File

An example configuration file called `ACME,scsi-hba.conf` for a CardBus device driver called `ACME,scsi-hba` follows:

```
#
# Copyright (c) 1995, ACME SCSI Host Bus Adaptor
# ident  "@(#)ACME,scsi-hba.conf 1.1 96/02/04"
name="ACME,scsi-hba" parent="/pci@1,0/pci@1f,4000"
    unit-address="3" scsi-initiator-id=6;
hba-advanced-mode="on";
hba-dma-speed=10;
```

In this example, a property `scsi-initiator-id` specifies the SCSI bus initiator id that the adapter should use, for just one particular instance of adapter installed in the machine. The `name` property identifies the driver and the `parent` property to identify the particular bus the card is plugged into. This example uses the parent's full path name to identify the bus. The `unit-address` property identifies the card itself, with device number of 3 and function number of 0.

EXAMPLE 1 Sample Configuration File *(Continued)*

Two global driver properties are also created: `hba-advanced-mode` (which has the string value `on`) and `hba-dma-speed` (which has the value `10 M bit/s`). These properties apply to all device nodes of the ACME, `scsi-hba`.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC, x86

See Also [driver.conf\(4\)](#), [attributes\(5\)](#), [ddi_intr_add_handler\(9F\)](#), [ddi_prop_lookup\(9F\)](#), [ddi_regs_map_setup\(9F\)](#)

Writing Device Drivers

IEEE 1275 PCI Bus Binding

Name compver – compatible versions file

Description compver is an ASCII file used to specify previous versions of the associated package which are upward compatible. It is created by a package developer.

Each line of the file specifies a previous version of the associated package with which the current version is backward compatible.

Since some packages may require installation of a specific version of another software package, compatibility information is extremely crucial. Consider, for example, a package called "A" which requires version "1.0" of application "B" as a prerequisite for installation. If the customer installing "A" has a newer version of "B" (version 1.3), the compver file for "B" must indicate that "1.3" is compatible with version "1.0" in order for the customer to install package "A".

Examples EXAMPLE 1 Sample compver file.

A sample compver file is shown below:

```
Version 1.3  
Version 1.0
```

See Also [pkginfo\(4\)](#)

Application Packaging Developer's Guide

Notes The comparison of the version string disregards white space and tabs. It is performed on a word-by-word basis. Thus, "Version 1.3" and "Version 1.3" would be considered the same.

The entries in the compver file must match the values assigned to the VERSION parameter in the [pkginfo\(4\)](#) files.

- Name** contents – list of files and associated packages
- Synopsis** /var/sadm/install/contents
- Description** The file /var/sadm/install/contents is a source of information about the packages installed on the system. This file must never be edited directly. Always use the package and patch commands (see SEE ALSO) to make changes to the contents file.

Each entry in the contents file is a single line. Fields in each entry are separated by a single space character.

Two major styles of entries exist, old style and new style. The following is the format of an old-style entry:

```
ftype class path package(s)
```

The following is the general format of a new-style entry:

```
path[=rpath] ftype class [ftype-optional-fields] package(s)
```

New-style entries differ for each *f*type. The *f*type designates the entry type, as specified in [pkgmap\(4\)](#). The format for new-style entries, for each *f*type, is as follows:

```
ftype s: path=rpath s class package
ftype l: path l class package
ftype d: path d class mode owner group package(s)
ftype b: path b class major minor mode owner group package
ftype c: path c class major minor mode owner group package
ftype f: path f class mode owner group size cksum modtime package
ftype x: path x class mode owner group package
ftype v: path v class mode owner group size cksum modtime package
ftype e: path e class mode owner group size cksum modtime package
```

A significant distinction between old- and new-style entries is that the former do not begin with a slash (/) character, while the latter (new-style) always do. For example, the following are old-style entries:

```
d none /dev SUNWcsd
e passwd /etc/passwd SUNWcsr
```

The following are new-style entries:

```
/dev d none 0755 root sys SUNWcsr SUNWcsd
/etc/passwd e passwd 0644 root sys 580 48299 1077177419 SUNWcsr
```

The following are the descriptions of the fields in both old- and new-style entries.

- path* The absolute path of the node being described. For *f* type *s* (indicating a symbolic link) this is the indirect pointer (link) name.
- rpath* The relative path to the real file or linked-to directory name.
- f*type A one-character field that indicates the entry type (see [pkgmap\(4\)](#)).

<i>class</i>	The installation class to which the file belongs (see pkgmap(4)).
<i>package</i>	The package associated with this entry. For <i>f</i> type <i>d</i> (directory) more than one package can be present.
<i>mode</i>	The octal mode of the file (see pkgmap(4)).
<i>owner</i>	The owner of the file (see pkgmap(4)).
<i>group</i>	The group to which the file belongs (see pkgmap(4)).
<i>major</i>	The major device number (see pkgmap(4)).
<i>minor</i>	The minor device number (see pkgmap(4)).
<i>size</i>	The actual size of the file in bytes as reported by <i>sum</i> (see pkgmap(4)).
<i>cksum</i>	The checksum of the file contents (see pkgmap(4)).
<i>modtime</i>	The time of last modification (see pkgmap(4)).

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTETYPE	ATTRIBUTEVALUE
Availability	system/core-os
Interface Stability	Uncommitted

See Also [pkgadd\(1M\)](#), [pkgadm\(1M\)](#), [pkgchk\(1M\)](#), [pkgmap\(4\)](#), [attributes\(5\)](#)

Notes As shown above, the interface stability of `/var/sadm/install/contents` is Uncommitted (see [attributes\(5\)](#)). It is common practice to use this file in a read-only manner to determine which files belong to which packages installed on a system. While this file has been present for many releases of the Solaris operating system, it might not be present in future releases. The fully supported way to obtain information from the installed package database is through [pkgchk\(1M\)](#). It is highly recommended that you use `pkgchk` rather than relying on the `contents` file.

Name contract – the contract file system

Synopsis /system/contract

Description The /system/contract file system acts as the primary interface to the contract subsystem. There is a subdirectory of /system/contract for each available contract type.

/system/contract can be mounted on any mount point, in addition to the standard /system/contract mount point, and can be mounted several places at once. Such additional mounts are allowed in order to facilitate the confinement of processes to subtrees of the file system using [chroot\(1M\)](#) and yet allow such processes to continue to use contract commands and interfaces.

A combination of standard system calls (for example, [open\(2\)](#), [close\(2\)](#), and [poll\(2\)](#)) and calls to [libcontract\(3LIB\)](#) access /system/contract files.

Consumers of the contract file system must be large file aware. See [largefile\(5\)](#) and [lfcompile64\(5\)](#).

DIRECTORY STRUCTURE At the top level, the /system/contract directory contains subdirectories named with each available contract type, and one special directory, `all`. Each of these directories is world-readable and world-searchable.

STRUCTURE OF /system/contract/type Each /system/contract/type directory contains a fixed number of files. It also contains a variable number of subdirectories corresponding to existing contracts of type *type* and named with the decimal representation of the contracts' IDs.

The following files are in a /system/contract/type directory:

`template` Opening this file returns a file descriptor for a new *type* contract template.

You can use the following [libcontract\(3LIB\)](#) calls on a template file descriptor:

>

```
ct_tmpl_activate(3contract)
ct_tmpl_clear(3contract)
ct_tmpl_create(3contract)
```

See **TERMS** for additional template functions.

`latest` Opening this file returns a file descriptor for the status file of the last *type* contract written by the opening LWP. See **STRUCTURE OF /system/contract/type/id**. If the opening LWP has not created a *type* contract, opening `latest` fails with `ESRCH`.

`bundle` Opening this file returns a file descriptor for an event endpoint which receives events from all *type* contracts on the system. No privileges are required to open a type bundle event endpoint. Events sent by contracts owned and written by

users other than the reader's effective user id are invisible, that is, they are silently skipped, unless the reader has {PRIV_CONTRACT_OBSERVER} in its effective set. See EVENTS.

pbundle Opening this file returns a file descriptor for an event endpoint which receives events from all *type* contracts held by the opening process. See EVENTS.

STRUCTURE OF /system/contract/all The /system/contract/all directory contains a numerically named file for each contract in the system. Each file is a symbolic link to the type-specific directory for that contract, that is /system/contract/all/*id* points to /system/contract/*type/id*.

STRUCTURE OF /system/contract/type/id Each /system/contract/*type/id* directory contains the following files:

ctl Opening this file returns a file descriptor for contract *id*'s control file. The open fails if the opening process does not hold contract *id* and the contract has not been inherited by the process contract of which the opening process is a member. See [process\(4\)](#).

The following [libcontract\(3LIB\)](#) calls can be made on a `ctl` file descriptor if the contract is owned by the caller:

```
ct_ctl_abandon(3contract)
ct_ctl_newct(3contract)
ct_ctl_ack(3contract)
ct_ctl_qack(3contract)
```

The following [libcontract\(3LIB\)](#) call can be made on a `ctl` file descriptor if the contract doesn't have an owner:

```
ct_ctl_adopt(3contract)
```

status Opening this file returns a file descriptor for contract *id*'s status file. The following [libcontract\(3LIB\)](#) calls can be made on a status file descriptor:

```
ct_status_read(3contract)
```

See STATUS.

events Opening this file returns a file descriptor for an event endpoint which receives events from contract *id*. See EVENTS.

Only a process which has the same effective user ID as the process owning the contract, the same effective user ID as the contract's author, or has {PRIV_CONTRACT_OBSERVER} in its effective set can open the event endpoint for a contract.

TERMS The following terms are defined for all contracts:

cookie	Specifies a 64-bit quantity that the contract author can use to identify the contract. Use <code>ct_tmpl_set_cookie(3CONTRACT)</code> to set this term.
informative event set	Selects which events are delivered as informative events. Use <code>ct_tmpl_set_informative(3CONTRACT)</code> to set this term.
critical event set	Selects which events are delivered as critical events. Use <code>ct_tmpl_set_critical(3CONTRACT)</code> to set this term.
STATUS	A status object returned by <code>ct_status_read(3CONTRACT)</code> contains the following pieces of information:
contract ID	The numeric ID of the contract. Use <code>ct_status_get_id(3CONTRACT)</code> to obtain this information.
contract type	The type of the contract, specified as a string. Obtained using <code>ct_status_get_type(3CONTRACT)</code> . The contract type is the same as its subdirectory name under <code>/system/contract</code> .
creator's zone ID	The zone ID of the process which created the contract. Obtained using <code>ct_status_get_zoneid(3CONTRACT)</code> .
ownership state	The state of the contract, specified as <code>CTS_OWNED</code> , <code>CTS_INHERITED</code> , <code>CTS_ORPHAN</code> , or <code>CTS_DEAD</code> . Use <code>ct_status_get_state(3CONTRACT)</code> to obtain this information.
contract holder	If the contract's state is <code>CTS_OWNED</code> , the ID of the process which owns the contract. If the contract's state is <code>CTS_INHERITED</code> , the ID of the contract which is acting as regent. If the contract's state is <code>CTS_ORPHAN</code> or <code>CTS_DEAD</code> , this is undefined. Use <code>ct_status_get_holder(3CONTRACT)</code> to obtain this information.
number of critical events	The number of unacknowledged critical events pending on the contract's event queue. Use <code>ct_status_get_nevents(3CONTRACT)</code> to obtain this information.
negotiation time	The time remaining before the current synchronous negotiation times out. Use <code>ct_status_get_ntime(3CONTRACT)</code> to obtain this information.
negotiation quantum time	The time remaining before the current negotiation quantum runs out. Use <code>ct_status_get_qtime(3CONTRACT)</code> to obtain this information.

negotiation event ID	The ID of the event which initiated the negotiation timeout. Use <code>ct_status_get_nevid(3CONTRACT)</code> to obtain this information.
cookie (term)	The contract's cookie term. Use <code>ct_status_get_cookie(3CONTRACT)</code> to obtain this information.
Informative event set (term)	The contract's informative event set. Use <code>ct_status_get_informative(3CONTRACT)</code> to obtain this information.
Critical event set (term)	The contract's critical event set. Use <code>ct_status_get_critical(3CONTRACT)</code> to obtain this information.

EVENTS All three event endpoints, `/system/contract/type/bundle`, `/system/contract/type/pbundle`, and `/system/contract/type/id/events`, are accessed in the same manner.

The following `libcontract(3LIB)` interfaces are used with an event endpoint file descriptor:

```
ct_event_read(3contract)
ct_event_read_critical(3contract)
ct_event_reset(3contract)
ct_event_next(3contract)
```

To facilitate processes watching multiple event endpoints, it is possible to `poll(2)` on event endpoints. When it is possible to receive on an endpoint file descriptor, `POLLIN` is set for that descriptor.

An event object returned by `ct_event_read(3CONTRACT)` contains the following information:

contract ID	The ID of the contract that generated the event. Use <code>ct_event_read(3CONTRACT)</code> to obtain this information.
event ID	The ID of the contract event. Use <code>ct_event_get_evid(3CONTRACT)</code> .
flags	A bit vector possibly including <code>CT_ACK</code> and <code>CTE_INFO</code> . Use <code>ct_event_get_flags(3CONTRACT)</code> to obtain this information.
event type	The type of event, equal to one of the constants specified in the contract type's manual page or <code>CT_EV_NEGEND</code> . Use <code>ct_event_get_type(3CONTRACT)</code> to obtain this information.

EVENTTYPES The following event types are defined:

CT_EV_NEGEND Some time after an exit negotiation is initiated, the CT_EV_NEGEND event is sent. This indicates that the negotiation ended. This might be because the operation was cancelled, or because the operation was successful. If successful, and the owner requested that a new contract be written, this contains the ID of that contract.

CT_EV_NEGEND cannot be included in a contract's informative or critical event set. It is always delivered and always critical. If CT_EV_NEGEND indicates that the operation was successful, no further events are sent. The contract's owner should use `ct_ctl_abandon(3CONTRACT)` to abandon the contract.

A CT_EV_NEGEND event contains:

negotiation ID	The ID of the negotiation which ended. Use <code>ct_event_get_nevid(3CONTRACT)</code> to obtain this information.
new contract ID	The ID of the newly created contract. This value is 0 if no contract was created, or the ID of the existing contract if the operation was not completed. Use <code>ct_event_get_newct(3CONTRACT)</code> to obtain this information.

Files	<code>/system/contract</code>	List of all contract types
	<code>/system/contract/all</code>	Directory of all contract IDs
	<code>/system/contract/all/id</code>	Symbolic link to the type-specific directory of contract <i>id</i>
	<code>/system/contract/type</code>	Specific type directory
	<code>/system/contract/type/template</code>	Template for the contract type
	<code>/system/contract/type/bundle</code>	Listening point for all contracts of that type
	<code>/system/contract/type/pbundle</code>	Listening point for all contracts of that type for the opening process
	<code>/system/contract/type /latest</code>	Status of most recent <i>type</i> contract created by the opening LWP
	<code>/system/contract/type/ID</code>	Directory for contract id
	<code>/system/contract/type/ID/events</code>	Listening point for contract id's events
	<code>/system/contract/type/ID/ctl</code>	Control file for contract ID

/system/contract/type/ID/status Status info for contract ID

See Also `ctrun(1)`, `ctstat(1)`, `ctwatch(1)`, `chroot(1M)`, `close(2)`, `ioctl(2)`, `open(2)`, `poll(2)`, `ct_ctl_abandon(3CONTRACT)`, `ct_event_read(3CONTRACT)`, `ct_event_get_evid(3CONTRACT)`, `ct_event_get_flags(3CONTRACT)`, `ct_event_get_nevid(3CONTRACT)`, `ct_event_get_newct(3CONTRACT)`, `ct_event_get_type(3CONTRACT)`, `ct_status_read(3CONTRACT)`, `ct_status_get_cookie(3CONTRACT)`, `ct_status_get_critical(3CONTRACT)`, `ct_status_get_holder(3CONTRACT)`, `ct_status_get_id(3CONTRACT)`, `ct_status_get_informative(3CONTRACT)`, `ct_status_get_nevid(3CONTRACT)`, `ct_status_get_nevents(3CONTRACT)`, `ct_status_get_ntime(3CONTRACT)`, `ct_status_get_qtime(3CONTRACT)`, `ct_status_get_state(3CONTRACT)`, `ct_status_get_type(3CONTRACT)`, `ct_tmpl_set_cookie(3CONTRACT)`, `ct_tmpl_set_critical(3CONTRACT)`, `ct_tmpl_set_informative(3CONTRACT)`, `libcontract(3LIB)`, `process(4)`, `largefile(5)`, `lfcompile(5)`, `privileges(5)`

Name copyright – copyright information file

Description copyright is an ASCII file used to provide a copyright notice for a package. The text may be in any format. The full file contents (including comment lines) are displayed on the terminal at the time of package installation.

See Also *Application Packaging Developer's Guide*

Name core – process core file

Description The operating system writes out a core file for a process when the process is terminated due to receiving certain signals. A core file is a disk copy of the contents of the process address space at the time the process received the signal, along with additional information about the state of the process. This information can be consumed by a debugger. Core files can also be generated by applying the [gcore\(1\)](#) utility to a running process.

Typically, core files are produced following abnormal termination of a process resulting from a bug in the corresponding application. Whatever the cause, the core file itself provides invaluable information to the programmer or support engineer to aid in diagnosing the problem. The core file can be inspected using a debugger such as [dbx\(1\)](#) or [mdb\(1\)](#) or by applying one of the [proc\(1\)](#) tools.

The operating system attempts to create up to two core files for each abnormally terminating process, using a global core file name pattern and a per-process core file name pattern. These patterns are expanded to determine the pathname of the resulting core files, and can be configured by [coreadm\(1M\)](#). By default, the global core file pattern is disabled and not used, and the per-process core file pattern is set to `core`. Therefore, by default, the operating system attempts to create a core file named `core` in the process's current working directory.

A process terminates and produces a core file whenever it receives one of the signals whose default disposition is to cause a core dump. The list of signals that result in generating a core file is shown in [signal.h\(3HEAD\)](#). Therefore, a process might not produce a core file if it has blocked or modified the behavior of the corresponding signal. Additionally, no core dump can be created under the following conditions:

- If normal file and directory access permissions prevent the creation or modification of the per-process core file pathname by the current process user and group ID. This test does not apply to the global core file pathname because, regardless of the UID of the process dumping core, the attempt to write the global core file is made as the superuser.
- Core files owned by the user `nobody` will not be produced. For example, core files generated for the superuser on an NFS directory are owned by `nobody` and are, therefore, not written.
- If the core file pattern expands to a pathname that contains intermediate directory components that do not exist. For example, if the global pattern is set to `/var/core/%n/core.%p`, and no directory `/var/core/'uname -n'` has been created, no global core files are produced.
- If the destination directory is part of a filesystem that is mounted read-only.
- If the resource limit `RLIMIT_CORE` has been set to `0` for the process, no per-process core file is produced. Refer to [setrlimit\(2\)](#) and [ulimit\(1\)](#) for more information on resource limits.

- If the core file name already exists in the destination directory and is not a regular file (that is, is a symlink, block or character special-file, and so forth).
- If the kernel cannot open the destination file `O_EXCL`, which can occur if same file is being created by another process simultaneously.
- If the process's effective user ID is different from its real user ID or if its effective group ID is different from its real group ID. Similarly, `set-user-ID` and `set-group-ID` programs do not produce core files as this could potentially compromise system security. These processes can be explicitly granted permission to produce core files using `coreadm(1M)`, at the risk of exposing secure information.

The core file contains all the process information pertinent to debugging: contents of hardware registers, process status, and process data. The format of a core file is object file specific.

For ELF executable programs (see [a.out\(4\)](#)), the core file generated is also an ELF file, containing ELF program and file headers. The `e_type` field in the file header has type `ET_CORE`. The program header contains an entry for every segment that was part of the process address space, including shared library segments. The contents of the mappings specified by `coreadm(1M)` are also part of the core image. Each program header has its `p_memsz` field set to the size of the mapping. The program headers that represent mappings whose data is included in the core file have their `p_filesz` field set the same as `p_memsz`, otherwise `p_filesz` is zero.

A mapping's data can be excluded due to the core file content settings (see `coreadm(1M)`), or due to some failure. If the data is excluded because of a failure, the program header entry will have the `PF_SUNW_FAILURE` flag set in its `p_flags` field.

The program headers of an ELF core file also contain entries for two NOTE segments, each containing several note entries as described below. The note entry header and core file note type (`n_type`) definitions are contained in `<sys/elf.h>`. The first NOTE segment exists for binary compatibility with old programs that deal with core files. It contains structures defined in `<sys/old_procfs.h>`. New programs should recognize and skip this NOTE segment, advancing instead to the new NOTE segment. The old NOTE segment is deleted from core files in a future release.

The old NOTE segment contains the following entries. Each has entry name "CORE" and presents the contents of a system structure:

<code>prpsinfo_t</code>	<code>n_type: NT_PRPSINFO</code> . This entry contains information of interest to the <code>ps(1)</code> command, such as process status, CPU usage, <code>nice</code> value, controlling terminal, user-ID, process-ID, the name of the executable, and so forth. The <code>prpsinfo_t</code> structure is defined in <code><sys/old_procfs.h></code> .
-------------------------	--

- `char array` `n_type: NT_PLATFORM`. This entry contains a string describing the specific model of the hardware platform on which this core file was created. This information is the same as provided by `sysinfo(2)` when invoked with the command `SI_PLATFORM`.
- `auxv_t array` `n_type: NT_AUXV`. This entry contains the array of `auxv_t` structures that was passed by the operating system as startup information to the dynamic linker. Auxiliary vector information is defined in `<sys/auxv.h>`.

Following these entries, for each active (non-zombie) light-weight process (LWP) in the process, the old NOTE segment contains an entry with a `prstatus_t` structure, plus other optionally-present entries describing the LWP, as follows:

- `prstatus_t` `n_type: NT_PRSTATUS`. This structure contains things of interest to a debugger from the operating system, such as the general registers, signal dispositions, state, reason for stopping, process-ID, and so forth. The `prstatus_t` structure is defined in `<sys/old_procfs.h>`.
- `prfpregset_t` `n_type: NT_PRFPREG`. This entry is present only if the LWP used the floating-point hardware. It contains the floating-point registers. The `prfpregset_t` structure is defined in `<sys/procfs_isa.h>`.
- `gwindows_t` `n_type: NT_GWINDOWS`. This entry is present only on a SPARC machine and only if the system was unable to flush all of the register windows to the stack. It contains all of the unspilled register windows. The `gwindows_t` structure is defined in `<sys/regset.h>`.
- `prxregset_t` `n_type: NT_PRXREG`. This entry is present only if the machine has extra register state associated with it. It contains the extra register state. The `prxregset_t` structure is defined in `<sys/procfs_isa.h>`.

The new NOTE segment contains the following entries. Each has entry name “CORE” and presents the contents of a system structure:

- `psinfo_t` `n_type: NT_PSINFO`. This structure contains information of interest to the `ps(1)` command, such as process status, CPU usage, nice value, controlling terminal, user-ID, process-ID, the name of the executable, and so forth. The `psinfo_t` structure is defined in `<sys/procfs.h>`.
- `pstatus_t` `n_type: NT_PSTATUS`. This structure contains things of interest to a debugger from the operating system, such as pending signals, state, process-ID, and so forth. The `pstatus_t` structure is defined in `<sys/procfs.h>`.
- `char array` `n_type: NT_PLATFORM`. This entry contains a string describing the specific model of the hardware platform on which this core file was created. This information is the same as provided by `sysinfo(2)` when invoked with the command `SI_PLATFORM`.

<code>auxv_t</code> array	<code>n_type: NT_AUXV</code> . This entry contains the array of <code>auxv_t</code> structures that was passed by the operating system as startup information to the dynamic linker. Auxiliary vector information is defined in <code><sys/auxv.h></code> .
<code>struct utsname</code>	<code>n_type: NT_UTSNAME</code> . This structure contains the system information that would have been returned to the process if it had performed a <code>uname(2)</code> system call prior to dumping core. The <code>utsname</code> structure is defined in <code><sys/utsname.h></code> .
<code>prcred_t</code>	<code>n_type: NT_PRCRED</code> . This structure contains the process credentials, including the real, saved, and effective user and group IDs. The <code>prcred_t</code> structure is defined in <code><sys/procfs.h></code> . Following the structure is an optional array of supplementary group IDs. The total number of supplementary group IDs is given by the <code>pr_ngroups</code> member of the <code>prcred_t</code> structure, and the structure includes space for one supplementary group. If <code>pr_ngroups</code> is greater than 1, there is <code>pr_ngroups - 1</code> <code>gid_t</code> items following the structure; otherwise, there is no additional data.
<code>char</code> array	<code>n_type: NT_ZONENAME</code> . This entry contains a string which describes the name of the zone in which the process was running. See <code>zones(5)</code> . The information is the same as provided by <code>getzonenamebyid(3C)</code> when invoked with the numerical ID returned by <code>getzoneid(3C)</code> .
<code>struct ssd</code> array	<code>n_type: NT_LDT</code> . This entry is present only on a 32-bit x86 machine and only if the process has set up a Local Descriptor Table (LDT). It contains an array of structures of type <code>struct ssd</code> , each of which was typically used to set up the <code>%gs</code> segment register to be used to fetch the address of the current thread information structure in a multithreaded process. The <code>ssd</code> structure is defined in <code><sys/sysi86.h></code> .
<code>core_content_t</code>	<code>n_type: NT_CONTENT</code> . This optional entry indicates which parts of the process image are specified to be included in the core file. See <code>coreadm(1M)</code> .

Following these entries, for each active and zombie LWP in the process, the new NOTE segment contains an entry with an `lwpsinfo_t` structure plus, for a non-zombie LWP, an entry with an `lwpsstatus_t` structure, plus other optionally-present entries describing the LWP, as follows. A zombie LWP is a non-detached LWP that has terminated but has not yet been reaped by another LWP in the same process.

<code>lwpsinfo_t</code>	<code>n_type: NT_LWPSINFO</code> . This structure contains information of interest to the <code>ps(1)</code> command, such as LWP status, CPU usage, <code>nice</code> value, LWP-ID, and so forth. The <code>lwpsinfo_t</code> structure is defined in <code><sys/procfs.h></code> . This is the only entry present for a zombie LWP.
-------------------------	--

<code>lwpstatus_t</code>	<code>n_type: NT_LWPSTATUS</code> . This structure contains things of interest to a debugger from the operating system, such as the general registers, the floating point registers, state, reason for stopping, LWP-ID, and so forth. The <code>lwpstatus_t</code> structure is defined in <code><sys/procfs.h></code> .
<code>gwindows_t</code>	<code>n_type: NT_GWINDOWS</code> . This entry is present only on a SPARC machine and only if the system was unable to flush all of the register windows to the stack. It contains all of the unspilled register windows. The <code>gwindows_t</code> structure is defined in <code><sys/regset.h></code> .
<code>prxregset_t</code>	<code>n_type: NT_PRXREG</code> . This entry is present only if the machine has extra register state associated with it. It contains the extra register state. The <code>prxregset_t</code> structure is defined in <code><sys/procfs_isa.h></code> .
<code>asrset_t</code>	<code>n_type: NT_ASRS</code> . This entry is present only on a SPARC V9 machine and only if the process is a 64-bit process. It contains the ancillary state registers for the LWP. The <code>asrset_t</code> structure is defined in <code><sys/regset.h></code> .

Depending on the [coreadm\(1M\)](#) settings, the section header of an ELF core file can contain entries for CTF, symbol table, and string table sections. The `sh_addr` fields are set to the base address of the first mapping of the load object that they came from to. This can be used to match those sections with the corresponding load object.

The size of the core file created by a process can be controlled by the user (see [getrlimit\(2\)](#)).

See Also [elfdump\(1\)](#), [gcore\(1\)](#), [mdb\(1\)](#), [proc\(1\)](#), [ps\(1\)](#), [coreadm\(1M\)](#), [getrlimit\(2\)](#), [setrlimit\(2\)](#), [setuid\(2\)](#), [sysinfo\(2\)](#), [uname\(2\)](#), [getzonebynameid\(3C\)](#), [getzoneid\(3C\)](#), [elf\(3ELF\)](#), [signal.h\(3HEAD\)](#), [a.out\(4\)](#), [proc\(4\)](#), [zones\(5\)](#)

ANSI C Programmer's Guide

Name crypt.conf – configuration file for pluggable crypt modules

Synopsis /etc/security/crypt.conf

Description crypt.conf is the configuration file for the pluggable crypt architecture. Each crypt module must provide a function to generate a password hash, [crypt_genhash_impl\(3C\)](#), and a function to generate the salt, [crypt_gensalt_impl\(3C\)](#).

The `module_path` field specifies the pathname to a shared library object that implements `crypt_genhash_impl()` and `crypt_gensalt_impl()`. If the pathname is not absolute, it is assumed to be relative to `/usr/lib/security/$ISA`. If the pathname contains the `$ISA` token, the token is replaced by an implementation-defined directory name that defines the path relative to the calling program's instruction set architecture.

The third field is used to pass module-specific options to the shared objects. See [crypt_genhash_impl\(3C\)](#) and [crypt_gensalt\(3C\)](#). It is the responsibility of the module to parse and interpret the options. The `params` field can be used by the modules to turn on debugging or to pass any module-specific parameters that control the output of the hashing algorithm

Examples **EXAMPLE 1** Increasing the Work Factor

The following example increases the work factor to 12 for the [crypt_bsdbf\(5\)](#) module.

```
2a /usr/lib/security/$ISA/crypt_bsdbf.so 12
```

EXAMPLE 2 Setting the Rounds

The following example sets the rounds for the [crypt_sunmd5\(5\)](#) module:

```
md5 /usr/lib/security/$ISA/crypt_sunmd5.so rounds=2000
```

EXAMPLE 3 Using the Default `/etc/security/crypt.conf`

The following default `/etc/security/crypt.conf` supports five plugins:

```
1      crypt_bsmd5.so.1
2a     crypt_bsdbf.so.1
md5    crypt_sunmd5.so.1
5      crypt_sha256.so.1
6      crypt_sha512.so.1
```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Committed

See Also `passwd(1)`, `crypt(3C)`, `crypt_genhash_impl(3C)`, `crypt_gensalt(3C)`,
`crypt_gensalt_impl(3C)`, `getpassphrase(3C)`, `passwd(4)`, `attributes(5)`, `crypt_bsdbf(5)`,
`crypt_bsdmd5(5)`, `crypt_sha256(5)`, `crypt_sha512(5)`, `crypt_sunmd5(5)`, `crypt_unix(5)`

Name crypto_certs – directory for certificate files for Solaris Cryptographic Framework

Synopsis /etc/crypto/certs/CA
 /etc/crypto/certs/SUNWobjectCA
 /etc/crypto/certs/*

Description The /etc/crypto/certs directory contains ASN.1 BER or PEM encoded certificate files for use by the Solaris Cryptographic Framework.

A default installation contains root anchors and signing certificates. The CA and SUNWobjectCA certificates are the trust anchors for all other certificates. Other certificates contain the certificates used to sign and verify the Solaris user and kernel cryptographic plug-ins

Additional signing certificates may be installed by third-party cryptographic providers. They should either be copied to /etc/crypto/certs or included in the package that delivers the provider.

Only certificates that are issued by the CA or SUNWobjectCA certificates and include the organization unit “Solaris Cryptographic Framework” in their subject distinguished names are accepted by the Solaris Cryptographic Framework. This restriction is in place due to US Export Law on the export of open cryptographic interfaces at the time of shipping this revision of the product.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also [elfsign\(1\)](#), [libpkcs11\(3LIB\)](#), [attributes\(5\)](#)

Name dacf.conf – device auto-configuration configuration file

Synopsis /etc/dacf.conf

Description The kernel uses the dacf.conf file to automatically configure hot plugged devices. Because the dacf.conf file contains important kernel state information, it should not be modified.

The format of the /etc/dacf.conf file is not public and might change in versions of the Solaris operating environment that are not compatible with Solaris 8.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also [attributes\(5\)](#)

Notes This document does not constitute an API. The /etc/dacf.conf file might not exist or might contain different contents or interpretations in versions of the Solaris operating environment that are not compatible with Solaris 8. The existence of this notice does not imply that any other documentation lacking this notice constitutes an API.

Name dat.conf – DAT static registry

Synopsis /etc/dat/dat.conf

Description The DAT static registry, /etc/dat/dat.conf is a system-wide data resource maintained by the system administrative command [datadm\(1M\)](#).

/etc/dat/dat.conf contains a list of interface adapters supported by uDAPL service providers. An interface adapter on Infiniband (IB) corresponds to an IB Partition instance, for example, p8001.ibp0. An IB partition object represents an IP interface plumbed by [ifconfig\(1M\)](#) on an IB partition/Host Channel Adapter port combination.

Each entry in the DAT static registry is a single line that contains eight fields. Fields are separated by a SPACE. Lines that begin with a pound sign (#) are considered comments. All characters that follow the # are ignored. Enclose Solaris specific strings (*Solaris_specific_string*) and service provider's instance data (*service_provider_instance_data*) in quotes.

The following shows the order of the fields in a dat.conf entry:

```
"interface_adapter_name" "API_version" "threadsafe | nonthreadsafe" \
"default | nondefault" "service_provider_library_pathname" \
"service_provider_version" "service_provider_instance_data" \
"Solaris_specific_string"
```

The fields are defined as follows:

<i>interface_adapter_name</i>	Specifies the Interface Adapter (IA) name. In IB, this is the IB partition data-link name, for example, p8001.ibp0. This represents an IP interface plumbed on an IB partition/port combination of the HCA.
<i>API_version</i>	Specifies the API version of the service provide library: For example, "u"major.minor is u1.2.
<i>threadsafe nonthreadsafe</i>	Specifies a threadsafe or non-threadsafe library.
<i>default nondefault</i>	Specifies a default or non-default version of library. A service provider can offer several versions of the library. If so, one version is designated as default with the rest as nondefault.
<i>service_provider_library_pathname</i>	Specifies the pathname of the library image.
<i>service_provider_version</i>	Specifies the version of the service provider. By convention, specify the company stock symbol as the service provider, followed by major and minor version numbers, for example, SUNW1.0.
<i>service_provider_instance_data</i>	Specifies the service provider instance data.

Solaris_specific_string

Specifies a platform specific string, for example, the device name in the `service_provider.conf` file.

Examples EXAMPLE 1 Sample `dat.conf` File

The following `dat.conf` file shows a uDAPL 1.2 service provider for `tavor`, `udapl_tavor.so.1` supporting two interfaces, `p8001.ibp0` and `p8001.ibp1`:

```
#
# dat.conf for uDAPL 1.2
#
p8001.ibp0 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=tavor"
p8001.ibp1 u1.2 nonthreadsafe default udapl_tavor.so.1 SUNW.1.0 ""
"driver_name=tavor"
```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/io/infiniband/udapl
Interface Stability	Committed
Standard	uDAPL 1.1, 1.2

See Also [datadm\(1M\)](#), [ifconfig\(1M\)](#), [libdat\(3LIB\)](#), [service_provider.conf\(4\)](#), [attributes\(5\)](#)

Notes An empty `dat.conf` is created during the package `system/io/infiniband/udapl` installation if no file is present beforehand. Entries in the file are added or removed by running [datadm\(1M\)](#).

The content of the platform specific string does not constitute an API. It is generated by [datadm\(1M\)](#) and might have a different content or interpretation in a future release.

Name defaultdomain – specify host's domain name

Synopsis svc:/network/nis/domain
/etc/defaultdomain

Description The service svc:/network/nis/domain determines a host's domain name for direct use by the NIS name service. For the purposes of backwards compatibility, the service also generates the defaultdomain file.

The property group/property config/domainname represents the NIS domainname represented by /etc/defaultdomain.

The contents of defaultdomain consists of a single line containing a host's domain name.

Examples EXAMPLE 1 Setting Up domainname and Enabling the nis/domain Service

The following example sets up domainname and enables the nis/domain service:

```
% svccfg -s nis/domain setprop config/domainname = \  
    hostname: nisdomain.example.com  
% svccfg -s nis/domain:default refresh  
% svcadm enable nis/domain
```

The service is not enabled by default. The nis/domain service sets up all the shared NIS configuration, of which domainname is the one mandatory configuration parameter.

See Also [uname\(1\)](#), [ldapclient\(1M\)](#), [ypbind\(1M\)](#), [ypinit\(1M\)](#), [resolv.conf\(4\)](#), [ldap\(5\)](#)

Notes The defaultdomain file is created and modified by Oracle Solaris installation and configuration scripts. Only users knowledgeable of name service configuration should edit the file.

Name default_fs, fs – specify the default file system type for local or remote file systems

Description When file system administration commands have both specific and generic components (for example, [fsck\(1M\)](#)), the file system type must be specified. If it is not explicitly specified using the `-F FSType` command line option, the generic command looks in `/etc/vfstab` in order to determine the file system type, using the supplied raw or block device or mount point. If the file system type can not be determined by searching `/etc/vfstab`, the command will use the default file system type specified in either `/etc/default/fs` or `/etc/dfs/dfstypes`, depending on whether the file system is local or remote.

The default local file system type is specified in `/etc/default/fs` by a line of the form `LOCAL=fstype` (for example, `LOCAL=ufs`). The default remote file system type is determined by the first entry in the `/etc/dfs/fstypes` file.

File system administration commands will determine whether the file system is local or remote by examining the specified device name. If the device name starts with “/” (slash), it is considered to be local; otherwise it is remote.

The default file system types can be changed by editing the default files with a text editor.

Files	<code>/etc/vfstab</code>	list of default parameters for each file system
	<code>/etc/default/fs</code>	the default local file system type
	<code>/etc/dfs/fstypes</code>	the default remote file system type

See Also [fsck\(1M\)](#), [fstypes\(4\)](#), [vfstab\(4\)](#)

Name defaultrouter – configuration file for default router(s)

Synopsis /etc/defaultrouter

Description The /etc/defaultrouter file specifies a IPv4 host's default router(s).

The format of the file is as follows:

IP_address

...

The /etc/defaultrouter file can contain the IP addresses or hostnames of one or more default routers, with each entry on its own line. If you use hostnames, each hostname must also be listed in the local /etc/hosts file, because no name services are running at the time that defaultrouter is read.

Lines beginning with the “#” character are treated as comments.

The default routes listed in this file replace those added by the kernel during diskless booting. An empty /etc/defaultrouter file will cause the default route added by the kernel to be deleted.

Use of a default route, whether received from a DHCP server or from /etc/defaultrouter, prevents a machine from acting as an IPv4 router. You can use [routeadm\(1M\)](#) to override this behavior.

Files /etc/defaultrouter Configuration file containing the hostnames or IP addresses of one or more default routers.

See Also [in.rdisc\(1M\)](#), [in.routed\(1M\)](#), [routeadm\(1M\)](#), [hosts\(4\)](#)

Name depend – software dependencies file

Description depend is an ASCII file used to specify information concerning software dependencies for a particular package. The file is created by a software developer.

Each entry in the depend file describes a single software package. The instance of the package is described after the entry line by giving the package architecture and/or version. The format of each entry and subsequent instance definition is:

```
type pkg name
  (arch)version
    (arch)version
    ...
```

The fields are:

type	Defines the dependency type. Must be one of the following characters:
P	Indicates a prerequisite for installation; for example, the referenced package or versions must be installed.
I	Implies that the existence of the indicated package or version is incompatible.
R	Indicates a reverse dependency. Instead of defining the package's own dependencies, this designates that another package depends on this one. This type should be used only when an old package does not have a depend file, but relies on the newer package nonetheless. Therefore, the present package should not be removed if the designated old package is still on the system since, if it is removed, the old package will no longer work.
pkg	Indicates the package abbreviation.
name	Specifies the full package name.
(arch)version	Specifies a particular instance of the software. A version name cannot begin with a left parenthesis. The instance specifications, both (arch) and version, are completely optional, but each (arch)version pair must begin on a new line that begins with white space. A null version set equates to any version of the indicated package.

Examples EXAMPLE 1 Sample of depend file

Here are the contents of a sample depend file, for the SUNWftpr (FTP Server) package, stored in /var/sadm/pkg/SUNWftpr/install:

```
P SUNWcar      Core Architecture, (Root)
P SUNWkvm      Core Architecture, (Kvm)
P SUNWcsr      Core Solaris, (Root)
```


EXAMPLE 1 Sample of depend file *(Continued)*

```
P SUNWcsu      Core Solaris, (Usr)
P SUNWcsd      Core Solaris Devices
P SUNWcsl      Core Solaris Libraries
R SUNWftpu     FTP Server, (Usr)
```

See Also [pkginfo\(4\)](#)

Application Packaging Developer's Guide

Name device_allocate – device_allocate file

Synopsis /etc/security/device_allocate

Description The `device_allocate` file is an ASCII file that resides in the `/etc/security` directory. It contains mandatory access control information about each physical device. Each device is represented by a one–line entry of the form:

device-name;device-type;reserved1;reserved2;auths;device-exec

where:

device-name

Represents an arbitrary ASCII string naming the physical device. This field contains no embedded white space or non-printable characters.

device-type

Represents an arbitrary ASCII string naming the generic device type. This field identifies and groups together devices of like type. This field contains no embedded white space or non-printable characters. The following types of devices are currently managed by the system: `audio`, `sr` (represents CDROM drives), `st` (represents tape drives), and `rmdisk` (removable media devices).

reserved1

On systems configured with Trusted Extensions, this field stores a colon-separated (`:`) list of key-value pairs that describe device allocation attributes used in Trusted Extensions. Zero or more keys can be specified. The following keys are currently interpreted by Trusted Extensions systems:

`minlabel`

Specifies the minimum label at which device can be allocated. Default value is `admin_low`.

`maxlabel`

Specifies the maximum label at which device can be allocated. Default value is `admin_high`.

`zone`

Specifies the name of the zone in which device is currently allocated.

`class`

Specifies a logical grouping of devices. For example, all Sun Ray devices of all device types. There is no default class.

`xdpy`

Specifies the X display name. This is used to identify devices associated with that X session. There is no default `xdpy` value.

reserved2

Represents a field reserved for future use.

auths

Represents a field that contains a comma-separated list of authorizations required to allocate the device, an asterisk (*) to indicate that the device is *not* allocatable, or an '@' symbol to indicate that no explicit authorization is needed to allocate the device. The default authorization is `solaris.device.allocate`. See [auths\(1\)](#).

device-exec

The physical device's data clean program to be run any time the device is acted on by [allocate\(1\)](#). This ensures that unmanaged data does not remain in the physical device between uses. This field contains the filename of a program in `/etc/security/lib` or the full pathname of a cleanup script provided by the system administrator.

Notes on
device_allocate

The `device_allocate` file is an ASCII file that resides in the `/etc/security` directory.

Lines in `device_allocate` can end with a '\ ' to continue an entry on the next line.

Comments can also be included. A '#' makes a comment of all further text until the next NEWLINE not immediately preceded by a '\ '.

White space is allowed in any field.

The `device_allocate` file must be created by the system administrator before device allocation is enabled.

The `device_allocate` file is owned by root, with a group of sys, and a mode of 0644.

Examples EXAMPLE 1 Declaring an Allocatable Device

Declare that physical device `st0` is a type `st`. `st` is allocatable, and the script used to clean the device after running [deallocate\(1\)](#) is named `/etc/security/lib/st_clean`.

```
# scsi tape
st0;\
    st;\
    reserved;\
    reserved;\
    solaris.device.allocate;\
    /etc/security/lib/st_clean
```

Making a device allocatable means that you need to allocate and deallocate it to use it (with [allocate\(1\)](#) and [deallocate\(1\)](#)). If a device is not allocatable, there is an asterisk (*) in the *auths* field, and no one can use the device.

Files `/etc/security/device_allocate` Contains list of allocatable devices

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Uncommitted

See Also [auths\(1\)](#), [allocate\(1\)](#), [deallocate\(1\)](#), [list_devices\(1\)](#), [auth_attr\(4\)](#), [attributes\(5\)](#)

Notes On systems configured with Trusted Extensions, the functionality is enabled by default. On such systems, the `device_allocate` file is updated automatically by the system.

Name device_contract – device contract type

Synopsis /system/contract/device

Description Device contracts allow processes to monitor events involving a device of interest and to react and/or block state changes involving such devices.

Device contracts are managed using the [contract\(4\)](#) file system and the [libcontract\(3LIB\)](#) library. The process contract type directory is /system/contract/device.

Creation A device contract may be created in one of two ways:

- A process may create and activate a template and then invoke `open` on a minor node of the device. The act of opening creates a contract based on the terms in the activated template.
- A process may create a contract after it has opened a device by creating a template, setting appropriate terms (including the path to a minor node) on the template and then invoking `ct_tmpl_create()` on the template.

States, Breaks and Events A state refers to the state of the device which is the subject of the contract. Currently, three states are defined for device contracts:

`CT_DEV_EV_ONLINE` The device is online and functioning normally.

`CT_DEV_EV_DEGRADED` The device is online, but functioning in a degraded capacity.

`CT_DEV_EV_OFFLINE` The device is offline and is not configured for use.

A process creates a device contract with the kernel to get a guarantee that the device is in an acceptable set of states as long as the contract is valid. This acceptable set (or “A-set”, for short) is specified as one of the terms of the contract when the contract is created.

When a device moves to a state outside the “A-set”, the contract is broken. The breaking of the contract may be either asynchronous or synchronous, depending on whether the transition that led to the breaking of the contract is synchronous or asynchronous.

If the breaking of a contract is asynchronous, then a critical event is generated and sent to the contract holder. The event is generated even if the contract holder has not subscribed to the event via the critical or informative event sets.

If the breaking of the contract is synchronous, a critical contract event is generated with the `CTE_NEG` flag set to indicate that this is a negotiation event. The contract holder is expected to either acknowledge (ACK) this change and allow the state change to occur or it may negatively acknowledge (NACK) the change to block it (if it has sufficient privileges).

The term “event” refers to the transition of a device from one state to another. The event is named by the state to which the device is transitioning. For instance, if a device is transitioning to the `OFFLINE` state, the name of the event is `CT_DEV_EV_OFFLINE`. An event may have no

consequence for a contract, or it may result in the asynchronous breaking of a contract or it may result in a synchronous (that is, negotiated) breaking of a contract. Events are delivered to a contract holder in three cases:

- The contract holder has subscribed to the event via the critical or informative event sets. The event may be either critical or informative in this case depending on the subscription.
- The device transitions to a state outside the contract's "A-set" and the transition is asynchronous. This results in the asynchronous breaking of the contract and a critical event is delivered to the holder.
- The device transitions to a state outside the contract's "A-set" and the transition is synchronous. This results in the synchronous breaking of the contract and a critical event with the CTE_NEG flag set is delivered to the holder.

In the last two cases, a critical event is delivered even if the holder has not subscribed to the event via the critical or informative event sets.

NEGOTIATION If the breaking of a contract is synchronous, the kernel begins negotiations with the contract holder by generating a critical event before the device changes state. The event has the CTE_NEG flag set indicating that this is a negotiation event. The contract owner is allowed a limited period of time in which to either ACK the contract event (thus, allowing the state change) or if it has appropriate privileges, NACK the state change (thus, blocking the state change). ACKs may be sent by the holder via `ct_ctl_ack(3CONTRACT)` and NACKs may be sent via `ct_ctl_nack(3CONTRACT)`. If a contract holder does not send either a NACK or ACK within a specified period of time, an ACK is assumed and the kernel proceeds with the state change.

Once the device state change is finalized, the contract subsystem sends negotiation end (NEGEND) critical messages to the contract owner indicating the final disposition of the state transition. That is, either success or failure.

Once a contract is broken, a contract owner may choose to create a replacement contract. It may do this after the contract is broken or it may choose to do this synchronously with the breaking of the old contract via `ct_ctl_newct(3CONTRACT)`.

TERMS The following common contract terms, defined in `contract(4)`, have device-contract specific attributes:

informative set The default value for the informative set is CT_DEV_EV_DEGRADE that is, transitions to the DEGRADED state will by default result in informative events. Use `ct_tmpl_set_informative(3CONTRACT)` to set this term.

critical set The default value for the informative set is CT_DEV_EV_OFFLINE. That is, transitions to the OFFLINE state will by default result in critical events. Use `ct_tmpl_set_critical(3CONTRACT)` to set this term.

The following contract terms can be read from or written to a device contract template using the named `libcontract(3LIB)` interfaces. These contract terms are in addition to those described in `contract(4)`.

`CTDP_ACCEPT` Acceptable set or “A-set”.

This term is required for every device contract. It defines the set of device states which the contract owner expects to exist as long as the contract is valid. If a device transitions to a state outside this “A-set”, then the contract breaks and is no longer valid. A critical contract event is sent to the contract owner to signal this break.

Use `ct_dev_tmpl_set_aset()` to set this term. The default “A-set” is `CT_DEV_EV_ONLINE | CT_DEV_EV_DEGRADE`. This term is mandatory. Use `ct_dev_tmpl_get_aset()` to query a template for this term.

`CTDP_MINOR` Specifies the `devfs` path to a minor node that is the subject of the contract. Used to specify the minor node to be used for creating a contract when contract creation takes place other than at open time.

If the contract is created synchronously at `open(2)` time, then this term is implied to be the minor node being opened. In this case, this term need not be explicitly be set.

Use `ct_dev_tmpl_set_minor()` to set this term. The default setting for this term is `NULL`. That is, no minor node is specified.

Use `ct_dev_tmpl_get_noneg()` to query a template for the setting of this term.

`CTDP_NONEG` If set, this term indicates that any negotiable departure from the contract terms should be NACKed. That is, the contract subsystem should assume a NACK for any negotiated breaking of the contract. This term is ignored for asynchronous contract breaks.

Use `ct_dev_tmpl_set_noneg()` to set this term. The default setting is off.

Use `ct_dev_tmpl_get_noneg()` to query a template for the setting of this term.

`STATUS` In addition to the standard items, the status object read from a status file descriptor contains the following items if `CTD_FIXED` is specified:

`CTDS_STATE` Returns the current state of the device. One of the following states is returned:

- `CT_DEV_EV_ONLINE`
- `CT_DEV_EV_DEGRADED`

- CT_DEV_EV_OFFLINE

Use `ct_dev_status_get_dev_state()` to obtain this information.

CTDS_ASET Returns the acceptable states (“A-set”) of the device contract. The return value is a bitset of device states and may include one or more of the following:

- CT_DEV_EV_ONLINE
- CT_DEV_EV_DEGRADED
- CT_DEV_EV_OFFLINE

Use `ct_dev_status_get_aset()` to obtain this information.

CTDS_NONEG Returns the current setting of the noneg flag. Returns 1 if the noneg flag is set, or 0 if the flag is not set. Use `ct_dev_status_get_noneg()` to obtain this information.

If **CTD_ALL** is specified, the following items are also available:

CTDS_MINOR The `devfs` path of the device which is the subject of the device contract. Use `ct_dev_status_get_minor()` to obtain this information.

EVENTS No new event related interfaces (beyond the standard contract event interfaces) are defined for device contract events.

Files `/usr/include/sys/contract/device.h` Contains definitions of events, status fields and event fields

See Also `ctrun(1)`, `ctstat(1)`, `ctwatch(1)`, `open(2)`, `ct_tmpl_set_critical(3CONTRACT)`, `ct_tmpl_set_informative(3CONTRACT)`, `ct_dev_tmpl_set_aset(3CONTRACT)`, `ct_dev_tmpl_get_aset(3CONTRACT)`, `ct_dev_tmpl_set_minor(3CONTRACT)`, `ct_dev_tmpl_get_minor(3CONTRACT)`, `ct_dev_tmpl_set_noneg(3CONTRACT)`, `ct_dev_tmpl_get_noneg(3CONTRACT)`, `ct_dev_status_get_dev_state(3CONTRACT)`, `ct_dev_status_get_aset(3CONTRACT)`, `ct_dev_status_get_minor(3CONTRACT)`, `libcontract(3LIB)`, `contract(4)`, `privileges(5)`

Name device_maps – device_maps file

Synopsis /etc/security/device_maps

Description The device_maps file contains access control information about each physical device. Each device is represented by a one line entry of the form:

device-name : *device-type* : *device-list* :

where

device-name

This is an arbitrary ASCII string naming the physical device. This field contains no embedded white space or non-printable characters.

device-type

This is an arbitrary ASCII string naming the generic device type. This field identifies and groups together devices of like type. This field contains no embedded white space or non-printable characters.

device-list

This is a list of the device special files associated with the physical device. This field contains valid device special file path names separated by white space.

The device_maps file is an ASCII file that resides in the /etc/security directory.

Lines in device_maps can end with a ‘\’ to continue an entry on the next line.

Comments may also be included. A ‘#’ makes a comment of all further text until the next NEWLINE not immediately preceded by a ‘\’.

Leading and trailing blanks are allowed in any of the fields.

The device_maps file must be created by the system administrator before device allocation is enabled.

This file is owned by root, with a group of sys, and a mode of 0644.

Examples EXAMPLE 1 A Sample device_maps File

The following is a sample device_maps file:

```
# scsi tape
st1:\
rmt:\
/dev/rst21 /dev/nrst21 /dev/rst5 /dev/nrst5 /dev/rst13 \
/dev/nrst13 /dev/rst29 /dev/nrst29 /dev/rmt/1l /dev/rmt/1m \
/dev/rmt/1 /dev/rmt/1h /dev/rmt/1u /dev/rmt/1ln /dev/rmt/1mn \
/dev/rmt/1n /dev/rmt/1hn /dev/rmt/1un /dev/rmt/1b /dev/rmt/1bn:\
```

Files /etc/security/device_maps Contains access control information for devices.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Uncommitted

See Also [allocate\(1\)](#), [deallocate\(1\)](#), [list_devices\(1\)](#), [dminfo\(1M\)](#), [device_allocate\(4\)](#), [attributes\(5\)](#)

Notes On systems configured with Trusted Extensions, the functionality is enabled by default. On such systems, the [device_allocate\(4\)](#) file is updated automatically by the system.

Name devices, devid_cache, snapshot_cache, mdi_scsi_vhci_cache, mdi_ib_cache, devname_cache, pci_unitaddr_persistent, infiniband_hca_persistent_cache – device configuration information

Synopsis /etc/devices
/etc/devices/devid_cache
/etc/devices/devname_cache
/etc/devices/mdi_ib_cache
/etc/devices/mdi_scsi_vhci_cache
/etc/devices/snapshot_cache
/etc/devices/pci_unitaddr_persistent
/etc/devices/pci_unitaddr_persistent_cache

Description The directory /etc/devices is a repository of device-related data. Files in this directory are used to preserve this information across reboots and are created and updated as necessary by the system.

There are no administrative actions necessary with respect to files in /etc/devices. Should the contents of a file become corrupted or an update fail, the file can simply be removed. The system re-creates the file as necessary.

Do not delete or modify the contents of /etc/devices/pci_unitaddr_persistent or /etc/devices/pci_unitaddr_persistent_cache.

See Also [devfsadm\(1M\)](#), [dev\(7FS\)](#), [ddi_devid_compare\(9F\)](#), [ddi_devid_compare\(9F\)](#)

Notes Files in this directory do not constitute an API. Files might not exist or might have a different content or interpretation in a future release. The existence of this notice does not imply that any other documentation that lacks this notice constitutes an API.

Name dfstab – file containing commands for sharing resources across a network

Description This file is obsolete in the Oracle Solaris 11 release.

Adding share entries to the `/etc/dfs/dfstab` file is no longer required to permanently share files across a network. File systems are shared in the following ways:

- The `share` command is used to create and publish a NFS or SMB share of a file system or directory. Using the `share` command to share a ZFS file system is considered a legacy operation.
- The `zfs set share` command is used to establish NFS or SMB share properties for a ZFS dataset. Then, set the ZFS `sharenfs` or `sharesmb` to `on` to publish an NFS or SMB share.

An NFS or SMB share is published automatically when a system is rebooted by an SMF service. Because an SMF service manages NFS or SMB shares, editing the `/etc/dfs/dfstab` file is obsolete.

For more information, see the [share\(1M\)](#) manual page.

Attributes See [attributes\(5\)](#) for a description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Obsolete

See Also [share\(1M\)](#), [share_nfs\(1M\)](#), [share_smb\(1M\)](#), [zfs\(1M\)](#), [shareall\(1M\)](#), [svcadm\(1M\)](#), [attributes\(5\)](#)

Name dhcp_inittab – information repository for DHCP options

Description The `/etc/dhcp/inittab` and the `/etc/dhcp/inittab6` files contain information about the Dynamic Host Configuration Protocol (DHCP) options, which are network configuration parameters passed from DHCP servers to DHCP clients when a client machine uses DHCP. Since many DHCP-related commands must parse and understand these DHCP options, this file serves as a central location where information about these options may be obtained.

The DHCP `inittab` and `inittab6` files provide three general pieces of information:

- A mnemonic alias, or symbol name, for each option number. For instance, option 12 is aliased to the name `Hostname`. This is useful for DHCP-related programs that require human interaction, such as `dhcpcinfo(1)`.
- Information about the syntax for each option. This includes information such as the type of the value, for example, whether it is a 16-bit integer or an IP address.
- The policy for what options are visible to which DHCP-related programs.

If you make any changes to the `/etc/dhcp/inittab` file, note that only additions of or changes to `SITE` options are preserved during upgrade. For `/etc/dhcp/inittab6`, no options are preserved during upgrade.

The `VENDOR` options defined here are intended for use by the Solaris DHCP client and DHCP management tools. The `SUNW` vendor space is owned by Sun, and changes are likely during upgrade. If you need to configure the Solaris DHCP server to support the vendor options of a different client, see `dhcptab(4)` for details.

Each DHCP option belongs to a certain category, which roughly defines the scope of the option; for instance, an option may only be understood by certain hosts within a given site, or it may be globally understood by all DHCP clients and servers. The following categories are defined; the category names are not case-sensitive:

STANDARD	All client and server DHCP implementations agree on the semantics. These are administered by the Internet Assigned Numbers Authority (IANA). These options are numbered from 1 to 127 for IPv4 DHCP, and 1 to 65535 for DHCPv6.
SITE	Within a specific site, all client and server implementations agree on the semantics. However, at another site the type and meaning of the option may be quite different. These options are numbered from 128 to 254 for IPv4 DHCP. DHCPv6 does not support site options.
VENDOR	Each vendor may define 254 options (65536 for DHCPv6) unique to that vendor. The vendor is identified within a DHCP packet by the “Vendor Class” option, number 60 (number 17 for DHCPv6). An option with a specific numeric identifier belonging to one vendor will, in general, have a type and semantics different from that of a different vendor. Vendor options are “super-encapsulated” into the vendor field number 43, as defined in <i>RFC 2132</i>

for IPv4 DHCP, and number 17 as defined in RFC 3315 for DHCPv6. The `/etc/dhcp/inittab` file contains only Sun vendor options. Define non-Sun vendor options in the `dhcptab` file.

- FIELD** This category allows the fixed fields within a DHCP packet to be aliased to a mnemonic name for use with `dhcpinfo(1)`.
- INTERNAL** This category is internal to the Solaris DHCP implementation and will not be further defined.

DHCP inittab and inittab6 Format Data entries are written one per line and have seven fields; each entry provides information for one option. Each field is separated by a comma, except for the first and second, which are separated by whitespace (as defined in `isspace(3C)`). An entry cannot be continued onto another line. Blank lines and those whose first non-whitespace character is '#' are ignored.

The fields, in order, are:

- **Mnemonic Identifier**

The Mnemonic Identifier is a user-friendly alias for the option number; it is not case sensitive. This field must be per-category unique and should be unique across all categories. The option names in the `STANDARD`, `SITE`, and `VENDOR` spaces should not overlap, or the behavior will be undefined. See `Mnemonic Identifiers for Options` section of this man page for descriptions of the option names.

- **Category (scope)**

The Category field is one of `STANDARD`, `SITE`, `VENDOR`, `FIELD`, or `INTERNAL` and identifies the scope in which the option falls. `SITE` is not used in `inittab6`.

- **Option Number**

The Option Number is the number of this option when it is in a DHCP packet. This field should be per-category unique and the `STANDARD` and `SITE` fields should not have overlapping code fields or the behavior is undefined.

- **Data Type**

Data Type is one of the following values, which are not case sensitive:

<code>Ascii</code>	A printable character string
<code>Bool</code>	Has no value. Scope limited to category limited to <code>INTERNAL</code> . Presence of an option of this type within a Solaris configuration file represents <code>TRUE</code> , absence represents <code>FALSE</code> .
<code>Octet</code>	An array of bytes
<code>Unumber8</code>	An 8-bit unsigned integer
<code>Snumber8</code>	An 8-bit signed integer
<code>Unumber16</code>	A 16-bit unsigned integer

Snumber16	A 16-bit signed integer
Unumber24	A 24-bit unsigned integer
Unumber32	A 32-bit unsigned integer
Snumber32	A 32-bit signed integer
Unumber64	A 64-bit unsigned integer
Snumber64	A 64-bit signed integer
Ip	An IPv4 address
Ipv6	An IPv6 address
Duid	An RFC 3315 Unique Identifier
Domain	An RFC 1035-encoded domain name

The data type field describes an indivisible unit of the option payload, using one of the values listed above.

- Granularity

The Granularity field describes how many indivisible units in the option payload make up a whole value or item for this option. The value must be greater than zero (0) for any data type other than Bool, in which case it must be zero (0).

- Maximum Number Of Items

This value specifies the maximum items of Granularity which are permissible in a definition using this symbol. For example, there can only be one IP address specified for a subnet mask, so the Maximum number of items in this case is one (1). A Maximum value of zero (0) means that a variable number of items is permitted.

- Visibility

The Visibility field specifies which DHCP-related programs make use of this information, and should always be defined as `sdmi` for newly added options.

Mnemonic Identifiers for IPv4 Options

The following table maps the mnemonic identifiers used in Solaris DHCP to *RFC 2132* options:

<i>Symbol</i>	<i>Code</i>	<i>Description</i>
Subnet	1	Subnet Mask, dotted Internet address (IP).
UTCoffst	2	Coordinated Universal time offset (seconds).
Router	3	List of Routers, IP.
Timeserv	4	List of RFC-868 servers, IP.

<i>Symbol</i>	<i>Code</i>	<i>Description</i>
IEN116ns	5	List of IEN 116 name servers, IP.
DNSserv	6	List of DNS name servers, IP.
Logserv	7	List of MIT-LCS UDP log servers, IP.
Cookie	8	List of RFC-865 cookie servers, IP.
Lprserv	9	List of RFC-1179 line printer servers, IP.
Impress	10	List of Imagen Impress servers, IP.
Resource	11	List of RFC-887 resource location servers, IP.
Hostname	12	Client's hostname, value from hosts database.
Bootsize	13	Number of 512 octet blocks in boot image, NUMBER.
Dumpfile	14	Path where core image should be dumped, ASCII.
DNSdmain	15	DNS domain name, ASCII.
Swapserv	16	Client's swap server, IP.
Rootpath	17	Client's Root path, ASCII.
ExtendP	18	Extensions path, ASCII.
IpFwF	19	IP Forwarding Enable/Disable, NUMBER.
NLrouteF	20	Non-local Source Routing, NUMBER.
PFilter	21	Policy Filter, IP.
MaxIpSiz	22	Maximum datagram Reassembly Size, NUMBER.
IpTTL	23	Default IP Time to Live, (1=<x<=255), NUMBER.
PathTO	24	RFC-1191 Path MTU Aging Timeout, NUMBER.
PathTbl	25	RFC-1191 Path MTU Plateau Table, NUMBER.
MTU	26	Interface MTU, x>=68, NUMBER.
SameMtuF	27	All Subnets are Local, NUMBER.
Broadcst	28	Broadcast Address, IP.
MaskDscF	29	Perform Mask Discovery, NUMBER.
MaskSupF	30	Mask Supplier, NUMBER.
RDiscvyF	31	Perform Router Discovery, NUMBER.
RSolicitS	32	Router Solicitation Address, IP.

<i>Symbol</i>	<i>Code</i>	<i>Description</i>
StaticRt	33	Static Route, Double IP (network router).
TrailerF	34	Trailer Encapsulation, NUMBER.
ArpTime0	35	ARP Cache Time out, NUMBER.
EthEncap	36	Ethernet Encapsulation, NUMBER.
TcpTTL	37	TCP Default Time to Live, NUMBER.
TcpKaInt	38	TCP Keepalive Interval, NUMBER.
TcpKaGbF	39	TCP Keepalive Garbage, NUMBER.
NISdmain	40	NIS Domain name, ASCII.
NISservs	41	List of NIS servers, IP.
NTPservs	42	List of NTP servers, IP.
NetBNms	44	List of NetBIOS Name servers, IP.
NetBDsts	45	List of NetBIOS Distribution servers, IP.
NetBNDT	46	NetBIOS Node type (1=B-node, 2=P, 4=M, 8=H).
NetBScop	47	NetBIOS scope, ASCII.
XFontSrv	48	List of X Window Font servers, IP.
XDispMgr	49	List of X Window Display managers, IP.
LeaseTim	51	Lease Time Policy, (-1 = PERM), NUMBER.
Message	56	Message to be displayed on client, ASCII.
T1Time	58	Renewal (T1) time, NUMBER.
T2Time	59	Rebinding (T2) time, NUMBER.
NW_dmain	62	NetWare/IP Domain Name, ASCII.
NWIPOpts	63	NetWare/IP Options, OCTET (unknown type).
TFTPsrvN	66	TFTP server hostname, ASCII.
OptBootF	67	Optional Bootfile path, ASCII.
MblIPAgT	68	Mobile IP Home Agent, IP.
SMTPserv	69	Simple Mail Transport Protocol Server, IP.
POP3serv	70	Post Office Protocol (POP3) Server, IP.
NNTPserv	71	Network News Transport Proto. (NNTP) Server, IP.

<i>Symbol</i>	<i>Code</i>	<i>Description</i>
WWWservs	72	Default WorldWideWeb Server, IP.
Fingersv	73	Default Finger Server, IP.
IRCServs	74	Internet Relay Chat Server, IP.
STServs	75	StreetTalk Server, IP.
STDAservs	76	StreetTalk Directory Assist. Server, IP.
UserClas	77	User class information, ASCII.
SLP_DA	78	Directory agent, OCTET.
SLP_SS	79	Service scope, OCTET.
AgentOpt	82	Agent circuit ID, OCTET.
FQDN	89	Fully Qualified Domain Name, OCTET.
PXEarch	93	Client system architecture, NUMBER.
BootFile	N/A	File to Boot, ASCII.
BootPath	N/A	Boot path prefix to apply to client's requested boot file, ASCII.
BootSrvA	N/A	Boot Server, IP.
BootSrvN	N/A	Boot Server Hostname, ASCII.
EchoVC	N/A	Echo Vendor Class Identifier Flag, (Present=TRUE)
LeaseNeg	N/A	Lease is Negotiable Flag, (Present=TRUE)

Mnemonic Identifiers for IPv6 Options The following table maps the mnemonic identifiers used in Solaris DHCP to RFC 3315, 3319, 3646, 3898, 4075, and 4280 options:

<i>Symbol</i>	<i>Code</i>	<i>Description</i>
ClientID	1	Unique identifier for client, DUID
ServerID	2	Unique identifier for server, DUID
Preference	7	Server preference, NUMBER
Unicast	12	Unicast server address, IPV6
UserClass	15	User classes for client, OCTET
VendorClass	16	Vendor client hardware items, OCTET
SIPNames	21	SIP proxy server name list, DOMAIN

<i>Symbol</i>	<i>Code</i>	<i>Description</i>
SIPAddresses	22	SIP proxy server addresses in preference order, IPV6
DNSAddresses	23	DNS server addresses in preference order, IPV6
DNSSearch	24	DNS search list, DOMAIN
NISServers	27	NIS server addresses in preference order, IPV6
NISDomain	29	NIS domain name, DOMAIN
SNTPServers	31	IPV6
InfoRefresh	32	UNNUMBER32
BCMCDomain	33	Broadcast/multicast control server name list, DOMAIN
BCMCAddresses	34	Broadcast/multicast control server addresses, IPV6

Examples EXAMPLE 1 Altering the DHCP inittab File

In general, the DHCP `inittab` file should only be altered to add SITE options. If other options are added, they will not be automatically carried forward when the system is upgraded. For instance:

```
ipPairs    SITE, 132, IP, 2, 0, sdmi
```

describes an option named `ipPairs`, that is in the SITE category. That is, it is defined by each individual site, and is option code 132, which is of type IP Address, consisting of a potentially infinite number of pairs of IP addresses.

Files `/etc/dhcp/inittab`
`/etc/dhcp/inittabv6`

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also [dhcpinfo\(1\)](#), [dhcpageant\(1M\)](#), [isspace\(3C\)](#), [dhcptab\(4\)](#), [attributes\(5\)](#), [dhcp\(5\)](#), [dhcp_modules\(5\)](#)

Oracle Solaris Administration: IP Services

Alexander, S., and R. Droms. *RFC 2132, DHCP Options and BOOTP Vendor Extensions*. Network Working Group. March 1997.

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.

Schulzrinne, H., and B. Volz. *RFC 3319, Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers*. Columbia University and Ericsson. July 2003.

Droms, R. *RFC 3646, DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. Cisco Systems. December 2003.

Kalusivalingam, V. *RFC 3898, Network Information Service (NIS) Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. Cisco Systems. October 2004.

Chowdhury, K., P. Yegani, and L. Madour. *RFC 4280, Dynamic Host Configuration Protocol (DHCP) Options for Broadcast and Multicast Control Servers*. Starent Networks, Cisco Systems, and Ericsson. November 2005.

Mockapetris, P.V. *RFC 1035, Domain names - implementation and specification*. ISI. November 1987.

Name dhcp_network – DHCP network tables

Description The Dynamic Host Configuration Protocol (DHCP) network tables are used to map the client identifiers of DHCP clients to IP addresses and the associated configuration parameters of that address. One DHCP network table exists for each network served by the DHCP server, and each table is named using the network's IP address. There is no table or file with the name dhcp_network.

The DHCP network tables can exist as ASCII text files or binary text files, depending on the data store used. Since the format of the file could change, the preferred method of managing the DHCP network tables is through the use of `dhcpcmgr(1M)` or the `pntadm(1M)` command.

The dhcp_network file is used as a policy mechanism for whether `in.dhcpd(1M)` leases addresses on a given network. If the DHCP server is not serving leases or information to a network, there should be no dhcp_network file for that network. To set the DHCP server in *informational* mode, where it responds to INFORM messages but does not lease addresses on that network, create an empty dhcp_network file for that network. For normal operations, where the DHCP server both leases addresses and responds to INFORM packets, create a dhcp_network file using `dhcpcmgr(1M)` or `pntadm(1M)` and populate it with leasable addresses.

The format of the records in a DHCP network table depends on the data store used to maintain the table. However, an entry in a DHCP network table must contain the following fields:

<code>Client_ID</code>	The client identifier field, <code>Client_ID</code> , is an ASCII hexadecimal representation of the unique octet string value of the DHCP Client Identifier Option (code 61) which identifies a DHCP client. In the absence of the DHCP Client Identifier Option, the DHCP client is identified using the form given below for BOOTP clients. The number of characters in this field must be an even number, with a maximum length of 64 characters. Valid characters are 0 - 9 and A-F. Entries with values of 00 are freely available for dynamic allocation to requesting clients. BOOTP clients are identified by the concatenation of the network's hardware type (as defined by RFC 1340, titled "Assigned Numbers") and the client's hardware address. For example, the following BOOTP client has a hardware type of '01' (10mb ethernet) and a hardware address of 8:0:20:11:12:b7, so its client identifier would be: 010800201112B7				
<code>Flags</code>	The <code>Flags</code> field is a decimal value, the bit fields of which can have a combination of the following values: <table style="margin-left: 2em;"> <tr> <td style="vertical-align: top;">1 (PERMANENT)</td> <td>Evaluation of the Lease field is turned off (lease is permanent). If this bit is not set, Evaluation of the Lease field is enabled and the Lease is DYNAMIC .</td> </tr> <tr> <td style="vertical-align: top;">2 (MANUAL)</td> <td>This entry has a manual client ID binding (cannot be reclaimed by DHCP server). Client will not be allocated another address.</td> </tr> </table>	1 (PERMANENT)	Evaluation of the Lease field is turned off (lease is permanent). If this bit is not set, Evaluation of the Lease field is enabled and the Lease is DYNAMIC .	2 (MANUAL)	This entry has a manual client ID binding (cannot be reclaimed by DHCP server). Client will not be allocated another address.
1 (PERMANENT)	Evaluation of the Lease field is turned off (lease is permanent). If this bit is not set, Evaluation of the Lease field is enabled and the Lease is DYNAMIC .				
2 (MANUAL)	This entry has a manual client ID binding (cannot be reclaimed by DHCP server). Client will not be allocated another address.				

	4 (UNUSABLE)	When set, this value means that either through ICMP echo or client DECLINE, this address has been found to be unusable. Can also be used by the network administrator to <i>prevent</i> a certain client from booting, if used in conjunction with the MANUAL flag.
	8 (BOOTP)	This entry is reserved for allocation to BOOTP clients only.
Client_IP		The Client_IP field holds the IP address for this entry. This value must be unique in the database.
Server_IP		This field holds the IP address of the DHCP server which <i>owns</i> this client IP address, and thus is responsible for initial allocation to a requesting client. On a multi-homed DHCP server, this IP address must be the first address returned by <code>gethostbyname(3NSL)</code> .
Lease		This numeric field holds the entry's absolute lease expiration time, and is in seconds since January 1, 1970. It can be decimal, or hexadecimal (if 0x prefixes number). The special value - 1 is used to denote a permanent lease.
Macro		This ASCII text field contains the <code>dhcptab</code> macro name used to look up this entry's configuration parameters in the <code>dhcptab(4)</code> database.
Comment		This ASCII text field contains an optional comment.

TREATISE ON LEASES This section describes how the DHCP/BOOTP server calculates a client's configuration lease using information contained in the `dhcptab(4)` and DHCP network tables. The server consults the `LeaseTim` and `LeaseNeg` symbols in the `dhcptab`, and the `Flags` and `Lease` fields of the chosen IP address record in the DHCP network table.

The server first examines the `Flags` field for the identified DHCP network table record. If the PERMANENT flag is on, then the client's lease is considered permanent.

If the PERMANENT flag is not on, the server checks if the client's lease as represented by the `Lease` field in the network table record has expired. If the lease is not expired, the server checks if the client has requested a new lease. If the `LeaseNeg` symbol has not been included in the client's `dhcptab` parameters, then the client's requested lease extension is ignored, and the lease is set to be the time remaining as shown by the `Lease` field. If the `LeaseNeg` symbol *has* been included, then the server will extend the client's lease to the value it requested if this requested lease is less than or equal to the current time plus the value of the client's `LeaseTim` `dhcptab` parameter.

If the client's requested lease is greater than policy allows (value of `LeaseTim`), then the client is given a lease equal to the current time plus the value of `LeaseTim`. If `LeaseTim` is not set, then the default `LeaseTim` value is one hour.

For more information about the `dhcptab` symbols, see `dhcptab(4)`.

Attributes See [attributes\(5\)](#) for a description of the following attribute:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	service/network/dhcp
Interface Stability	Committed

See Also [dhcpconfig\(1M\)](#), [dhcpgmgr\(1M\)](#), [dhtadm\(1M\)](#), [in.dhcpd\(1M\)](#), [pntadm\(1M\)](#), [dhcptab\(4\)](#), [dhcp\(5\)](#), [dhcp_modules\(5\)](#), [attributes\(5\)](#)

Oracle Solaris DHCP Service Developer's Guide

Oracle Solaris Administration: IP Services

Reynolds, J. and J. Postel, *Assigned Numbers*, STD 2, RFC 1340, USC/Information Sciences Institute, July 1992.

Name dhcpcsvc.conf – file containing service configuration parameters for the DHCP service

Description The `dhcpcsvc.conf` file resides in directory `/etc/inet` and contains parameters for specifying Dynamic Host Configuration Protocol (DHCP) service configuration settings, including the type and location of DHCP data store used.

The description of the `dhcpcsvc.conf` file in this man page is informational only. The preferred method of setting or modifying values within the `dhcpcsvc.conf` file is by using [dhcpcconfig\(1M\)](#) or the [dhcpcmgr\(1M\)](#) utility. Do not edit the `dhcpcsvc.conf` file.

The `dhcpcsvc.conf` file format is ASCII; comment lines begin with the crosshatch (`#`) character. Parameters consist of a keyword followed by an equals (`=`) sign followed by the parameter value, of the form:

Keyword=Value

The following *Keyword* and *Value* parameters are supported:

BOOTP_COMPAT	String. <code>automatic</code> or <code>manual</code> . Enables support of BOOTP clients. Default is <code>no BOOTP</code> . Value selects BOOTP address allocation method. <code>automatic</code> to support all BOOTP clients, <code>manual</code> to support only registered BOOTP clients. <code>server mode only</code> parameter.
CACHE_TIMEOUT	Integer. Number of seconds the server caches data from data store. Used to improve performance. Default is 10 seconds. <code>server mode only</code> parameter.
CONVER	Integer. Container version. Used by DHCP administrative tools to identify which version of the public module is being used to administer the data store. <code>CONVER</code> should <i>not</i> be changed manually.
DAEMON_ENABLED	TRUE/FALSE. If TRUE, the DHCP daemon can be run. If FALSE, DHCP daemon process exits immediately if the daemon is started. Default is TRUE. Generic parameter.
HOSTS_DOMAIN	String. Defines name service domain that DHCP administration tools use when managing the hosts table. Valid only when <code>HOSTS_RESOURCE</code> is set to <code>dns</code> .
HOSTS_RESOURCE	String. Defines what name service resource should be used by the DHCP administration tools when managing the hosts table. Current valid values are <code>files</code> and <code>dns</code> .

ICMP_VERIFY	TRUE/FALSE. Toggles ICMP echo verification of IP addresses. Default is TRUE. server mode only parameter.
INTERFACES	String. Comma-separated list of interface names to listen to. Generic parameter.
LOGGING_FACILITY	Integer. Local facility number (0–7 inclusive) to log DHCP events to. Default is not to log transactions. Generic parameter.
OFFER_CACHE_TIMEOUT	Integer. Number of seconds before OFFER cache timeouts occur. Default is 10 seconds. server mode only parameter.
OWNER_IP	String. List of supplemental ownership addresses that will be used by the DHCP server in determining the <code>dhcp_network</code> records that are under its management. Addresses are in the dotted Internet form of an IPv4 address. Primary value is the IP address associated with the system's primary interface (<code>nodename == hostname</code>). Server-mode-only parameter. Note that using OWNER_IP has some performance impact, thus using a large number might not be advisable.
PATH	Path to DHCP data tables within the data store specified by the RESOURCE parameter. The value of the PATH keyword is specific to the RESOURCE.
RELAY_DESTINATIONS	String. Comma-separated list of host names and/or IP addresses of relay destinations. relay mode only parameter.
RELAY_HOPS	Integer. Max number of BOOTP relay hops before packet is dropped. Default is 4. Generic parameter.
RESCAN_INTERVAL	Integer. Number of minutes between automatic <code>dhcptab</code> rescans. Default is not to do rescans. server mode only parameter.
RESOURCE	Data store resource used. Use this parameter to name the public module. See the PATH keyword in dhcp_modules(5) .

RESOURCE_CONFIG	<p>String. The private layer provides for module-specific configuration information through the use of the RESOURCE_CONFIG keyword. See dhcp_modules(5).</p> <p>Providers can access RESOURCE_CONFIG using the configure function by specifying an optional service provider layer API function:</p> <pre>int configure(const char *configp);</pre> <p>If this function is defined by the public module provider, it is called during module load time by the private layer, with the contents of the RESOURCE_CONFIG string acquired by the administrative interface (in the case of the dhcpmgr, through the use of a public module-specific java bean extending the dhcpmgr to provide a configuration dialog for this information.</p>
RUN_MODE	<p>server or relay. Selects daemon run mode. Default is server.</p>
SECONDARY_SERVER_TIMEOUT	<p>Integer. The number of seconds a secondary server waits for a primary server to respond before responding itself. Default is 20 seconds. This is a server mode only parameter.</p>
UPDATE_TIMEOUT	<p>Integer. Number of seconds to wait for a response from the DNS server before timing out. If this parameter is present, the DHCP daemon updates DNS on behalf of DHCP clients, and waits the number of seconds specified for a response before timing out. You can use UPDATE_TIMEOUT without specifying a number to enable DNS updates with the default timeout of 15 seconds. If this parameter is not present, the DHCP daemon does not update DNS for DHCP clients.</p>
VERBOSE	<p>TRUE/FALSE. Toggles verbose mode, determining amount of status and error messages reported by the daemon. Default is FALSE. Set to TRUE only for debugging. Generic parameter.</p>

See Also [dhcpmgr\(1M\)](#), [in.dhcpd\(1M\)](#), [dhcp\(5\)](#), [dhcp_modules\(5\)](#)

Oracle Solaris Administration: IP Services

Name dhcptab – DHCP configuration parameter table

Description The dhcptab configuration table allows network administrators to organize groups of configuration parameters as macro definitions, which can then be referenced in the definition of other useful macros. These macros are then used by the DHCP server to return their values to DHCP and BOOTP clients.

The preferred method of managing the dhcptab is through the use of the `dhcpmgr(1M)` or `dhtadm(1M)` utility. The description of dhcptab entries included in this manual page is intended for informational purposes only, and should not be used to manually edit entries.

You can view the contents of the dhcptab using the DHCP manager's tabs for Macros and Options, or using the `dhtadm -P` command.

Syntax of dhcptab Entries The format of a dhcptab table depends on the data store used to maintain it. However, any dhcptab must contain the following fields in each record:

Name	This field identifies the macro or symbol record and is used as a search key into the dhcptab table. The name of a macro or symbol must consist of ASCII characters, with the length limited to 128 characters. Names can include spaces, except at the end of the name. The name is not case-sensitive.
Type	This field specifies the type of record and is used as a search key into the dhcptab. Currently, there are only two legal values for Type: <ul style="list-style-type: none"> <code>m</code> This record is a DHCP macro definition. <code>s</code> This record is a DHCP symbol definition. It is used to define vendor and site-specific options.
Value	This field contains the value for the specified type of record. For the <code>m</code> type, the value will consist of a series of symbol=value pairs, separated by the colon (:) character. For the <code>s</code> type, the value will consist of a series of fields, separated by a comma (,), which define a symbol's characteristics. Once defined, a symbol can be used in macro definitions.

Symbol Characteristics The Value field of a symbols definition contain the following fields describing the characteristics of a symbol:

Context	This field defines the context in which the symbol definition is to be used. It can have one of the following values:
Site	This symbol defines a site-specific option, codes 128-254.
Vendor=Client Class ...	This symbol defines a vendor-specific option, codes 1-254. The Vendor context takes ASCII string arguments which identify the client class that this vendor

option is associated with. Multiple client class names can be specified, separated by white space. Only those clients whose client class matches one of these values will see this option. For Sun machines, the Vendor client class matches the value returned by the command `uname -i` on the client, with periods replacing commas.

Code	This field specifies the option code number associated with this symbol. Valid values are 128-254 for site-specific options, and 1-254 for vendor-specific options.
Type	This field defines the type of data expected as a value for this symbol, and is not case-sensitive. Legal values are: ASCII NVT ASCII text. Value is enclosed in double-quotes ("). Granularity setting has no effect on symbols of this type, since ASCII strings have a natural granularity of one (1). BOOLEAN No value is associated with this data type. Presence of symbols of this type denote boolean TRUE, whereas absence denotes FALSE. Granularity and Maximum values have no meaning for symbols of this type. IP Dotted decimal form of an Internet address. Multi-IP address granularity is supported. NUMBER An unsigned number with a supported granularity of 1, 2, 4, and 8 octets. Valid NUMBER types are: UNNUMBER8, SNUMBER8, UNNUMBER16, SNUMBER16, UNNUMBER32, SNUMBER32, UNNUMBER64, and SNUMBER64. See dhcp_inittab(4) for details. OCTET Uninterpreted ASCII representation of binary data. The client identifier is one example of an OCTET string. Valid characters are 0-9, a-f, A-F. One ASCII character represents one nibble (4 bits), thus two ASCII characters are needed to represent an 8 bit quantity. The granularity setting has no effect on symbols of this type, since OCTET strings have a natural granularity of one (1). For example, to encode a sequence of bytes with decimal values 77, 82, 5, 240, 14, the option value would be encoded as 4d5205f00e. A macro which supplies a value for option code

78, SLP_DA, with a 0 Mandatory byte and Directory Agents at 192.168.1.5 and 192.168.0.133 would appear in the dhcptab as:

```
slpparams
Macro
:SLP_DA=00c0a80105c0a80085:
```

- Granularity** This value specifies how many objects of Type define a single instance of the symbol value. For example, the static route option is defined to be a variable list of routes. Each route consists of two IP addresses, so the Type is defined to be IP, and the data's granularity is defined to be 2 IP addresses. The granularity field affects the IP and NUMBER data types.
- Maximum** This value specifies the maximum items of Granularity which are permissible in a definition using this symbol. For example, there can only be one IP address specified for a subnet mask, so the Maximum number of items in this case is one (1). A Maximum value of zero (0) means that a variable number of items is permitted.

The following example defines a site-specific option (symbol) called `MystatRt`, of code 130, type IP, and granularity 2, and a Maximum of 0. This definition corresponds to the internal definition of the static route option (`StaticRt`).

```
MystatRt s Site,130,IP,2,0
```

The following example demonstrates how a SLP Service Scope symbol (`SLP_SS`) with a scope value of `happy` and mandatory byte set to 0 is encoded. The first octet of the option is the Mandatory octet, which is set either to 0 or 1. In this example, it is set to 0 (00). The balance of the value is the hexadecimal ASCII code numbers representing the name `happy`, that is, `6861707079`.

```
SLP_SS=006861707079
```

- Macro Definitions** The following example illustrates a macro defined using the `MystatRt` site option symbol just defined:

```
10netnis m :MystatRt=3.0.0.0 10.0.0.30:
```

Macros can be specified in the `Macro` field in DHCP network tables (see [dhcp_network\(4\)](#)), which will bind particular macro definitions to specific IP addresses.

Up to four macro definitions are consulted by the DHCP server to determine the options that are returned to the requesting client.

These macros are processed in the following order:

- Client Class** A macro named using the ASCII representation of the client class (e.g. `SUNW.Ultra-30`) is searched for in the dhcptab. If found, its

symbol/value pairs will be selected for delivery to the client. This mechanism permits the network administrator to select configuration parameters to be returned to all clients of the same class.

Network	A macro named by the dotted Internet form of the network address of the client's network (for example, 10.0.0.0) is searched for in the dhcptab. If found, its symbol/value pairs will be combined with those of the Client Class macro. If a symbol exists in both macros, then the Network macro value overrides the value defined in the Client Class macro. This mechanism permits the network administrator to select configuration parameters to be returned to all clients on the same network.
IP Address	This macro may be named anything, but must be specified in the DHCP network table for the IP address record assigned to the requesting client. If this macro is found in the dhcptab, then its symbol/value pairs will be combined with those of the Client Class macro and the Network macro. This mechanism permits the network administrator to select configuration parameters to be returned to clients using a particular IP address. It can also be used to deliver a macro defined to include "server-specific" information by including this macro definition in all DHCP network table entries owned by a specific server.
Client Identifier	A macro named by the ASCII representation of the client's unique identifier as shown in the DHCP network table (see dhcp_network(4)). If found, its symbol/value pairs are combined to the sum of the Client Class, Network, and IP Address macros. Any symbol collisions are replaced with those specified in the client identifier macro. The client mechanism permits the network administrator to select configuration parameters to be returned to a particular client, regardless of what network that client is connected to.

Refer to *Oracle Solaris Administration: IP Services* for more information about macro processing.

Refer to the `dhcp_inittab(4)` man page for more information about symbols used in Solaris DHCP.

See Also `dhcpgmr(1M)`, `dhtadm(1M)`, `in.dhcpd(1M)`, `dhcp_inittab(4)`, `dhcp_network(4)`, `dhcp(5)`
Oracle Solaris Administration: 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.

Name dialups – list of terminal devices requiring a dial-up password

Synopsis /etc/dialups

Description dialups is an ASCII file which contains a list of terminal devices that require a dial-up password. A dial-up password is an additional password required of users who access the computer through a modem or dial-up port. The correct password must be entered before the user is granted access to the computer. The set of ports that require a dial-up password are listed in the dialups file.

Each entry in the dialups file is a single line of the form:

terminal-device

where

terminal-device The full path name of the terminal device that will require a dial-up password for users accessing the computer through a modem or dial-up port.

The dialups file should be owned by the root user and the root group. The file should have read and write permissions for the owner (root) only.

Examples EXAMPLE 1 A sample dialups file.

Here is a sample dialups file:

```
/dev/term/a
/dev/term/b
/dev/term/c
```

Files /etc/d_passwd dial-up password file
/etc/dialups list of dial-up ports requiring dial-up passwords

See Also [d_passwd\(4\)](#)

Name dir_ufs, dir – format of ufs directories

Synopsis #include <sys/param.h>
 #include <sys/types.h>
 #include <sys/fs/ufs/fsdir.h>

Description A directory consists of some number of blocks of DIRBLKSIZ bytes, where DIRBLKSIZ is chosen such that it can be transferred to disk in a single atomic operation, for example, 512 bytes on most machines.

Each DIRBLKSIZ-byte block contains some number of directory entry structures, which are of variable length. Each directory entry has a `struct direct` at the front of it, containing its inode number, the length of the entry, and the length of the name contained in the entry. These entries are followed by the name padded to a 4 byte boundary with null bytes. All names are guaranteed null-terminated. The maximum length of a name in a directory is MAXNAMLEN.

```
#define DIRBLKSIZ                DEV_BSIZE
#define MAXNAMLEN                256
struct direct {
    ulong_t  d_ino;                /* inode number of entry */
    ushort_t d_reclen;            /* length of this record */
    ushort_t d_namlen;            /* length of string in d_name */
    char     d_name[MAXNAMLEN + 1]; /* maximum name length */
};
```

Attributes See [attributes\(5\)](#) for a description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Uncommitted

See Also [attributes\(5\)](#), [ufs\(7FS\)](#)

Name d_passwd – dial-up password file

Synopsis /etc/d_passwd

Description A dial-up password is an additional password required of users who access the computer through a modem or dial-up port. The correct password must be entered before the user is granted access to the computer.

d_passwd is an ASCII file which contains a list of executable programs (typically shells) that require a dial-up password and the associated encrypted passwords. When a user attempts to log in on any of the ports listed in the dialups file (see [dialups\(4\)](#)), the login program looks at the user's login entry stored in the passwd file (see [passwd\(4\)](#)), and compares the login shell field to the entries in d_passwd. These entries determine whether the user will be required to supply a dial-up password.

Each entry in d_passwd is a single line of the form:

```
login-shell:password:
```

where

login-shell The name of the login program that will require an additional dial-up password.

password An encrypted password. Users accessing the computer through a dial-up port or modem using *login-shell* will be required to enter this password before gaining access to the computer.

d_passwd should be owned by the root user and the root group. The file should have read and write permissions for the owner (root) only.

If the user's login program in the passwd file is not found in d_passwd or if the login shell field in passwd is empty, the user must supply the default password. The default password is the entry for /usr/bin/sh. If d_passwd has no entry for /usr/bin/sh, then those users whose login shell field in passwd is empty or does not match any entry in d_passwd will not be prompted for a dial-up password.

Dial-up logins are disabled if d_passwd has only the following entry:

```
/usr/bin/sh:*
```

Examples EXAMPLE 1 Sample d_passwd file.

Here is a sample d_passwd file:

```
/usr/lib/uucp/uucico:q.mJzTnu8icF0:  
/usr/bin/csh:6k/7KCFRPNVXg:  
/usr/bin/ksh:9df/FDf.4jkRt:  
/usr/bin/sh:41FuGVzGcDJlw:
```

Generating An Encrypted Password

The `passwd` (see [passwd\(1\)](#)) utility can be used to generate the encrypted password for each login program. `passwd` generates encrypted passwords for users and places the password in the `shadow` (see [shadow\(4\)](#)) file. Passwords for the `d_passwd` file will need to be generated by first adding a temporary user id using `useradd` (see [useradd\(1M\)](#)), and then using [passwd\(1\)](#) to generate the desired password in the `shadow` file. Once the encrypted version of the password has been created, it can be copied to the `d_passwd` file.

For example:

1. Type `useradd tempuser` and press Return. This creates a user named `tempuser`.
2. Type `passwd tempuser` and press Return. This creates an encrypted password for `tempuser` and places it in the `shadow` file.
3. Find the entry for `tempuser` in the `shadow` file and copy the encrypted password to the desired entry in the `d_passwd` file.
4. Type `userdel tempuser` and press Return to delete `tempuser`.

These steps must be executed as the root user.

Files

<code>/etc/d_passwd</code>	dial-up password file
<code>/etc/dialups</code>	list of dial-up ports requiring dial-up passwords
<code>/etc/passwd</code>	password file
<code>/etc/shadow</code>	shadow password file

See Also [passwd\(1\)](#), [useradd\(1M\)](#), [dialups\(4\)](#), [passwd\(4\)](#), [shadow\(4\)](#)

Warnings When creating a new dial-up password, be sure to remain logged in on at least one terminal while testing the new password. This ensures that there is an available terminal from which you can correct any mistakes that were made when the new password was added.

Name driver – user administered driver.conf

Synopsis /etc/driver/drv

Description The directory /etc/driver/drv is a location where administrative driver.conf customizations can be placed.

Separating vendor driver.conf files and administratively-customized driver.conf files allows the system to be upgraded with new vendor driver.conf files without overwriting the administrative customizations.

See Also [prtconf\(1M\)](#), [driver.conf\(4\)](#), [ddi_prop_exists\(9F\)](#), [ddi_prop_lookup\(9F\)](#)

Name driver.conf – driver configuration files

Synopsis driver.conf

Description Driver configuration files provide values for device properties. The values override values provided by the devices themselves. Most modern devices provide enough property values to make a driver configuration file unnecessary.

The system associates a driver with its configuration file by name. For example, a driver in `/usr/kernel/drv` called `wombat` has the driver configuration file `wombat.conf`, also stored in `/usr/kernel/drv`, associated with it. On systems capable of support 64-bit drivers, the driver configuration file should be placed in the directory in which the 32-bit driver is (or would be) located, even if only a 64-bit version is provided. For example, a 64-bit driver stored in `/usr/kernel/drv/sparcv9` stores its driver configuration file in `/usr/kernel/drv`.

The value of the `name` property is the node name. In a `driver.conf` file, where the generic node name and `compatible` property associated with a self-identifying devices are typically not used, the node name must be a binding name. The binding name is the name chosen by the system to bind a driver to the device. The binding name is either an alias associated with the driver established by `add_drv(1M)` or the driver name itself.

The syntax of a single entry in a driver configuration file takes one of three forms:

```
name="node name" parent="parent name" [property-name=value ...];
```

In this form, the parent name can be either the binding name of the parent nexus driver or a specific full pathname, beginning with a slash (`/`) character, identifying a specific instance of a parent bus. If a binding name is used then all parent nodes bound to that driver match. A generic name (for example, `pci`) is not a valid binding name even though it can appear in the full pathname of all intended parents.

Alternatively, the parent can be specified by the type of interface it presents to its children.

```
name="node name" class="class name" [property-name=value ...];
```

For example, the driver for the SCSI host adapter can have different names on different platforms, but the target drivers can use class `scsi` to insulate themselves from these differences.

Entries of either form above correspond to a device information (`devinfo`) node in the kernel device tree. Each node has a `name` which is usually the name of the driver, and a `parent` name which is the name of the parent `devinfo` node to which it will be connected. Any number of name-value pairs can be specified to create properties on the prototype `devinfo` node. These properties can be retrieved using the DDI property interfaces (for example, `ddi_prop_get_int(9F)` and `ddi_prop_lookup(9F)`). The prototype `devinfo` node specification must be terminated with a semicolon (`;`).

The third form of an entry is simply a list of properties.

```
[property-name=value ...];
```

A property created in this way is treated as global to the driver. It can be overridden by a property with the same name on a particular `devinfo` node, either by creating one explicitly on the prototype node in the `driver.conf` file or by the driver.

Items are separated by any number of newlines, SPACE or TAB characters.

The configuration file can contain several entries to specify different device configurations and parent nodes. The system can call the driver for each possible prototype `devinfo` node, and it is generally the responsibility of the drivers [probe\(9E\)](#) routine to determine if the hardware described by the prototype `devinfo` node is really present.

Property names must not violate the naming conventions for Open Boot PROM properties or for IEEE 1275 names. In particular, property names should contain only printable characters, and should not contain at-sign (@), slash (/), backslash (\), colon (:), or square brackets ([]). Property values can be decimal integers or strings delimited by double quotes ("). Hexadecimal integers can be constructed by prefixing the digits with 0x.

A comma separated list of integers can be used to construct properties whose value is an integer array. The value of such properties can be retrieved inside the driver using [ddi_prop_lookup_int_array\(9F\)](#).

Comments are specified by placing a # character at the beginning of the comment string, the comment string extends for the rest of the line.

In addition to the vendor `driver.conf` files provided by the system and driver providers in `/kernel` and `/platform`, user-administered `driver.conf` files may be added in `/etc/driver/drv`. Files in `/etc/driver/drv` are available for local customizations to supplement the vendor driver configuration.

The format of an `/etc/driver/drv` `driver.conf` file is identical to that of a vendor `driver.conf` file.

For each driver, the vendor and admin `driver.conf` files are merged together and made available to the driver and system, as a merged set of device specifications with per-device properties and a global set of properties visible to every device instance bound to this driver.

These merged per-device and global properties are visible to an instance of the driver via the [ddi_prop_lookup\(9F\)](#) and related interfaces automatically, with no intervention by the driver required.

A driver that does provide configurable options by way of `driver.conf` should keep this property merge behavior in mind. Inverting the meaning of an option from one release to the next could cause problems for users that may have configured a system with the previous interpretation of that option.

A driver which has converted its configurable options from one revision to another may find it useful to convert older option configurations to the new supported set. The system maintains two separate property lists providing the default vendor properties and the admin customizations.

Each property originally derived from the vendor `driver.conf` file and then merged with an updated value will be entered both the vendor and admin property lists. Each list may be searched via `ddi_prop_lookup(9F)` as either the `DDI_PROP_VENDOR` or `DDI_PROP_ADMIN`, respectively. The property on the vendor list records the property's original value, while the property on the admin list provides the customization.

There is no property entry on the vendor list for a property derived solely from the admin `driver.conf`. Such properties have entries on the admin list only, providing the admin-specified value.

Supplementing the Vendor Driver Configuration

First, it is important to understand the format of a `driver.conf` entry. Each entry can be one of two types, specifying either a device information (`devinfo`) node as a parent specification or a set of properties globally available to all device instances bound to this driver.

An entry defining a device info node must specify the node's parent as either the binding name of the parent nexus driver, a pathname or by the class of interface presented to its children. Other properties on the parent specification define qualifiers required to correctly identify the specific device, and depend upon the parent driver or class of interface. An example of such a device qualifier property is 'target', used by SCSI hba adapters to identifier a specific device on its bus. Such qualifiers must be retained unchanged so that the system will continue to properly enumerate that device instance.

To modify one or more property values on such a parent specification, the targeted parent specification in the vendor `driver.conf` must be duplicated in the admin `driver.conf`, preserving each device qualifying property. Then the admin entry can be customized, supplementing it with additional or modified configurable options.

Parent specifications in the admin `driver.conf` that do not correspond to a match in the vendor `driver.conf` file are added to the list of device information nodes available to be potentially enumerated by the system for that driver.

The second type of `driver.conf` entry defines global properties for that driver. The admin `driver.conf` file can both update existing properties in the vendor `driver.conf` file with new values, and provide new global property name-value pairs available to all device instances bound to that driver.

Considerations for Driver Writers

The process of upgrading a Solaris system from one release to another involving delivering new versions of drivers including that driver's vendor `driver.conf`. It is desired that a system's earlier configuration should continue to work as before, with the new drivers and vendor `driver.conf` files together with the admin customized `driver.conf` files.

For a driver to work well under this model, each driver needs to be properly designed to present a disciplined set of configurable options. It would be best to carefully define a driver's options with this in mind and to fully describe the model presented in the driver's documentation or man page.

If a driver makes a change in its configuration options that would invalidate or supersede the admin settings, the driver should make the effort to discover the admin settings via the prior options and honor them.

For example, let's say a driver supports a timeout configuration in units of seconds. A new version of the driver now supports a finer timeout granularity in units of milliseconds. The new property should be named so it can be distinguished from the first. The driver can then look up the earlier property on the admin list and if found, continue to honor it.

```
/*
 * Has the timeout been locally configured using the
 * prior option of timeout in units of seconds?
 */
if (ddi_prop_lookup_int(DDI_DEV_T_ANY, dip,
    DDI_PROP_ADMIN, "timeout",) ==
    DDI_PROP_SUCCESS) {
    if (n != 1) {
        ddi_prop_free(ivalues);
        return (EINVAL);
    }
    /* yes - convert our working timeout accordingly */
    dip->ms_timeout = 1000 * ivalues[0];
    /* record the new parameter setting for confirmation */
    (void) ddi_prop_update_int(DDI_DEV_T_NONE,
        dip, "ms-timeout", dip->ms_timeout);
    ddi_prop_free(ivalues);
}
}
```

Examples EXAMPLE 1 Using a Configuration File for a PCI Bus Frame Buffer

The following is an example of a configuration file called `ACME,simple.conf` for a PCI bus frame buffer called `ACME,simple`.

```
#
# Copyright (c) 1993, by ACME Fictitious Devices, Inc.
#
#ident "@(#)ACME,simple.conf 1.3 1999/09/09"
```


EXAMPLE 1 Using a Configuration File for a PCI Bus Frame Buffer (Continued)

```
name="ACME,simple" class="pci" unit-address="3,1"
    debug-mode=12;
```

This example creates a prototype devinfo node called `ACME,simple` under all parent nodes of class `pci`. The node has device and function numbers of 3 and 1, respectively; the property `debug-mode` is provided for all instances of the driver.

EXAMPLE 2 Using a Configuration File for a Pseudo Device Driver

The following is an example of a configuration file called `ACME,example.conf` for a pseudo device driver called `ACME,example`.

```
#
# Copyright (c) 1993, ACME Fictitious Devices, Inc.
#
#ident "@(#)ACME,example.conf 1.2 93/09/09"
name="ACME,example" parent="pseudo" instance=0
    debug-level=1;

name="ACME,example" parent="pseudo" instance=1;

whizzy-mode="on";
debug-level=3;
```

This creates two devinfo nodes called `ACME,example` which attaches below the pseudo node in the kernel device tree. The `instance` property is only interpreted by the pseudo node, see [pseudo\(4\)](#) for further details. A property called `debug-level` is created on the first devinfo node which has the value 1. The `example` driver is able to fetch the value of this property using [ddi_prop_get_int\(9F\)](#).

Two global driver properties are created, `whizzy-mode` (which has the string value "on") and `debug-level` (which has the value 3). If the driver looks up the property `whizzy-mode` on either node, it retrieves the value of the global `whizzy-mode` property ("on"). If the driver looks up the `debug-level` property on the first node, it retrieves the value of the `debug-level` property on that node (1). Looking up the same property on the second node retrieves the value of the global `debug-level` property (3).

EXAMPLE 3 Modifying a Driver Global Property

The `bge.conf` provides default values for the receive and xmit rings.

```
#
# The properties below represents the number of receive and send ring used.
# For BCM5705, BCM5782, etc, there are only 1 receive ring and 1 send ring.
# Otherwise, there can be up to 16 receive rings and 4 send rings.
#
```

EXAMPLE 3 Modifying a Driver Global Property *(Continued)*

```
bge-rx-rings      = 16;
bge-tx-rings      = 1;
```

To customize the `bge-tx-rings` value, place a `bge.conf` file in `/etc/driver/drv` with the following line:

```
bge-tx-rings      = 2;
```

When the `bge` driver is next loaded, the updated value can be observed with `prtconf`:

```
pci108e,534d (pci14e4,16a7), instance #0
  System software properties:
    name='bge-known-subsystems' type=int items=16
    name='bge-rx-rings' type=int items=1
      value=00000010
    name='bge-tx-rings' type=int items=1
      value=00000002
```

Additionally, `prtconf -u` can be used to display both the original default and the updated `bge-tx-rings` property values:

```
Admin global properties:
  name='bge-tx-rings' type=int items=1
    value=00000002
Vendor global properties:
  name='bge-tx-rings' type=int items=1
    value=00000001
```

EXAMPLE 4 Modifying configurable values on a specific device

To modify the configurable parameter `'retries'` on an `sd` device at target `0`, lun `0` and `'queue-max'` on the device target `1`, lun `0`, place an `sd.conf` file in `/etc/driver/drv` with the following lines:

```
name="sd" class="scsi" target=0 lun=0 retries=4;
name="sd" class="scsi" target=1 lun=0 queue-max=16;
```

The updated values can be observed with `prtconf`:

```
sd, instance #1
  System properties:
    name='lun' type=int items=1
      value=00000000
    name='target' type=int items=1
      value=00000000
    name='class' type=string items=1
      value='scsi'
    name='retries' type=int items=1
```

EXAMPLE 4 Modifying configurable values on a specific device *(Continued)*

```

        value=00000004
        name='ddi-devid-registrant' type=int items=1
        value=00000001

sd, instance #2
System properties:
    name='lun' type=int items=1
    value=00000000
    name='target' type=int items=1
    value=00000001
    name='class' type=string items=1
    value='scsi'
    name='queue-max' type=int items=1
    value=00000010
    name='ddi-devid-registrant' type=int items=1
    value=00000001

```

With `prtconf -u`, the admin property values are displayed. The vendor property list in this case contains no properties as the vendor `driver.conf` contained no specification for such properties.

```

sd, instance #1
Admin properties:
    name='retries' type=int items=1
    value=00000004

sd, instance #2
Admin properties:
    name='queue-max' type=int items=1
    value=00000010

```

EXAMPLE 5 Adding a New Device Instance

For purposes of illustration, suppose that the vendor `sd.conf` contains only the following parent specification:

```
name="sd" class="scsi" target=0 lun=0 max-retries=4;
```

and that it is desired to also support a `target=1` device instance. Further suppose that `target=0` should be configured with the `max-retries` parameter set to 10 and queueing set to 32, and that `target=1` with `max-retries` to 10 and queueing to 64. Place the following lines in the `sd.conf` file in `/etc/driver/drv`:

```
name="sd" class="scsi" target=0 lun=0 max-retries=8 queue=32;
name="sd" class="scsi" target=1 lun=0 max-retries=10 queue=64;
```

EXAMPLE 5 Adding a New Device Instance (Continued)

These changes can be observed with `prtconf`. For target 0, the vendor list contains the vendor setting for the number of retries and the updated value in the admin list. There was no specification for this target as delivered, so the vendor list for target 1 is empty and all specified parameters from the admin list are displayed.

```
sd, instance #1
System properties:
  name='max-retries' type=int items=1
  value=00000008
  name='lun' type=int items=1
  value=00000000
  name='target' type=int items=1
  value=00000000
  name='class' type=string items=1
  value='scsi'
  name='queue' type=int items=1
  value=00000020
  name='ddi-devid-registrant' type=int items=1
  value=00000001
Admin properties:
  name='queue' type=int items=1
  value=00000020
  name='max-retries' type=int items=1
  value=00000008
Vendor properties:
  name='max-retries' type=int items=1
  value=00000004

sd, instance #2 (driver not attached)
System properties:
  name='queue' type=int items=1
  value=00000040
  name='max-retries' type=int items=1
  value=0000000a
  name='lun' type=int items=1
  value=00000000
  name='target' type=int items=1
  value=00000001
  name='class' type=string items=1
  value='scsi'
  name='ddi-devid-registrant' type=int items=1
  value=00000001
Admin properties:
  name='queue' type=int items=1
  value=00000040
```

EXAMPLE 5 Adding a New Device Instance *(Continued)*

```

name='max-retries' type=int items=1
  value=0000000a
name='lun' type=int items=1
  value=00000000
name='target' type=int items=1
  value=00000001
name='class' type=string items=1
  value='scsi'

```

See Also [add_drv\(1M\)](#), [driver\(4\)](#), [pci\(4\)](#), [pseudo\(4\)](#), [sbus\(4\)](#), [scsi\(4\)](#), [probe\(9E\)](#), [ddi_getlongprop\(9F\)](#), [ddi_getprop\(9F\)](#), [ddi_getproplen\(9F\)](#), [ddi_prop_get_int\(9F\)](#), [ddi_prop_lookup\(9F\)](#), [ddi_prop_op\(9F\)](#)

Writing Device Drivers

Warnings To avoid namespace collisions between multiple driver vendors, it is strongly recommended that the *name* property of the driver should begin with a vendor-unique string. A reasonably compact and unique choice is the vendor over-the-counter stock symbol.

Notes The [update_drv\(1M\)](#) command should be used to prompt the kernel to reread `driver.conf` files. Using [modunload\(1M\)](#) to update `driver.conf` continues to work in release 9 of the Solaris operating environment, but the behavior will change in a future release.

It is not currently possible to either remove or undefine a property, or to remove a parent specification, defined in a vendor `driver.conf` file with an addition to an `/etc/driver/drvdriver.conf` file.

Name ds.log – Availability Suite data services log file

Description The `/var/adm/ds.log` file contains the Availability Suite data services command log. The administration commands log activities to the file in the format:

```
date time product: message
```

Note that when the size of the log file exceeds 10 Mbytes, `ds.log` is renamed `/var/adm/ds.log.bak` and a new `/var/adm/ds.log` file is created.

The `ds.log` fields are:

date

The date format is *mmm nn*, where *mmm* is the local three-character abbreviation for the month and *nn* is the day of the month on which the event occurred.

time

The time of the event, in *hh:mm:ss* format.

product

A product code that identifies which component of the data services produced the event. The code is separated from the message that follows by a colon (:) and a space.

message

A message that can extend over more than one line describing the event that occurred. The second or following lines are not prefixed by the date, time, and product code strings.

Examples The example below shows sample `ds.log` file content:

```
Jan 25 05:26:17 ii: iiboot suspend cluster tag <none>
Jan 25 05:32:02 ii: iiboot resume cluster tag <none>
Jan 25 05:32:04 sv: svboot: resume /dev/vx/rdisk/bigmaster
Jan 25 05:32:04 sv: svboot: resume /dev/vx/rdisk/bigshadow
Jan 25 05:32:04 sv: svboot: resume /dev/vx/rdisk/mstvxfs
Jan 25 05:32:04 sv: svboot: resume /dev/vx/rdisk/master01
```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	x86
Availability	storage/avs/avs-cache-management
Interface Stability	Committed

See Also [iiadm\(1M\)](#), [sndradm\(1M\)](#), [svadm\(1M\)](#), [attributes\(5\)](#)

Name ethers – Ethernet address to hostname database or domain

Description The ethers file is a local source of information about the (48-bit) Ethernet addresses of hosts on the Internet. The ethers file can be used in conjunction with or instead of other ethers sources, including the NIS maps ethers.byname and ethers.byaddr, or Ethernet address data stored on an LDAP server. Programs use the ethers(3SOCKET) routines to access this information.

The ethers file has one line for each host on an Ethernet. The line has the following format:

Ethernet-address official-host-name

Items are separated by any number of SPACE and/or TAB characters. A '#' indicates the beginning of a comment extending to the end of line.

The standard form for Ethernet addresses is "x:x:x:x:x" where x is a hexadecimal number between 0 and ff, representing one byte. The address bytes are always in network order. Host names may contain any printable character other than SPACE, TAB, NEWLINE, or comment character.

Files /etc/ethers file listing Ethernet hosts

See Also ethers(3SOCKET), hosts(4), nsswitch.conf(4)

Name exec_attr – execution profiles database

Synopsis /etc/security/exec_attr

Description /etc/security/exec_attr is a local database that specifies the execution attributes associated with profiles. The exec_attr file can be used with other sources for execution profiles, including the exec_attr NIS map. Programs use the [getexecattr\(3C\)](#) routines to access this information.

The search order for multiple execution profile sources is specified in the /etc/nsswitch.conf file, as described in the [nsswitch.conf\(4\)](#) man page. The search order follows the entry for [prof_attr\(4\)](#).

A profile is a logical grouping of authorizations and commands that is interpreted by a profile shell to form a secure execution environment. The shells that interpret profiles are pfcsh, pfksh, and pfish. See the [pfsh\(1\)](#) man page. Each user's account is assigned zero or more profiles in the [user_attr\(4\)](#) database file.

Each entry in the exec_attr database consists of one line of text containing seven fields separated by colons (:). Line continuations using the backslash (\) character are permitted. The basic format of each entry is:

name:policy:type:res1:res2:id:attr

name The name of the profile. Profile names are case-sensitive.

policy The security policy that is associated with the profile entry. The valid policy type is solaris. The solaris policy recognizes privileges. See [privileges\(5\)](#).

type The type of object defined in the profile. The valid type is cmd. The cmd type specifies that the ID field is a command that would be executed by a shell.

res1 The characters R0 in this field indicate it is read only and not modifiable by the tools that update this database.

res2 Reserved for future use.

id A string that uniquely identifies the object described by the profile. For a profile of type cmd, the id is either the full path to the command or the asterisk (*) symbol, which is used to allow all commands. An asterisk that replaces the filename component in a pathname indicates all files in a particular directory.

To specify arguments, the pathname should point to a shell script that is written to execute the command with the desired argument. In a Bourne shell, the effective UID is reset to the real UID of the process when the effective UID is less than 100 and not equal to the real UID. Depending on the euid and egid values, Bourne shell limitations might make other shells preferable. To prevent the effective UIDs from being reset to real UIDs, you can start the script with the -p option.


```
#!/bin/sh -p
```

attr An optional list of semicolon-separated (;) key-value pairs that describe the security attributes to apply to the object upon execution. Zero or more keys may be specified. The list of valid key words depends on the policy enforced. The following key words are valid: `eid`, `uid`, `egid`, `gid`, `privs`, and `limitprivs`.

`eid` and `uid` contain a single user name or a numeric user ID. Commands designated with `eid` run with the effective UID indicated, which is similar to setting the `setuid` bit on an executable file. Commands designated with `uid` run with both the real and effective UIDs. Setting `uid` may be more appropriate than setting the `eid` on privileged shell scripts.

`egid` and `gid` contain a single group name or a numeric group ID. Commands designated with `egid` run with the effective GID indicated, which is similar to setting the `setgid` bit on a file. Commands designated with `gid` run with both the real and effective GIDs. Setting `gid` may be more appropriate than setting `gid` on privileged shell scripts.

`privs` contains a privilege set which will be added to the inheritable set prior to running the command.

`limitprivs` contains a privilege set which will be assigned to the limit set prior to running the command.

`privs` and `limitprivs` are only valid for the `solaris` policy.

Examples EXAMPLE 1 Using Privileges

The following example shows the `audit` command specified in the Audit Control profile to execute with the `proc_owner` privilege:

```
Audit Control:solaris:cmd:::/usr/sbin/audit:privs=proc_owner
```

Files /etc/nsswitch.conf

/etc/user_attr

/etc/security/exec_attr Locally added entries. Make sure that the shipped header remains intact.

/etc/security/exec_attr.d/* Entries added by package installation.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	See below.

The command-line syntax is Committed. The output is Uncommitted.

Notes Because the list of legal keys is likely to expand, any code that parses this database must be written to ignore unknown key-value pairs without error. When any new keywords are created, the names should be prefixed with a unique string, such as the company's stock symbol, to avoid potential naming conflicts.

The following characters are used in describing the database format and must be escaped with a backslash if used as data: colon (:), semicolon (;), equals (=), and backslash (\).

Authorizations required to set various fields are listed:

```

name      profile name          solaris.profile.cmd.manage
policy    security policy        solaris.profile.cmd.manage
type      type of object defined solaris.profile.cmd.manage
           in the profile
id full path of command      solaris.profile.cmd.manage
attr security attributes of the command

euid      euid of the command     solaris.profile.cmd.setuid
           process
uid       uid of the command     solaris.profile.cmd.setuid
           process
egid      egid of the command   solaris.group.assign/delegate
           process
gid       gid of the command   solaris.group.assign/delegate
           process
privs     privileges added to
           the inheritable set of solaris.privilege.assign/delegate
           privileges for the
           command
limitprivs privileges assigned to
           the limit set of      solaris.privilege.assign/delegate
           privileges for the
           command

```

The value of `limitprivs` that can be set by an authorized user for a given command are limited to the `limitprivs` privileges that are granted to the user.

The value of the `privs` that can be set by an authorized user for a given command are limited to the `defaultprivs` privileges granted to the user.

The `solaris.privilege.assign` authorization allows the authorized user to grant any privilege to a user. The `solaris.privilege.delegate` allows the authorized user to grant privileges from the user's privilege sets. See [group\(4\)](#) for more information on `solaris.group.assign/delegate`.

See Also [auths\(1\)](#), [profiles\(1\)](#), [roles\(1\)](#), [sh\(1\)](#), [makedbm\(1M\)](#), [getauthattr\(3C\)](#), [getexecattr\(3C\)](#), [getprofattr\(3C\)](#), [getuserattr\(3C\)](#), [kva_match\(3C\)](#), [auth_attr\(4\)](#), [group\(4\)](#), [prof_attr\(4\)](#), [user_attr\(4\)](#), [attributes\(5\)](#), [privileges\(5\)](#)

Name fd – file descriptor files

Description These files, conventionally called `/dev/fd/0`, `/dev/fd/1`, `/dev/fd/2`, and so on, refer to files accessible through file descriptors. If file descriptor *n* is open, these two system calls have the same effect:

```
fd = open("/dev/fd/n", mode);  
fd = dup(n);
```

On these files [creat\(2\)](#) is equivalent to `open`, and `mode` is ignored. As with `dup`, subsequent reads or writes on `fd` fail unless the original file descriptor allows the operations.

For convenience in referring to standard input, standard output, and standard error, an additional set of names is provided: `/dev/stdin` is a synonym for `/dev/fd/0`, `/dev/stdout` for `/dev/fd/1`, and `/dev/stderr` for `/dev/fd/2`.

See Also [creat\(2\)](#), [dup\(2\)](#), [open\(2\)](#)

Diagnostics [open\(2\)](#) returns `-1` and `EBADF` if the associated file descriptor is not open.

Name fdi – HAL device information file format

Synopsis /usr/share/lib/xml/dtd/fdi.dtd.1

Description The hardware abstraction layer facility, described in [hal\(5\)](#), uses an XML-based file format to merge arbitrary properties onto device objects. The way device information files works is that once all physical properties are merged onto a device object, it is tried against the set of installed device information files. Device information files are used for both merging facts and policy settings for devices.

Each device information file has a number of match directives that are tested against the properties of the device object. The directives have the form:

```
<match key="property" [string|int|bool|..]="value">
```

If all the match directives pass, then the device information can include the following property directives in the form:

```
<[merge|append|prepend] key="property" type="[string|int|bool|..]">
```

These directives are used to merge new properties or append to existing properties on the device object. Any previously property stemming from device detection can be overridden by a device information file.

The match, merge, append, and prepend directives require that the key attribute be either a property name on the device object in question or a path to a property on another device object. The path to a property is expressed either through direct specification of the UDI, such as /org/freedesktop/Hal/devices/computer:foo.bar or through indirect references such as "@info.parent:baz", meaning that the device object specified by the UDI in the string property "info.parent" should be used to query the property "baz". It is also possible to use multiple indirections. For example, for a volume on a USB memory stick, the indirection "@block.storage_device:@storage.physical_device:usb.vendor_id" references the "usb.vendor_id" property on the device object representing the USB interface.

When the property to match has been determined, the following attributes can be used within the "match" tag:

string	Match a string property. For example, <match key="foo.bar" string="baz"> matches only if "foo.bar" is a string property assuming the value "baz".
int	Match an integer property
uint64	Match property with the 64-bit unsigned type
bool	Match a boolean property
double	Match a property of type double

<code>exists</code>	Used as <code><match key="foo.bar" exists="true"></code> . This attribute can be used with "true" and "false", respectively to match when a property exists or does not exist.
<code>empty</code>	This attribute can only be used on string properties with "true" and "false". The semantics for "true" is to match only when the string is non-empty.
<code>is_absolute_path</code>	Matches only when a string property represents an absolute path (the path does not have to exist). This attribute can be used with "true" or "false".
<code>is_ascii</code>	Matches only when a string property contains ASCII characters. This attribute can be used with "true" or "false".
<code>compare_lt</code>	This attribute can be used with <code>int</code> , <code>uint64</code> , <code>double</code> and <code>string</code> properties to compare with a constant. It matches when the given property is less than the given constant using the default ordering.
<code>compare_le</code>	Similar to <code>compare_lt</code> , but matches when the given property is less than or equal than the given constant using the default ordering.
<code>compare_gt</code>	Similar to <code>compare_lt</code> , but matches when the given property is greater than the given constant using the default ordering. Similar to <code>compare_lt</code> , but matches when the given property is greater than or equal than the given constant using the default ordering.
<code>contains</code>	This attribute can only be used with <code>string</code> and <code>strlist</code> (string list). For a string key, this matches when the property contains the given (sub)string. For a string list, this matches if the given string matches a item in the list.
<code>contains_ncase</code>	Similar to <code>contains</code> , but the property and the given key are converted to lowercase before it is checked.

The `merge`, `append`, and `prepend` directives all require the attribute type which specifies what is to be merged. The following values are supported:

<code>string</code>	The value is copied to the property. For example, <code><merge key="foo bar" type="string"> baz</merge></code> merges the value "baz" into the property "foo.bar".
<code>strlist</code>	For <code>merge</code> , the value is copied to the property and the current property is overwritten. For <code>append</code> and <code>prepend</code> , the value is appended or prepended to the list as a new item.
<code>bool</code>	This attribute can merge the values "true" or "false"

<code>int</code>	Merges an integer
<code>uint64</code>	Merges an unsigned 64-bit integer
<code>double</code>	Merges a double precision floating point number
<code>copy_property</code>	Copies the value of a given property; supports paths with direct and indirect UDI's. For example, <code><merge key="foo.bar" type="copy_property">@info.parent:baz.bat</merge></code> merges the value of the property "baz.bat" on the device object with the UDI from the property "info.parent" into the property "foo.bar" on the device object being processed.

The `remove` directive requires only a key and can be used with all keys. For `strlist`, there is also a special syntax to remove a item from the string list. For example, to remove item "bla" from property "foo.bar", use the following syntax:

```
<remove key="foo.bar" type="strlist">bla</merge>
```

Device Information files are stored in the following standard hierachy with the following default top level directories `information`, `policy` and `preprobe`:

<code>information</code>	Device information files to merge device information.
<code>10freedesktop</code>	Device information files included with the hal tarball.
<code>20thirdparty</code>	Device information files from the device manufacturer and installed from media accompanying the hardware.
<code>30user</code>	Device information for specific devices.
<code>policy</code>	Device information files to merge policy propertys
<code>10osvendor</code>	Device information files included with the hal tarball and supplied by the operating system vendor for policy rules
<code>20thirdparty</code>	Policy rules from the device manufacturer and installed from media accompanying the hardware
<code>30user</code>	Policy rules for specific devices.
<code>preprobe</code>	Device information files to merge information before probe devices.
<code>10osvendor</code>	Device information files included with the hal tarball and supplied by the operating system vendor.
<code>20thirdparty</code>	Device information files from the device manufacturer and installed from media accompanying the hardware.
<code>30user</code>	Device information for specific devices.

All device information files are matched for every hal device object.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/hal
Interface Stability	Volatile

See Also [hald\(1M\)](#), [attributes\(5\)](#), [hal\(5\)](#), [locale\(5\)](#), [smf\(5\)](#)

Name flash_archive – format of flash archive

Synopsis flash_archive

Description 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. The machine that contains a flash archive is referred to as a *master* system. A machine that receives a copy of a flash archive is called a *clone* system.

There are two types of flash archives: full and differential. A full archive is used for initial installation or whenever a complete, fresh installation is called for. A differential archive is used to update an installation. A full archive contains all of the files from a master and overwrites the installed software on a clone completely. A differential archive contains only the differences between the software on a master and on a clone. These differences include new files, changed files, and deleted files. (These will be deleted on clones, as well). Installation of a differential archive is faster and consumes fewer resources than installation of a full archive.

Flash archives are monolithic files containing both archive identification information and the actual files that have been copied from a master system and that will be extracted onto a clone system. The standard extension for a flash archive is `.flar`.

The flash archive is laid out in the following sections:

- archive cookie
- archive identification
- manifest (for differential archives only)
- predeployment
- postdeployment
- reboot
- summary
- user-defined (optional)
- archive files

The only assumptions regarding section number and placement that an application processing the archive can make is that there is an identification section located immediately after the archive cookie and that the last section in the archive is an archive files section.

These sections are described in the following subsections.

Archive Cookie The very beginning of the archive contains a cookie, which serves to identify the file as a flash archive. It is also used by the deployment code for identification and validation purposes.

The case-sensitive, newline-terminated cookie that identifies version 1.*n* flash archives, is `FLaSH-aRcHiVe-1.n`, where *n* is an integer in the range 0 through 9.

The archive version is designed to allow for the future evolution of the flash archive specification while allowing applications that process flash archives to determine whether specific archives are of a format that can be handled correctly. The archive version is a number of the form *x.y*, where *x* is the major version number, and *y* is the minor version number.

When an application encounters a flash archive with an unknown major version number, it should issue an error message and exit.

Archive Identification Section

The archive identification section is plain text, delimited with newline characters. It is composed of a series of keyword/value pairs, with one pair allowed per line. Keywords and values are separated by a single equal sign. There are no limits to the length of individual lines. Binary data to be included as the value to a keyword is base64 encoded. The keywords themselves are case-insensitive. The case-sensitivity of the values is determined by the definition of the keyword, though most are case-insensitive.

The global order of the keywords within the identification section is undefined, save for the section boundary keywords. The identification section must begin with `section_begin=ident` and must end with `section_end=ident`.

In addition to the keywords defined for the flash archive and enumerated below, users can define their own. These user-defined keywords are ignored by the flash mechanisms, but can be used by user-provided scripts or programs that process the identification section. User-defined keywords must begin with *X*, and contain characters other than linefeeds, equal signs, and null characters. For example, *X-department* is a valid user-defined keyword. *department*, which lacks the *X-* prefix, is not. Suggested naming conventions for user-defined keyword include the underscore-delimited descriptive method used for the pre-defined keywords, or a federated convention similar to that used to name Java packages.

Applications that process the identification section will process unrecognized non-user-defined keywords differently, depending on whether the archive version is known. If the application recognizes the archive specification version, it will reject any unrecognized non-user-defined keyword. If the application does not recognize the specification version, that is, if the minor version number is higher than the highest minor version it knows how to process, unrecognized non-user-defined keywords will be ignored. These ignored keyword are reported to the user by means of a non-fatal warning message.

The keywords defined for this version of the Flash archive specification are listed below.

Keyword	Value	Required
<code>section_begin</code>	text	yes
<code>section_end</code>	text	yes
<code>archive_id</code>	text	no
<code>files_archived_method</code>	text	no

Keyword	Value	Required
files_compressed_method	text	no
files_archived_size	numeric	no
files_unarchived_size	numeric	no
creation_date	text	no
creation_master	text	no
content_name	text	yes
content_type	text	no
content_description	text	no
content_author	text	no
content_architectures	text list	no
creation_node	text	no
creation_hardware_class	text	no
creation_platform	text	no
creation_processor	text	no
creation_release	text	no
creation_os_name	text	no
creation_os_version	text	no

Future versions of the identification section might define additional keywords. The only guarantee regarding the new keywords is that they will not intrude upon the user-defined keyword namespace as given above.

The following is an example identification section:

```
section_begin=identification
files_archived_method=cpio
files_compressed_method=compress
files_archived_size=259323342
files_unarchived_size=591238111
creation_date=20000131221409
creation_master=pumbaa
content_name=Finance Print Server
content_type=server
content_description=Solaris 8 Print Server
content_author=Mighty Matt
content_architectures=sun4u
```

```
creation_node=pumbaa
creation_hardware_class=sun4u
creation_platform=SUNW,Sun-Fire
creation_processor=sparc
creation_release=5.9
creation_os_name=SunOS
creation_os_version=s81_49
x-department=Internal Finance
section_end=identification
```

The following are descriptions of the identification section keywords:

```
section_begin
section_end
```

These keywords are used to delimit sections in the archive and are not limited exclusively to the identification section. For example, the archive files section includes a `section_begin` keyword, though with a different value. User-defined archive sections will be delimited by `section_begin` and `section_end` keywords, with values appropriate to each section. The currently defined section names are given in the table below. User-defined names should follow the same convention as user-defined identification sections, with the additional restriction that they not contain forward slashes (`/`).

Section	Boundary
identification	identification
archive files	archive
archive cookie	cookie

The archive cookie does not use section boundaries, and thus has no need for a section name within the archive itself.

The following keywords, used in the archive identification section, describe the contents of the archive files section.

`archive_id` This optional keyword *uniquely* describes the contents of the archive. It is computed as a unique hash value of the bytes representing the archive. Currently this value is represented as an ASCII hexadecimal 128-bit MD5 hash of the archive contents. This value is used by the installation software only to validate the contents of the archive during archive installation.

If the keyword is present, the hash value is recomputed during extraction based on the contents of the archive being

extracted. If the recomputed value does not match the stored value in the identification section, the archive is deemed corrupt, and appropriate actions can be taken by the application.

If the keyword is not present, no integrity check is performed.

files_archived_method

This keyword describes the archive method used in the files section. If this keyword is not present, the files section is assumed to be in `cpio(1)` format with ASCII headers (the `-c` option to `cpio`). If the keyword is present, it can have the following value:

- `pax` The archive format in the files section is `pax(1)` with extended `tar(1)` interchange format. Also allows archiving and extracting files whose size is greater than 4 GB.
- `cpio` The archive format in the files section is `cpio` with ASCII headers.

The compression method indicated by the `files_compressed_method` keyword (if present) is applied to the archive file created by the archive method.

The introduction of additional archive methods will require a change in the major archive specification version number, as applications aware only of `cpio/pax` will be unable to extract archives that use other archive methods.

files_compressed_method

This keyword describes the compression algorithm (if any) used on the files section. If this keyword is not present, the files section is assumed to be uncompressed. If the keyword is present, it can have one of the following values:

- `none` The files section is not compressed.
- `compress` The files section is compressed using `compress(1)`.

The compression method indicated by this keyword is applied to the archive file created by the archive method indicated by the value of the `files_archived_method` keyword (if any). `gzip` compression of the flash archive is not currently supported, as the `gzip` decompression program is not included in the standard miniroot.

Introduction of an additional compression algorithm would require a change in the major archive specification version number, as applications aware only of the above methods will be unable to extract archives that use other compression algorithms.

`files_archived_size`

The value associated with this keyword is the size of the archived files section, in bytes. This value is used by the deployment software only to give extraction progress information to the user. While the deployment software can easily determine the size of the archived files section prior to extraction, it cannot do so in the case of archive retrieval via a stream. To determine the compressed size when extracting from a stream, the extraction software would have to read the stream twice. This double read would result in an unacceptable performance penalty compared to the value of the information gathered.

If the keyword is present, the value is used only for the provision of status information. Because this keyword is only advisory, deployment software must be able to handle extraction of archives for which the actual file section size does not match the size given in `files_archive_size`.

If `files_archive_size` is not present and the archive is being read from a stream device that does not allow the prior determination of size information, such as a tape drive, completion status information will not be generated. If the keyword is not present and the archive is being read from a random-access device such as a mounted file system, or from a stream that provides size information, the compressed size will be generated dynamically and used for the provision of status information.

`files_unarchived_size`

This keyword defines the cumulative size in bytes of the extracted archive. The value is used for file system size verification. The following verification methods are possible using this approach:

- | | |
|--------------------|--|
| No checking | If the <code>files_unarchived_size</code> keyword is absent, no space checking will be performed. |
| Aggregate checking | If the <code>files_unarchived_size</code> keyword is present and the associated value is an integer, the |

integer will be compared against the aggregate free space created by the requested file system configuration.

The following keywords provide descriptive information about the archive as a whole. They are generally used to assist the user in archive selection and to aid in archive management. These keywords are all optional and are used by the deployment programs only to assist the user in distinguishing between individual archives.

<code>creation_date</code>	<p>The value of the <code>creation_date</code> keyword is a textual timestamp representing the time of creation for the archive. The timestamp must be in ISO-8601 complete basic calendar format without the time designator (ISO-8601, §5.4.1(a)) as follows:</p> <p>CCYYMMDDhhmss</p> <p>For example:</p> <p>20000131221409 (January 31st, 2000 10:14:09pm)</p> <p>The date and time included in the value should be in GMT.</p>
<code>creation_master</code>	<p>The value of the <code>creation_master</code> keyword is the name of the master machine used to create the archive. The value can be overridden at archive creation time.</p>
<code>content_name</code>	<p>The value of the <code>content_name</code> keyword should describe the archive's function and purpose. It is similar to the <code>NAME</code> parameter found in Solaris packages.</p> <p>The value of the <code>content_name</code> keyword is used by the deployment utilities to identify the archive and can be presented to the user during the archive selection process and/or the extraction process. The value must be no longer than 256 characters.</p>
<code>content_type</code>	<p>The value of this keyword specifies a category for the archive. This category is defined by the user and is used by deployment software for display purposes. This keyword is the flash analog of the Solaris packaging <code>CATEGORY</code> keyword.</p>
<code>content_description</code>	<p>The value of this keyword is used to provide the user with a description of what the archive contains and should build on the description provided in <code>content_name</code>. In this respect, <code>content_description</code> is analogous to the <code>DESC</code> keyword used in Solaris packages.</p>

There is no length limit to the value of `content_description`. To facilitate display, the value can contain escaped newline characters. As in C, the escaped newline takes the form of `\n`. Due to the escaped newline, backslashes must be included as `\\`. The description is displayed in a non-proportional font, with at least 40 characters available per line. Lines too long for display are wrapped.

<code>content_author</code>	The value of this keyword is a user-defined string identifying the creator of the archive. Suggested values include the full name of the creator, the creator's email address, or both.
<code>content_architectures</code>	The value of this keyword is a comma-delimited list of the kernel architectures supported by the given archive. The value of this keyword is generated at archive creation time, and can be overridden by the user at that time. If this keyword is present in the archive, the extraction mechanism validates the kernel architecture of the clone system with the list of architectures supported by the archive. The extraction fails if the kernel architecture of the clone is not supported by the archive. If the keyword is not present, no architecture validation is performed.

The keywords listed below have values filled in by `uname(2)` at the time the flash archive is created. If you create a flash archive in which the root directory is not `/`, the flash archive software inserts the string `UNKNOWN` for all of the `creation_*` keywords except `creation_node`, `creation_release`, and `creation_os_name`. For `creation_node`, the flash software uses the contents of the `nodename(4)` file. For `creation_release` and `creation_os_name`, the flash software attempts to use the contents of `<root_directory>/var/sadm/system/admin/INST_RELEASE`. If it is unsuccessful in reading this file, it assigns the value `UNKNOWN`.

Regardless of their sources, you cannot override the values of the `creation_*` keywords.

<code>creation_node</code>	The return from <code>uname -n</code> .
<code>creation_hardware_class</code>	The return from <code>uname -m</code> .
<code>creation_platform</code>	The return from <code>uname -i</code> .
<code>creation_processor</code>	The return from <code>uname -p</code> .
<code>creation_release</code>	The return from <code>uname -r</code> .
<code>creation_os_name</code>	The return from <code>uname -s</code> .
<code>creation_os_version</code>	The return from <code>uname -v</code> .

- Manifest Section** The manifest section is used only for differential flash archives. It contains a filter that specifies selection of an operating environment image and a list of the files to be retained in, added to, and deleted from a clone system. The list contains permissions, modification times, and other information on each file. The flash software uses this section to perform a consistency check prior to deployment of an archive on a clone.
- The manifest section is for the exclusive use of the flash software. The format of this section is internal to Sun and is subject to change.
- Predeployment, Postdeployment, and Reboot Sections** Contain internal information that the flash software uses before and after deploying an operating environment image. These sections are for the exclusive use of the flash software.
- Summary Section** Contains a summary of archive creation. This section records the activities of predeployment and postdeployment scripts.
- User-Defined Sections** Following the identification section can be zero or more user-defined sections. These sections are not processed by the archive extraction code and can be used for any purpose.
- User-defined sections must be line-oriented, terminated with newline (ASCII 0x0a) characters. There is no limit on the length of individual lines. If binary data is to be included in a user-defined section, it should be encoded using base64 or a similar algorithm.
- Archive Files Section** The archive files section contains the files gathered from the master system. While the length of this section should be the same as the value of the `files_archived_size` keyword in the identification section, you should not assume that these two values are equal. This section begins with `section_begin=archive`, but it does not have an ending section boundary.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWinst
Interface Stability	Committed

See Also [compress\(1\)](#), [cpio\(1\)](#), [pax\(1\)](#), [tar\(1\)](#), [md5\(3EXT\)](#), [attributes\(5\)](#)

Name format.dat – disk drive configuration for the format command

Description format . dat enables you to use your specific disk drives with `format(1M)`. On Solaris 2.3 and compatible systems, format will automatically configure and label SCSI drives, so that they need not be defined in format . dat. Three things can be defined in the data file:

- search paths
- disk types
- partition tables.

Syntax The following syntax rules apply to the data file:

- The pound # sign is the comment character. Any text on a line after a pound sign is not interpreted by format.
- Each definition in the format . dat file appears on a single logical line. If the definition is more than one line long, all but the last line of the definition must end with a backslash (\).
- A definition consists of a series of assignments that have an identifier on the left side and one or more values on the right side. The assignment operator is the equal sign (=). Assignments within a definition must be separated by a colon (:).
- White space is ignored by `format(1M)`. If you want an assigned value to contain white space, enclose the entire value in double quotes ("). This will cause the white space within quotes to be preserved as part of the assignment value.
- Some assignments can have multiple values on the right hand side. Separate values by a comma (,).

Keywords The data file contains disk definitions that are read in by `format(1M)` when it starts up. Each definition starts with one of the following keywords: `search_path`, `disk_type`, and `partition`.

`search_path` 4.x: Tells format which disks it should search for when it starts up. The list in the default data file contains all the disks in the GENERIC configuration file. If your system has disks that are not in the GENERIC configuration file, add them to the `search_path` definition in your data file. The data file can contain only one `search_path` definition. However, this single definition lets you specify all the disks you have in your system.

5.x: By default, `format(1M)` understands all the logical devices that are of the form `/dev/rdisk/cntndnsn`; hence `search_path` is not normally defined on a 5.x system.

`disk_type` Defines the controller and disk model. Each `disk_type` definition contains information concerning the physical geometry of the disk. The default data file contains definitions for the controllers and disks that the Solaris operating environment supports. You need to add a new `disk_type` only if you have an unsupported disk. You can add as many `disk_type` definitions to the data file as you want.

The following controller types are supported by `format(1M)`:

XY450	Xylogics 450 controller (SMD)
XD7053	Xylogics 7053 controller (SMD)
SCSI	True SCSI (CCS or SCSI-2)
ISP-80	IPI panther controller

The keyword itself is assigned the name of the disk type. This name appears in the disk's label and is used to identify the disk type whenever `format(1M)` is run. Enclose the name in double quotes to preserve any white space in the name.

Below are lists of identifiers for supported controllers. Note that an asterisk (*) indicates the identifier is mandatory for that controller -- it is not part of the keyword name.

The following identifiers are assigned values in all `disk_type` definitions:

<code>acyl*</code>	alternate cylinders
<code>asect</code>	alternate sectors per track
<code>atrks</code>	alternate tracks
<code>fmt_time</code>	formatting time per cylinder
<code>ncyl*</code>	number of logical cylinders
<code>nhead*</code>	number of logical heads
<code>nsect*</code>	number of logical sectors per track
<code>pcyl*</code>	number of physical cylinders
<code>phead</code>	number of physical heads
<code>psect</code>	number of physical sectors per track
<code>rpm*</code>	drive RPM

These identifiers are for SCSI and MD-21 Controllers

<code>read_retries</code>	page 1 byte 3 (read retries)
<code>write_retries</code>	page 1 byte 8 (write retries)
<code>cyl_skew</code>	page 3 bytes 18-19 (cylinder skew)
<code>trk_skew</code>	page 3 bytes 16-17 (track skew)
<code>trks_zone</code>	page 3 bytes 2-3 (tracks per zone)

cache page 38 byte 2 (cache parameter)
prefetch page 38 byte 3 (prefetch parameter)
max_prefetch page 38 byte 4 (minimum prefetch)
min_prefetch page 38 byte 6 (maximum prefetch)

Note: The Page 38 values are device-specific. Refer the user to the particular disk's manual for these values.

For SCSI disks, the following geometry specifiers may cause a mode select on the byte(s) indicated:

asect page 3 bytes 4-5 (alternate sectors per zone)
atrks page 3 bytes 8-9 (alt. tracks per logical unit)
phead page 4 byte 5 (number of heads)
psect page 3 bytes 10-11 (sectors per track)

And these identifiers are for SMD Controllers Only

bps* bytes per sector (SMD)
bpt* bytes per track (SMD)

Note: under SunOS 5.x, bpt is only required for SMD disks. Under SunOS 4.x, bpt was required for all disk types, even though it was only used for SMD disks.

And this identifier is for XY450 SMD Controllers Only

drive_type* drive type (SMD) (just call this "xy450 drive type")

partition Defines a partition table for a specific disk type. The partition table contains the partitioning information, plus a name that lets you refer to it in [format\(1M\)](#). The default data file contains default partition definitions for several kinds of disk drives. Add a partition definition if you repartitioned any of the disks on your system. Add as many partition definitions to the data file as you need.

Partition naming conventions differ in SunOS 4.x and in SunOS 5.x.

4.x: the partitions are named as a, b, c, d, e, f, g, h.

5.x: the partitions are referred to by numbers 0, 1, 2, 3, 4, 5, 6, 7.

Examples EXAMPLE 1 A sample disk_type and partition.

Following is a sample disk_type and partition definition in format.dat file for SUN0535 disk device.

```
disk_type = "SUN0535" \  
  : ctlr = SCSI : fmt_time = 4 \  
  : ncyl = 1866 : acyl = 2 : pcyl = 2500 : nhead = 7 : nsect = 80 \  
  : rpm = 5400  
partition = "SUN0535" \  
  : disk = "SUN0535" : ctlr = SCSI \  
  : 0 = 0, 64400 : 1 = 115, 103600 : 2 = 0, 1044960 : 6 = 300, 876960
```

Files /etc/format.dat default data file if format -x is not specified, nor is there a format.dat file in the current directory.

See Also [format\(1M\)](#)

Oracle Solaris Administration: Common Tasks

Name fspec – format specification in text files

Description It is sometimes convenient to maintain text files on the system with non-standard tabs, (tabs that are not set at every eighth column). Such files must generally be converted to a standard format, frequently by replacing all tabs with the appropriate number of spaces, before they can be processed by system commands. A format specification occurring in the first line of a text file specifies how tabs are to be expanded in the remainder of the file.

A format specification consists of a sequence of parameters separated by blanks and surrounded by the brackets <: and :>. Each parameter consists of a keyletter, possibly followed immediately by a value. The following parameters are recognized:

ttabs The *t* parameter specifies the tab settings for the file. The value of *tabs* must be one of the following:

- A list of column numbers separated by commas, indicating tabs set at the specified columns.
- A '-' followed immediately by an integer *n*, indicating tabs at intervals of *n* columns.
- A '-' followed by the name of a "canned" tab specification.

Standard tabs are specified by *t-8*, or equivalently, *t1,9,17,25*, etc. The canned tabs that are recognized are defined by the [tabs\(1\)](#) command.

ssize The *s* parameter specifies a maximum line size. The value of *size* must be an integer. Size checking is performed after tabs have been expanded, but before the margin is prepended.

mmargin The *m* parameter specifies a number of spaces to be prepended to each line. The value of *margin* must be an integer.

d The *d* parameter takes no value. Its presence indicates that the line containing the format specification is to be deleted from the converted file.

e The *e* parameter takes no value. Its presence indicates that the current format is to prevail only until another format specification is encountered in the file.

Default values, which are assumed for parameters not supplied, are *t-8* and *m0*. If the *s* parameter is not specified, no size checking is performed. If the first line of a file does not contain a format specification, the above defaults are assumed for the entire file. The following is an example of a line containing a format specification:

```
* <:t5,10,15 s72:> *
```

If a format specification can be disguised as a comment, it is not necessary to code the *d* parameter.

See Also [ed\(1\)](#), [newform\(1\)](#), [tabs\(1\)](#)

Name `fstypes` – file that registers distributed file system packages

Description `fstypes` resides in directory `/etc/dfs` and lists distributed file system utilities packages installed on the system. For each installed distributed file system type, there is a line that begins with the file system type name (for example, “`nfs`”), followed by white space and descriptive text.

The file system indicated in the first line of the file is the default file system; when Distributed File System (DFS) Administration commands are entered without the option `-F fstypes`, the system takes the file system type from the first line of the `fstypes` file.

The default file system can be changed by editing the `fstypes` file with any supported text editor.

See Also [dfmounts\(1M\)](#), [dfshares\(1M\)](#), [share\(1M\)](#), [shareall\(1M\)](#), [unshare\(1M\)](#)

Name ftp – FTP client configuration file

Synopsis /etc/default/ftp

Description Use the ftp file to configure the behavior of the FTP client. Lines that begin with a hash symbol (“#”) are treated as comment lines and are ignored.

Behavior Directives The ftp file supports the following behavior directives:

FTP_LS_SENDS_NLST=yes | no The ls command of the ftp client sends an NLST to the FTP Server by default. Several non-Solaris clients send LIST instead. In order to make the Solaris ftp client send LIST when the ls command is issued, set FTP_LS_SENDS_NLST to no. The value of FTP_LS_SENDS_NLST is yes by default.

If the user sets a value for FTP_LS_SENDS_NLST in the user's environment, this value will override any FTP_LS_SENDS_NLST directive that is specified in /etc/default/ftp.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/ftp

See Also [ftp\(1\)](#), [attributes\(5\)](#)

Name ftputers – file listing users to be disallowed ftp login privileges

Synopsis /etc/ftpd/ftputers

Description The ftputers file lists users for whom ftp login privileges are disallowed. Each ftpuser entry is a single line of the form:

name

where name is the user's login name.

The FTP Server, in `ftpd -lm`, reads the ftputers file. If the login name of the user matches one of the entries listed, it rejects the login attempt.

The ftputers file has the following default configuration entries:

```
root
daemon
bin
sys
adm
lp
uccp
nuucp
dladm
smmsp
listen
gdm
upnp
xvm
mysql
openldap
webservd
svctag
unknown
nobody
noaccess
nobody4
```

These entries match the default instantiated entries from [passwd\(4\)](#). The list of default entries typically contains the superuser root and other administrative and system application identities.

The root entry is included in the ftputers file as a security measure since the default policy is to disallow remote logins for this identity. This policy is also set in the default value of the `CONSOLE` entry in the `/etc/default/login` file. See [login\(1\)](#). If you allow root login privileges by deleting the root entry in ftputers, you should also modify the security policy in `/etc/default/login` to reflect the site security policy for remote login access by root.

Other default entries are administrative identities that are typically assumed by system applications but never used for local or remote login, for example `sys` and `nobody`. Since these entries do not have a valid password field instantiated in [shadow\(4\)](#), no login can be performed.

If a site adds similar administrative or system application identities in [passwd\(4\)](#) and [shadow\(4\)](#), for example, `majordomo`, the site should consider including them in the `ftputers` file for a consistent security policy.

Lines that begin with `#` are treated as comment lines and are ignored.

Files	<code>/etc/ftpd/ftputers</code>	A file that lists users for whom ftp login privileges are disallowed by default.
	<code>/etc/ftputers</code>	This is symbolic link to <code>/etc/ftpd/ftputers</code> .
	<code>/etc/default/login</code>	Establishes login environment.
	<code>/etc/passwd</code>	Password file.
	<code>/etc/shadow</code>	Shadow password file.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTETYPE	ATTRIBUTEVALUE
Availability	service/network/ftp
Interface Stability	See below.

The Interface Stability for `/etc/ftpd/ftputers` is Volatile.

See Also [login\(1\)](#), [passwd\(4\)](#), [shadow\(4\)](#), [attributes\(5\)](#), [environ\(5\)](#)

Name fx_dptbl – fixed priority dispatcher parameter table

Synopsis fx_dptbl

Description The process scheduler or dispatcher is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the notion of scheduling classes, where each class defines a scheduling policy used to schedule processes within that class. Associated with each scheduling class is a set of priority queues on which ready-to-run processes are linked. These priority queues are mapped by the system configuration into a set of global scheduling priorities, which are available to processes within the class. The dispatcher always selects for execution the process with the highest global scheduling priority in the system. The priority queues associated with a given class are viewed by that class as a contiguous set of priority levels numbered from 0 (lowest priority) to n (highest priority—a configuration-dependent value). The set of global scheduling priorities that the queues for a given class are mapped into might not start at zero and might not be contiguous, depending on the configuration.

Processes in the fixed priority class are scheduled according to the parameters in a fixed-priority dispatcher parameter table (fx_dptbl). The fx_dptbl table consists of an array (config_fx_dptbl[]) of parameter structures (struct fxdpent_t), one for each of the n priority levels used by fixed priority processes in user mode. The structures are accessed by way of a pointer, (fx_dptbl), to the array. The properties of a given priority level i are specified by the i th parameter structure in this array (fx_dptbl[i]).

A parameter structure consists of the following members. These are also described in the /usr/include/sys/fx.h header.

- fx_globpri The global scheduling priority associated with this priority level. The mapping between fixed-priority priority levels and global scheduling priorities is determined at boot time by the system configuration. fx_globpri can not be changed with [dispadmin\(1M\)](#).
- fx_quantum The length of the time quantum allocated to processes at this level in ticks (hz). The time quantum value is only a default or starting value for processes at a particular level, as the time quantum of a fixed priority process can be changed by the user with the [prioctl\(1\)](#) command or the [prioctl\(2\)](#) system call.

In the high resolution clock mode (hires_tick set to 1), the value of hz is set to 1000. Increase quantum to maintain the same absolute time quantum.

An administrator can affect the behavior of the fixed priority portion of the scheduler by reconfiguring the fx_dptbl. There are two methods available for doing this: reconfigure with a loadable module at boot-time or by using [dispadmin\(1M\)](#) at run-time.

fx_dptbl Loadable Module The `fx_dptbl` can be reconfigured with a loadable module that contains a new fixed priority dispatch table. The module containing the dispatch table is separate from the FX loadable module, which contains the rest of the fixed priority software. This is the only method that can be used to change the number of fixed priority priority levels or the set of global scheduling priorities used by the fixed priority class. The relevant procedure and source code is described in [Replacing the fx_dptbl Loadable Module](#) below.

dispadmin Configuration File The `fx_quantum` values in the `fx_dptbl` can be examined and modified on a running system using the `dispadmin(IM)` command. Invoking `dispadmin` for the fixed-priority class allows the administrator to retrieve the current `fx_dptbl` configuration from the kernel's in-core table or overwrite the in-core table with values from a configuration file. The configuration file used for input to `dispadmin` must conform to the specific format described as follows:

- Blank lines are ignored and any part of a line to the right of a # symbol is treated as a comment.
- The first non-blank, non-comment line must indicate the resolution to be used for interpreting the time quantum values. The resolution is specified as:

```
RES=res
```

where *res* is a positive integer between 1 and 1,000,000,000 inclusive and the resolution used is the reciprocal of *res* in seconds (for example, `RES=1000` specifies millisecond resolution). Although you can specify very fine (nanosecond) resolution, the time quantum lengths are rounded up to the next integral multiple of the system clock's resolution.

- The remaining lines in the file are used to specify the `fx_quantum` values for each of the fixed-priority priority levels. The first line specifies the quantum for fixed-priority level 0, the second line specifies the quantum for fixed-priority level 1, and so forth. There must be exactly one line for each configured fixed priority priority level. Each `fx_quantum` entry must be a positive integer specifying the desired time quantum in the resolution given by *res*.

See [Examples](#) for an example of an excerpt of a `dispadmin` configuration file.

Replacing the fx_dptbl Loadable Module To change the size of the fixed priority dispatch table, you must build the loadable module that contains the dispatch table information. Save the existing module before using the following procedure.

1. Place the dispatch table code shown below in a file called `fx_dptbl.c`. See [EXAMPLES](#), below, for an example of this file.
2. Compile the code using the given compilation and link lines supplied:

```
cc -c -O -D_KERNEL fx_dptbl.c
ld -r -o FX_DPTBL fx_dptbl.o
```

3. Copy the current dispatch table in `/usr/kernel/sched` to `FX_DPTBL.bak`.

4. Replace the current `FX_DPTBL` in `/usr/kernel/sched`.
5. Make changes in the `/etc/system` file to reflect the changes to the sizes of the tables. See [system\(4\)](#). The variables affected is `fx_maxupri`. The syntax for setting this is as follows:


```
set  FX:fx_maxupri=(value for max fixed-priority user priority)
```
6. Reboot the system to use the new dispatch table.

Exercise great care in using the preceding method to replace the dispatch table. A mistake can result in panics, thus making the system unusable.

Examples **EXAMPLE 1** Configuration File Excerpt

The following excerpt from a `dispadmin` configuration file illustrates the correct format. Note that, for each line specifying a set of parameters, there is a comment indicating the corresponding priority level. These level numbers indicate priority within the fixed priority class; the mapping between these fixed-priority priorities and the corresponding global scheduling priorities is determined by the configuration specified in the `FX_DPTBL` loadable module. The level numbers are strictly for the convenience of the administrator reading the file and, as with any comment, they are ignored by `dispadmin`. The `dispadmin` command assumes that the lines in the file are ordered by consecutive, increasing priority level (from 0 to the maximum configured fixed-priority priority). For the sake of someone reading the file, the level numbers in the comments should agree with this ordering. If for some reason they do not, `dispadmin` is unaffected.

```
# Fixed Priority Dispatcher Configuration File RES=1000
```

```
RES=1000
#  TIME QUANTUM          PRIORITY
# (fx_quantum)          LEVEL
200                      #  0
200                      #  1
200                      #  2
200                      #  3
200                      #  4
200                      #  5
200                      #  6
200                      #  7
.                          .  .
.                          .  .
.                          .  .
20                        #  58
20                        #  59
20                        #  60
```

EXAMPLE 2 `fx_dptbl.c` File Used for Building the New `fx_dptbl`

The following is an example of a `fx_dptbl.c` file used for building the new `fx_dptbl`.

EXAMPLE 2 fx_dptbl.c File Used for Building the New fx_dptbl (Continued)

```

/* BEGIN fx_dptbl.c */

#include <sys/proc.h>
#include <sys/priocntl.h>
#include <sys/class.h>
#include <sys/disp.h>
#include <sys/fx.h>
#include <sys/fxpriocntl.h>

/*
 * This is the loadable module wrapper.
 */

#include <sys/modctl.h>

extern struct mod_ops mod_miscops;

/*
 * Module linkage information for the kernel.
 */

static struct modlmisc modlmisc = {
    &mod_miscops, "Fixed priority dispatch table"
};

static struct modlinkage modlinkage = {
    MODREV_1, &modlmisc, 0
};

_init()
{
    return (mod_install(&modlinkage));
}

_info(modinfop)
    struct modinfo *modinfop;
{
    return (mod_info(&modlinkage, modinfop));
}

#define FXGPUP0 0 /* Global priority for FX user priority 0 */
fxdpent_t config_fx_dptbl[] = {

/* glbpri      qntm */

```

EXAMPLE 2 fx_dptbl.c File Used for Building the New fx_dptbl (Continued)

```
FXGPUP0+0, 20,  
FXGPUP0+1, 20,  
FXGPUP0+2, 20,  
FXGPUP0+3, 20,  
FXGPUP0+4, 20,  
FXGPUP0+5, 20,  
FXGPUP0+6, 20,  
FXGPUP0+7, 20,  
FXGPUP0+8, 20,  
FXGPUP0+9, 20,  
FXGPUP0+10, 16,  
FXGPUP0+11, 16,  
FXGPUP0+12, 16,  
FXGPUP0+13, 16,  
FXGPUP0+14, 16,  
FXGPUP0+15, 16,  
FXGPUP0+16, 16,  
FXGPUP0+17, 16,  
FXGPUP0+18, 16,  
FXGPUP0+19, 16,  
FXGPUP0+20, 12,  
FXGPUP0+21, 12,  
FXGPUP0+22, 12,  
FXGPUP0+23, 12,  
FXGPUP0+24, 12,  
FXGPUP0+25, 12,  
FXGPUP0+26, 12,  
FXGPUP0+27, 12,  
FXGPUP0+28, 12,  
FXGPUP0+29, 12,  
FXGPUP0+30, 8,  
FXGPUP0+31, 8,  
FXGPUP0+32, 8,  
FXGPUP0+33, 8,  
FXGPUP0+34, 8,  
FXGPUP0+35, 8,  
FXGPUP0+36, 8,  
FXGPUP0+37, 8,  
FXGPUP0+38, 8,  
FXGPUP0+39, 8,  
FXGPUP0+40, 4,  
FXGPUP0+41, 4,  
FXGPUP0+42, 4,  
FXGPUP0+43, 4,
```


EXAMPLE 2 fx_dptbl.c File Used for Building the New fx_dptbl (Continued)

```

    FXGPUP0+44,  4,
    FXGPUP0+45,  4,
    FXGPUP0+46,  4,
    FXGPUP0+47,  4,
    FXGPUP0+48,  4,
    FXGPUP0+49,  4,
    FXGPUP0+50,  4,
    FXGPUP0+51,  4,
    FXGPUP0+52,  4,
    FXGPUP0+53,  4,
    FXGPUP0+54,  4,
    FXGPUP0+55,  4,
    FXGPUP0+56,  4,
    FXGPUP0+57,  4,
    FXGPUP0+58,  4,
    FXGPUP0+59,  2,
    FXGPUP0+60,  2,
};

pri_t config_fx_maxumdpr =
    sizeof (config_fx_dptbl) / sizeof (fxdpent_t) - 1;

/*
 * Return the address of config_fx_dptbl
 */
fxdpent_t *
fx_getdptbl()
{
    return (config_fx_dptbl);
}

/*
 * Return the address of fx_maxumdpr
 */
pri_t
fx_getmaxumdpr()
{
    /*
     * the config_fx_dptbl table.
     */
    return (config_fx_maxumdpr);
}

```

See Also [priocntl\(1\)](#), [dispadmin\(1M\)](#), [priocntl\(2\)](#), [system\(4\)](#)

System Administration Guide, Volume 1, System Interface Guide

Notes In order to improve performance under heavy system load, both the `nfsd` daemon and the `lockd` daemon utilize the maximum priority in the FX class. Unusual `fx_dptbl` configurations may have significant negative impact on the performance of the `nfsd` and `lockd` daemons.

Name gateways – configuration file for /usr/sbin/in.routed IPv4 network routing daemon

Synopsis /etc/gateways

Description The /etc/gateways file is used by the routing daemon, [in.routed\(1M\)](#). When the daemon starts, it reads /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.

The /etc/gateways file consists of a series of lines, each in one of the two formats shown below or consisting of parameters described later. Blank lines and lines starting with “#” are treated as comments.

One format specifies networks:

```
net Nname[/mask] gateway Gname metric value <passive | active | external>
```

The other format specifies hosts:

```
host Hname gateway Gname metric value <passive | active | external>
```

Host *hname* is equivalent to net *nname/32*.

The parameters in the lines shown above are described as follows:

<i>Nname</i> or <i>Hname</i>	Name of the destination network or host. It can be a symbolic network name or an Internet address specified in <i>dot</i> notation (see inet(3SOCKET)). If it is a name, then it must either be defined in /etc/networks or /etc/hosts, or a naming service must have been started before in.routed(1M) .
<i>Mask</i>	An optional number between 1 and 32 indicating the netmask associated with <i>Nname</i> .
<i>Gname</i>	Name or address of the gateway to which RIP responses should be forwarded.
<i>Value</i>	The hop count to the destination host or network.
passive active external	One of these keywords must be present to indicate whether the gateway should be treated as passive or active, or whether the gateway is external to the scope of the RIP protocol. A passive gateway is not expected to exchange routing information, while gateways marked active should be willing to exchange RIP packets. See in.routed(1M) for further details.

After turning on debugging in `in.routed` with the `-t` option, you can see that lines that follow the format described above create pseudo-interfaces. To set parameters for remote or external interfaces, use a line starting with `if=alias(Hname)`, `if=remote(Hname)`, and so forth.

For backward compatibility with the previous Solaris `in.routed` implementation, three special keyword formats are accepted. If present, these forms must each be on a separate line, and must not be combined on the same line with any of the keywords listed elsewhere in this document. These three forms are:

<code>norip ifname</code>	Disable all RIP processing on the specified interface.
<code>noripin ifname</code>	Disable the processing of received RIP responses on the specified interface.
<code>noripout ifname</code>	Disable RIP output on the specified interface.

Lines that start with neither `net` nor `host` must consist of one or more of the following parameter settings, separated by commas or blanks:

`if=ifname`

Indicates that the other parameters on the line apply only to the interface name *ifname*. If this parameter is not specified, then other parameters on the line apply to all interfaces.

`subnet=nname[/mask][,metric]`

Advertises a route to network *nname* with mask *mask* and the supplied metric (default 1). This is useful for filling *holes* in CIDR allocations. This parameter must appear by itself on a line. The network number must specify a full, 32-bit value, as in `192.0.2.0` instead of `192.0.2`.

`ripv1_mask=nname/mask1,mask2`

Specifies that the netmask of the network of which *nname/mask1* is a subnet should be *mask2*. For example, `ripv1_mask=192.0.2.16/28,27` marks `192.0.2.16/28` as a subnet of `192.0.2.0/27` instead of `192.0.2.0/24`. It is better to turn on RIPv2 instead of using this facility. See the description of `ripv2_out`, below.

`passwd=XXX[|KeyID[start|stop]]`

Specifies a RIPv2 cleartext password that will be included on all RIPv2 responses sent, and checked on all RIPv2 responses received. Any blanks, tab characters, commas, or “#”, “|”, or NULL characters in the password must be escaped with a backslash (\). The common escape sequences `\n`, `\r`, `\t`, `\b`, and `\xxx` have their usual meanings. The *KeyID* must be unique but is ignored for cleartext passwords. If present, *start* and *stop* are timestamps in the form `year/month/day@hour:minute`. They specify when the password is valid. The valid password with the longest future is used on output packets, unless all passwords have expired, in which case the password that expired most recently is used. If no passwords are valid yet, no password is output. Incoming packets can carry any password that is valid, will be valid within 24 hours, or that was valid within 24 hours. To protect password secrecy, the `passwd` settings are valid only in the `/etc/gateways` file and only when that file is readable only by UID 0.

`md5_passwd=XXX|KeyID[start|stop]`
Specifies a RIPv2 MD5 password. Except that a KeyID is required, this keyword is similar to `passwd` (described above).

`no_ag`
Turns off aggregation of subnets in RIPv1 and RIPv2 responses.

`no_host`
Turns off acceptance of host routes.

`no_super_ag`
Turns off aggregation of networks into supernets in RIPv2 responses.

`passive`
Marks the interface not to be advertised in updates sent over other interfaces, and turns off all RIP and router discovery through the interface.

`no_rip`
Disables all RIP processing on the specified interface. If no interfaces are allowed to process RIP packets, `in.routed` acts purely as a router discovery daemon.

Note that turning off RIP without explicitly turning on router discovery advertisements with `rdisc_adv` or `-s` causes `in.routed` to act as a client router discovery daemon, which does not advertise.

`no_rip_mcast`
Causes RIPv2 packets to be broadcast instead of multicast.

`no_ripv1_in`
Causes RIPv1 received responses to be ignored.

`no_ripv2_in`
Causes RIPv2 received responses to be ignored.

`ripv2_out`
Turns on RIPv2 output and causes RIPv2 advertisements to be multicast when possible.

`ripv2`
Equivalent to `no_ripv1_in` and `ripv2_out`. This enables RIPv2 and disables RIPv1.

`no_rdisc`
Disables the Internet Router Discovery Protocol.

`no_solicit`
Disables the transmission of Router Discovery Solicitations.

`send_solicit`
Specifies that Router Discovery solicitations should be sent, even on point-to-point links, which, by default, only listen to Router Discovery messages.

`no_rdisc_adv`
Disables the transmission of Router Discovery Advertisements.

rdisc_adv

Specifies that Router Discovery Advertisements should be sent, even on point-to-point links, which by default only listen to Router Discovery messages.

bcast_rdisc

Specifies that Router Discovery packets should be broadcast instead of multicast.

rdisc_pref=*N*

Sets the preference in Router Discovery Advertisements to the optionally signed integer *N*. The default preference is 0. Default routes with higher or less negative preferences are preferred by clients.

rdisc_interval=*N*

Sets the nominal interval with which Router Discovery Advertisements are transmitted to *N* seconds and their lifetime to $3*N$.

fake_default=*metric*

Has an identical effect to `-F net[/mask][=metric]` with the network number and netmask coming from the specified interface.

pm_rdisc

Similar to `fake_default`. To prevent RIPv1 listeners from receiving RIPv2 routes when those routes are multicast, this feature causes a RIPv1 default route to be broadcast to RIPv1 listeners. Unless modified with `fake_default`, the default route is broadcast with a metric of 14. That serves as a *poor man's router discovery* protocol.

trust_gateway=*rtr_name* [| *net1/mask1* | *net2/mask2*] . . .]

Causes RIP packets from that router and other routers named in other `trust_gateway` keywords to be accepted, and packets from other routers to be ignored. If networks are specified, then routes to other networks will be ignored from that router.

redirect_ok

Causes RIP to allow ICMP Redirect messages when the system is acting as a router and forwarding packets. Otherwise, ICMP Redirect messages are overridden.

rip_neighbor=*x.x.x.x*

By default, RIPv1 advertisements over point-to-point links are sent to the peer's address (255.255.255.255, if none is available), and RIPv2 advertisements are sent to either the RIP multicast address or the peer's address if `no_rip_mcast` is set. This option overrides those defaults and configures a specific address to use on the indicated interface. This can be used to set a broadcast type advertisement on a point-to-point link.

See Also [in.routed\(1M\)](#), [route\(1M\)](#), [rtquery\(1M\)](#), [inet\(3SOCKET\)](#),

Internet Transport Protocols, X SIS 028112, Xerox System Integration Standard

Name geniconvtbl – geniconvtbl input file format

Description An input file to `geniconvtbl` is an ASCII text file that contains an `iconv` code conversion definition from one codeset to another codeset.

The `geniconvtbl` utility accepts the code conversion definition file(s) and writes code conversion binary table file(s) that can be used in `iconv(1)` and `iconv(3C)` to support user-defined code conversions. See `iconv(1)` and `iconv(3C)` for more detail on the `iconv` code conversion and `geniconvtbl(1)` for more detail on the utility.

The Lexical Conventions The following lexical conventions are used in the `iconv` code conversion definition:

CONVERSION_NAME	A string of characters representing the name of the <code>iconv</code> code conversion. The <code>iconv</code> code conversion name should start with one or more printable ASCII characters followed by a percentage character '%' followed by another one or more of printable ASCII characters. Examples: <code>ISO8859-1%ASCII</code> , <code>646%eucJP</code> , <code>CP_939%ASCII</code> .
NAME	A string of characters starts with any one of the ASCII alphabet characters or the underscore character, '_', followed by one or more ASCII alphanumeric characters and underscore character, '_'. Examples: <code>_a1</code> , <code>ABC_codeset</code> , <code>K1</code> .
HEXADECIMAL	A hexadecimal number. The hexadecimal representation consists of an escape character, '0' followed by the constant 'x' or 'X' and one or more hexadecimal digits. Examples: <code>0x0</code> , <code>0x1</code> , <code>0x1a</code> , <code>0X1A</code> , <code>0x1B3</code> .
DECIMAL	A decimal number, represented by one or more decimal digits. Examples: <code>0</code> , <code>123</code> , <code>2165</code> .

Each comment starts with `//'` ends at the end of the line.

The following keywords are reserved:

automatic	between	binary
break	condition	default
dense	direction	discard
else	error	escapeseq
false	if	index
init	input	inputsize
map	maptype	no_change_copy

operation	output	output_byte_length
outputsize	printchr	printhd
printint	reset	return
true		

Additionally, the following symbols are also reserved as tokens:

{ 14; 14; } 14; 14; [14; 14;] 14; 14; (14; 14;) 14; 14; ; 14; 14; , 14;

The precedence and associativity

The following table shows the precedence and associativity of the operators from lower precedence at the top to higher precedence at the bottom of the table allowed in the iconv code conversion definition:

Operator (Symbol)	Associativity

Assignment (=)	Right

Logical OR ()	Left

Logical AND (&&)	Left

Bitwise OR ()	Left

Exclusive OR (^)	Left

Bitwise AND (&)	Left

Equal-to (= =), Inequality (!=)	Left

Less-than (<), Less-than-or-equal-to (<=), Greater-than (>), Greater-than-or-equal-to (>=)	Left

Left-shift (<<), Right-shift (>>)	Left

Addition (+), Subtraction (-)	Left

Multiplication (*), Division (/), Remainder (%)	Left

Logical negation (!),	Right

Bitwise complement (~),
 Unary minus (-)

The Syntax Each iconv code conversion definition starts with `CONVERSION_NAME` followed by one or more semicolon separated code conversion definition elements:

```
// a US-ASCII to ISO8859-1 iconv code conversion example:
US-ASCII%ISO8859-1 {

    // one or more code conversion definition elements here.

    :
    :

}
```

Each code conversion definition element can be any one of the following elements:

direction
 condition
 operation
 map

To have a meaningful code conversion, there should be at least one direction, operation, or map element in the iconv code conversion definition.

The direction element contains one or more semicolon separated condition-action pairs that direct the code conversion:

```
direction For_US-ASCII_2_ISO8859-1 {

    // one or more condition-action pairs here.
    :
    :

}
```

Each condition-action pair contains a conditional code conversion that consists of a condition element and an action element.

condition action

If the pre-defined condition is met, the corresponding action is executed. If there is no pre-defined condition met, `iconv(3C)` will return -1 with `errno` set to `EILSEQ`. The condition can be a condition element, a name to a pre-defined condition element, or a condition literal value, `true`. The `'true'` condition literal value always yields success and thus the corresponding action is always executed. The action also can be an action element or a name to a pre-defined action element.

The condition element specifies one or more condition expression elements. Since each condition element can have a name and also can exist stand-alone, a pre-defined condition element can be referenced by the name at any action pairs later. To be used in that way, the corresponding condition element should be defined beforehand:

```
condition For_US-ASCII_2_ISO8859-1 {  
  
    // one or more condition expression elements here.  
    :  
    :  
  
}
```

The name of the condition element in the above example is For_US-ASCII_2_ISO8859-1. Each condition element can have one or more condition expression elements. If there are more than one condition expression elements, the condition expression elements are checked from top to bottom to see if any one of the condition expression elements will yield a true. Any one of the following can be a condition expression element:

between
escapeseq
expression

The between condition expression element defines one or more comma-separated ranges:

```
between 0x0...0x1f, 0x7f...0x9f ;  
between 0xa1a1...0xfefe ;
```

In the first expression in the example above, the covered ranges are 0x0 to 0x1f and 0x7f to 0x9f inclusively. In the second expression, the covered range is the range whose first byte is 0xa1 to 0xfe and whose second byte is between 0xa1 to 0xfe. This means that the range is defined by each byte. In this case, the sequence 0xa280 does not meet the range.

The escapeseq condition expression element defines an equal-to condition for one or more comma-separated escape sequence designators:

```
// ESC $ ) C sequence:  
escapeseq 0x1b242943;  
  
// ESC $ ) C sequence or ShiftOut (SO) control character code, 0x0e:  
escapeseq 0x1b242943, 0x0e;
```

The expression can be any one of the following and can be surrounded by a pair of parentheses, '(' and ')':

```
// HEXADECIMAL:  
0xa1a1
```

```
// DECIMAL
12

// A boolean value, true:
true

// A boolean value, false:
false

// Addition expression:
1 + 2

// Subtraction expression:
10 - 3

// Multiplication expression:
0x20 * 10

// Division expression:
20 / 10

// Remainder expression:
17 % 3

// Left-shift expression:
1 << 4

// Right-shift expression:
0xa1 >> 2

// Bitwise OR expression:
0x2121 | 0x8080

// Exclusive OR expression:
0xa1a1 ^ 0x8080

// Bitwise AND expression:
0xa1 & 0x80

// Equal-to expression:
0x10 == 16

// Inequality expression:
0x10 != 10

// Less-than expression:
0x20 < 25
```

```
// Less-than-or-equal-to expression:
10 <= 0x10

// Bigger-than expression:
0x10 > 12

// Bigger-than-or-equal-to expression:
0x10 >= 0xa

// Logical OR expression:
0x10 || false

// Logical AND expression:
0x10 && false

// Logical negation expression:
! false

// Bitwise complement expression:
~0

// Unary minus expression:
-123
```

There is a single type available in this expression: integer. The boolean values are two special cases of integer values. The 'true' boolean value's integer value is 1 and the 'false' boolean value's integer value is 0. Also, any integer value other than 0 is a true boolean value. Consequently, the integer value 0 is the false boolean value. Any boolean expression yields integer value 1 for true and integer value 0 for false as the result.

Any literal value shown at the above expression examples as operands, that is, DECIMAL, HEXADECIMAL, and boolean values, can be replaced with another expression. There are a few other special operands that you can use as well in the expressions: 'input', 'inputsize', 'outputsize', and variables. `input` is a keyword pointing to the current input buffer. `inputsize` is a keyword pointing to the current input buffer size in bytes. `outputsize` is a keyword pointing to the current output buffer size in bytes. The NAME lexical convention is used to name a variable. The initial value of a variable is 0. The following expressions are allowed with the special operands:

```
// Pointer to the third byte value of the current input buffer:
input[2]

// Equal-to expression with the 'input':
input == 0x8020

// Alternative way to write the above expression:
0x8020 == input
```

```

// The size of the current input buffer size:
inputsize

// The size of the current output buffer size:
outputsize

// A variable:
saved_second_byte

// Assignment expression with the variable:
saved_second_byte = input[1]

```

The input keyword without index value can be used only with the equal-to operator, '='. When used in that way, the current input buffer is consecutively compared with another operand byte by byte. An expression can be another operand. If the input keyword is used with an index value n , it is a pointer to the $(n+1)$ th byte from the beginning of the current input buffer. An expression can be the index. Only a variable can be placed on the left hand side of an assignment expression.

The action element specifies an action for a condition and can be any one of the following elements:

```

direction
operation
map

```

The operation element specifies one or more operation expression elements:

```

operation For_US-ASCII_2_ISO8859-1 {

    // one or more operation expression element definitions here.
    :
    :

}

```

If the name of the operation element, in the case of the above example, For_US-ASCII_2_ISO8859-1, is either `init` or `reset`, it defines the initial operation and the reset operation of the iconv code conversion:

```

// The initial operation element:
operation init {

    // one or more operation expression element definitions here.
    :
    :

}

```

```
// The reset operation element:
operation reset {

    // one or more operation expression element definitions here.
    :
    :
}

```

The initial operation element defines the operations that need to be performed in the beginning of the iconv code conversion. The reset operation element defines the operations that need to be performed when a user of the iconv(3) function requests a state reset of the iconv code conversion. For more detail on the state reset, refer to [iconv\(3C\)](#).

The operation expression can be any one of the following three different expressions and each operation expression should be separated by an ending semicolon:

```
if-else operation expression
output operation expression
control operation expression

```

The if-else operation expression makes a selection depend on the boolean expression result. If the boolean expression result is true, the true task that follows the 'if' is executed. If the boolean expression yields false and if a false task is supplied, the false task that follows the 'else' is executed. There are three different kinds of if-else operation expressions:

```
// The if-else operation expression with only true task:
if (expression) {

    // one or more operation expression element definitions here.
    :
    :
}

// The if-else operation expression with both true and false
// tasks:
if (expression) {

    // one or more operation expression element definitions here.
    :
    :

} else {

    // one or more operation expression element definitions here.
    :
    :
}

```

```

}

// The if-else operation expression with true task and
// another if-else operation expression as the false task:
if (expression) {

    // one or more operation expression element definitions here.
    :
    :

} else if (expression) {

    // one or more operation expression element definitions here.
    :
    :

} else {

    // one or more operation expression element definitions here.
    :
    :

}

```

The last if-else operation expression can have another if-else operation expression as the false task. The other if-else operation expression can be any one of above three if-else operation expressions.

The output operation expression saves the right hand side expression result to the output buffer:

```

// Save 0x8080 at the output buffer:
output = 0x8080;

```

If the size of the output buffer left is smaller than the necessary output buffer size resulting from the right hand side expression, the iconv code conversion will stop with E2BIG errno and (size_t) - 1 return value to indicate that the code conversion needs more output buffer to complete. Any expression can be used for the right hand side expression. The output buffer pointer will automatically move forward appropriately once the operation is executed.

The control operation expression can be any one of the following expressions:

```

// Return (size_t)-1 as the return value with an EINVAL errno:
error;

// Return (size_t)-1 as the return value with an EBADF errno:
error 9;

```

```
// Discard input buffer byte operation. This discards a byte from
// the current input buffer and move the input buffer pointer to
// the 2'nd byte of the input buffer:
discard;

// Discard input buffer byte operation. This discards
// 10 bytes from the current input buffer and move the input
// buffer pointer to the 11'th byte of the input buffer:
discard 10;

// Return operation. This stops the execution of the current
// operation:
return;

// Operation execution operation. This executes the init
// operation defined and sets all variables to zero:
operation init;

// Operation execution operation. This executes the reset
// operation defined and sets all variables to zero:
operation reset;

// Operation execution operation. This executes an operation
// defined and named 'ISO8859_1_to_ISO8859_2':
operation ISO8859_1_to_ISO8859_2;

// Direction operation. This executes a direction defined and
// named 'ISO8859_1_to_KOI8_R':
direction ISO8859_1_to_KOI8_R;

// Map execution operation. This executes a mapping defined
// and named 'Map_ISO8859_1_to_US_ASCII':
map Map_ISO8859_1_to_US_ASCII;

// Map execution operation. This executes a mapping defined
// and named 'Map_ISO8859_1_to_US_ASCII' after discarding
// 10 input buffer bytes:
map Map_ISO8859_1_to_US_ASCII 10;
```

In case of init and reset operations, if there is no pre-defined init and/or reset operations in the iconv code conversions, only system-defined internal init and reset operations will be executed. The execution of the system-defined internal init and reset operations will clear the system-maintained internal state.

There are three special operators that can be used in the operation:

```
printchr expression;
printhd expression;
printint expression;
```


The above three operators will print out the given expression as a character, a hexadecimal number, and a decimal number, respectively, at the standard error stream. These three operators are for debugging purposes only and should be removed from the final version of the iconv code conversion definition file.

In addition to the above operations, any valid expression separated by a semicolon can be an operation, including an empty operation, denoted by a semicolon alone as an operation.

The map element specifies a direct code conversion mapping by using one or more map pairs. When used, usually many map pairs are used to represent an iconv code conversion definition:

```
map For_US-ASCII_2_ISO8859-1 {
    // one or more map pairs here
    :
    :
}
```

Each map element also can have one or two comma-separated map attribute elements like the following examples:

```
// Map with densely encoded mapping table map type:
map maptype = dense {
    // one or more map pairs here
    :
    :
}

// Map with hash mapping table map type with hash factor 10.
// Only hash mapping table map type can have hash factor. If
// the hash factor is specified with other map types, it will be
// ignored.
map maptype = hash : 10 {
    // one or more map pairs here.
    :
    :
}

// Map with binary search tree based mapping table map type:
map maptype = binary {
    // one more more map pairs here.
    :
```

```
    :
}

// Map with index table based mapping table map type:
map maptype = index {

    // one or more map pairs here.
    :
    :

}

// Map with automatic mapping table map type. If defined,
// system will assign the best possible map type.
map maptype = automatic {

    // one or more map pairs here.
    :
    :

}

// Map with output_byte_length limit set to 2.
map output_byte_length = 2 {

    // one or more map pairs here.
    :
    :

}

// Map with densely encoded mapping table map type and
// output_bute_length limit set to 2:
map maptype = dense, output_byte_length = 2 {

    // one or more map pairs here.
    :
    :

}
```

If no maptype is defined, automatic is assumed. If no output_byte_length is defined, the system figures out the maximum possible output byte length for the mapping by scanning all the possible output values in the mappings. If the actual output byte length scanned is bigger than the defined output_byte_length, the geniconvtbl utility issues an error and stops generating the code conversion binary table(s).

The following are allowed map pairs:

```

// Single mapping. This maps an input character denoted by
// the code value 0x20 to an output character value 0x21:
0x20      0x21

// Multiple mapping. This maps 128 input characters to 128
// output characters. In this mapping, 0x0 maps to 0x10, 0x1 maps
// to 0x11, 0x2 maps to 0x12, ..., and, 0x7f maps to 0x8f:
0x0...0x7f 0x10

// Default mapping. If specified, every undefined input character
// in this mapping will be converted to a specified character
// (in the following case, a character with code value of 0x3f):
default    0x3f;

// Default mapping. If specified, every undefined input character
// in this mapping will not be converted but directly copied to
// the output buffer:
default    no_change_copy;

// Error mapping. If specified, during the code conversion,
// if input buffer contains the byte value, in this case, 0x80,
// the iconv(3) will stop and return (size_t)-1 as the return
// value with EILSEQ set to the errno:
0x80      error;

```

If no default mapping is specified, every undefined input character in the mapping will be treated as an error mapping, and thus the `iconv(3C)` will stop the code conversion and return `(size_t) - 1` as the return value with `EILSEQ` set to the `errno`.

The syntax of the `iconv` code conversion definition in extended BNF is illustrated below:

```

iconv_conversion_definition
    : CONVERSION_NAME '{' definition_element_list '}'
    ;

definition_element_list
    : definition_element ';'
    | definition_element_list definition_element ';'
    ;

definition_element
    : direction
    | condition
    | operation
    | map
    ;

direction
    : 'direction' NAME '{' direction_unit_list '}'

```

```
        | 'direction' '{' direction_unit_list '}'
        ;
direction_unit_list
: direction_unit
| direction_unit_list direction_unit
;

direction_unit
: condition action ';'
| condition NAME ';'
| NAME action ';'
| NAME NAME ';'
| 'true' action ';'
| 'true' NAME ';'
;

action
: direction
| map
| operation
;

condition
: 'condition' NAME '{' condition_list '}'
| 'condition' '{' condition_list '}'
;

condition_list
: condition_expr ';'
| condition_list condition_expr ';'
;

condition_expr
: 'between' range_list
| expr
| 'escapeseq' escseq_list ';'
;

range_list
: range_pair
| range_list ',' range_pair
;

range_pair
: HEXADECIMAL '...' HEXADECIMAL
;

escseq_list
```

```

        : escseq
        | escseq_list ',' escseq
        ;

escseq : HEXADECIMAL
        ;

map    : 'map' NAME '{' map_list '}'
        | 'map' '{' map_list '}'
        | 'map' NAME map_attribute '{' map_list '}'
        | 'map' map_attribute '{' map_list '}'
        ;

map_attribute
    : map_type ',' 'output_byte_length' '=' DECIMAL
    | map_type
    | 'output_byte_length' '=' DECIMAL ',' map_type
    | 'output_byte_length' '=' DECIMAL
    ;

map_type: 'maptype' '=' map_type_name : DECIMAL
        | 'maptype' '=' map_type_name
        ;

map_type_name
    : 'automatic'
    | 'index'
    | 'hash'
    | 'binary'
    | 'dense'
    ;

map_list
    : map_pair
    | map_list map_pair
    ;

map_pair
    : HEXADECIMAL HEXADECIMAL
    | HEXADECIMAL '...' HEXADECIMAL HEXADECIMAL
    | 'default' HEXADECIMAL
    | 'default' 'no_change_copy'
    | HEXADECIMAL 'error'
    ;

operation
    : 'operation' NAME '{' op_list '}'
    | 'operation' '{' op_list '}'
    | 'operation' 'init' '{' op_list '}'

```

```

        | 'operation' 'reset' '{ op_list }'
        ;

op_list : op_unit
        | op_list op_unit
        ;

op_unit : ';'
        | expr ';'
        | 'error' ';'
        | 'error' expr ';'
        | 'discard' ';'
        | 'discard' expr ';'
        | 'output' '=' expr ';'
        | 'direction' NAME ';'
        | 'operation' NAME ';'
        | 'operation' 'init' ';'
        | 'operation' 'reset' ';'
        | 'map' NAME ';'
        | 'map' NAME expr ';'
        | op_if_else
        | 'return' ';'
        | 'printchr' expr ';'
        | 'printhd' expr ';'
        | 'printint' expr ';'
        ;

op_if_else
      : 'if' '(' expr ')' '{ op_list }'
      | 'if' '(' expr ')' '{ op_list }' 'else' op_if_else
      | 'if' '(' expr ')' '{ op_list }' 'else' '{ op_list }'
      ;

expr  : '(' expr ')'
      | NAME
      | HEXADECIMAL
      | DECIMAL
      | 'input' '[' expr ']'
      | 'outputsize'
      | 'inputsize'
      | 'true'
      | 'false'
      | 'input' '==' expr
      | expr '==' 'input'
      | '!' expr
      | '~' expr
      | '-' expr
      | expr '+' expr

```

```

| expr '-' expr
| expr '*' expr
| expr '/' expr
| expr '%' expr
| expr '<<' expr
| expr '>>' expr
| expr '|' expr
| expr '^' expr
| expr '&' expr
| expr '==' expr
| expr '!=' expr
| expr '>' expr
| expr '>=' expr
| expr '<' expr
| expr '<=' expr
| NAME '=' expr
| expr '||' expr
| expr '&&' expr
;

```

Examples EXAMPLE 1 Code conversion from ISO8859-1 to ISO646

```

ISO8859-1%ISO646 {
    // Use dense-encoded internal data structure.
    map matype = dense {
        default      0x3f
        0x0...0x7f   0x0
    };
}

```

EXAMPLE 2 Code conversion from eucJP to ISO-2022-JP

```

// Iconv code conversion from eucJP to ISO-2022-JP

#include <sys/errno.h>

eucJP%ISO-2022-JP {
    operation init {
        codesetnum = 0;
    };

    operation reset {
        if (codesetnum != 0) {
            // Emit state reset sequence, ESC ( J, for
            // ISO-2022-JP.
            output = 0x1b284a;
        }
        operation init;
    };
}

```

EXAMPLE 2 Code conversion from eucJP to ISO-2022-JP (Continued)

```
direction {
  condition {                // JIS X 0201 Latin (ASCII)
    between 0x00...0x7f;
  } operation {
    if (codesetnum != 0) {
      // We will emit four bytes.
      if (outputsize <= 3) {
        error E2BIG;
      }
      // Emit state reset sequence, ESC ( J.
      output = 0x1b284a;
      codesetnum = 0;
    } else {
      if (outputsize <= 0) {
        error E2BIG;
      }
    }
    output = input[0];

    // Move input buffer pointer one byte.
    discard;
  };

  condition {                // JIS X 0208
    between 0xalal...0xfefe;
  } operation {
    if (codesetnum != 1) {
      if (outputsize <= 4) {
        error E2BIG;
      }
      // Emit JIS X 0208 sequence, ESC $ B.
      output = 0x1b2442;
      codesetnum = 1;
    } else {
      if (outputsize <= 1) {
        error E2BIG;
      }
    }
    output = (input[0] & 0x7f);
    output = (input[1] & 0x7f);

    // Move input buffer pointer two bytes.
    discard 2;
  };
};
```


EXAMPLE 2 Code conversion from eucJP to ISO-2022-JP (Continued)

```

condition {          // JIS X 0201 Kana
    between 0x8ea1...0x8edf;
} operation {
    if (codesetnum != 2) {
        if (outputsize <= 3) {
            error E2BIG;
        }
        // Emit JIS X 0201 Kana sequence,
        // ESC ( I.
        output = 0x1b2849;
        codesetnum = 2;
    } else {
        if (outputsize <= 0) {
            error E2BIG;
        }
    }
    output = (input[1] & 127);

    // Move input buffer pointer two bytes.
    discard 2;
};

condition {          // JIS X 0212
    between 0x8fa1a1...0x8ffefe;
} operation {
    if (codesetnum != 3) {
        if (outputsize <= 5) {
            error E2BIG;
        }
        // Emit JIS X 0212 sequence, ESC $ ( D.
        output = 0x1b242844;
        codesetnum = 3;
    } else {
        if (outputsize <= 1) {
            error E2BIG;
        }
    }
    output = (input[1] & 127);
    output = (input[2] & 127);
    discard 3;
};

true operation {    // error
    error EILSEQ;
};
};

```

EXAMPLE 2 Code conversion from eucJP to ISO-2022-JP (Continued)

```
}
```

Files /usr/bin/geniconvtbl
the utility geniconvtbl

/usr/lib/iconv/geniconvtbl/binarytables/*.bt
conversion binary tables

/usr/lib/iconv/geniconvtbl/srcs/*
conversion source files for user reference

See Also [cpp\(1\)](#), [geniconvtbl\(1\)](#), [iconv\(1\)](#), [iconv\(3C\)](#), [iconv_close\(3C\)](#), [iconv_open\(3C\)](#), [attributes\(5\)](#), [environ\(5\)](#)

International Language Environments Guide

Notes The maximum length of HEXADECIMAL and DECIMAL digit length is 128. The maximum length of a variable is 255. The maximum nest level is 16.

Name group – group file

Description The group file is a local source of group information. The group file can be used in conjunction with other group sources, including the NIS maps, `group.byname` and `group.bygid`, or group information stored on an LDAP server. Programs use the `getgrnam(3C)` routines to access this information.

The group file contains a one-line entry for each group recognized by the system, of the form:

groupname:password:gid:user-list

where

groupname The name of the group. A string consisting of lower case alphabetic characters and numeric characters. Neither a colon (:) nor a NEWLINE can be part of a *groupname*. The string can not exceed, `MAXGLEN - 1`, which is usually eight characters.

gid The group's unique numerical ID (GID) within the system.

user-list A comma-separated list of users allowed in the group.

The maximum value of the *gid* field is 2147483647. To maximize interoperability and compatibility, administrators are recommended to assign groups using the range of GIDs below 60000 where possible.

A password can be demanded by `newgrp(1)` if the group password field is not empty. The only way to create a password for a group is to use `passwd(1)`, then cut and paste the password from `/etc/shadow` to `/etc/group`. Group passwords are antiquated and not often used.

During user identification and authentication, the supplementary group access list is initialized sequentially from information in this file. If a user is in more groups than the system is configured for, `{NGROUPS_MAX}`, a warning is given and subsequent group specifications is ignored.

Malformed entries cause routines that read this file to halt, in which case group assignments specified further along are never made. To prevent this from happening, use `grpck(1B)` to check the `/etc/group` database from time to time.

If the number of characters in an entry exceeds 2047, group maintenance commands, such as `groupdel(1M)` and `groupmod(1M)`, fail.

Previous releases used a group entry beginning with a '+' (plus sign) or '-' (minus sign) to selectively incorporate entries from a naming service source (for example, an NIS map or data from an LDAP server) for group. If still required, this is supported by specifying `group:compat` in `nsswitch.conf(4)`. The `compat` source might not be supported in future releases. A possible sources is `files` followed by `ldap`. This has the effect of incorporating information from an LDAP server after the group file.

Examples EXAMPLE 1 An Example group File

The following is an example of a group file:

```
root::0:root
stooges:q.mJzTnu8icF.:10:larry,moe,curly
```

and the sample group entry from `nsswitch.conf`:

```
group: files ldap
```

With these entries, the group `stooges` has members `larry`, `moe`, and `curly`, and all groups listed on the LDAP server are effectively incorporated after the entry for `stooges`.

If the group file was:

```
root::0:root
stooges:q.mJzTnu8icF.:10:larry,moe,curly
+:
```

and the group entry from `nsswitch.conf`:

```
group: compat
```

all the groups listed in the NIS `group.bygid` and `group.byname` maps would be effectively incorporated after the entry for `stooges`.

See Also [groups\(1\)](#), [grpck\(1B\)](#), [newgrp\(1\)](#), [groupadd\(1M\)](#), [groupdel\(1M\)](#), [groupmod\(1M\)](#), [getgrnam\(3C\)](#), [initgroups\(3C\)](#), [nsswitch.conf\(4\)](#), [unistd.h\(3HEAD\)](#)

Oracle Solaris Administration: Common Tasks

Notes An administrator must have `solaris.group.manage` authorization to add a new group. An administrator can add a user to any group or modify any group for which it has a matching authorization of the form `solaris.group.assign/groupname`. An administrator must have both `solaris.group.manage` and either `solaris.group.assign` or an authorization of the form `solaris.group.assign/groupname` to delete a group.

Name gsscred.conf – Generic Security Services credential configuration file

Synopsis /etc/gss/gsscred.conf

Description The gsscred.conf file contains GSS credential information including options that can be set by the system administrator.

The options that are in this file include:

SYSLOG_UID_MAPPING=yes

If this option is set to yes, GSS cred to Unix cred mapping results will be logged to [syslog\(3C\)](#) at level auth.debug.

Files /etc/gss/gsscred.conf Contains GSS credential information.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Committed

See Also [gsscred\(1M\)](#), [gssd\(1M\)](#), [syslog\(3C\)](#), [krb5.conf\(4\)](#), [attributes\(5\)](#), [kerberos\(5\)](#)

Name hba.conf – configuration file for the HBAAPI library

Description The `/etc/hba.conf` file is used to specify the Vendor-Specific Libraries that are installed on the system. This file is used by the Common Library to load the individual VSLs when `HBA_LoadLibrary(3HBAAPI)` is called. If changes are made to the file while the library is in use, the library should be freed and reloaded. A version 1 VSL is compatible only with a version 1 Common Library. A version 2 VSL is compatible with both a version 1 and a version 2 Common Library.

Each VSL entry is a single line of the form:

```
"name"      "library path"
```

where:

name is the description of library. The library name should be prepended with the domain of the manufacturer of the library.

library path is the absolute path to the shared object library file.

Examples EXAMPLE 1 Contents of `/etc/hba.conf`

```
#
# This file contains names and references to HBA libraries
#
# Format:
#
# <library name> <library pathname>
#
# The library name should be prepended with the domain of
# the manufacturer or driver author.
com.sun.fchba32      /usr/lib/libsun_fc.so.1
com.sun.fchba64      /usr/lib/sparcv9/libsun_fc.so.1
```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Committed
Standard	FC-MI 1.92 (API version 1) FC-HBA Version 4 (API version 2)

See Also [HBA_LoadLibrary\(3HBAAPI\)](#), [libhbaapi\(3LIB\)](#), [attributes\(5\)](#)

Bugs The HBAAPI is provided in both 32- and 64-bit versions, but only one configuration file is specified. As a result, both 32- and 64-bit VSL libraries must be specified within the same file. When using the 32-bit Common Library, the 64-bit VSLs will fail to load. When using the

64-bit Common Library, the 32-bit VSLs will fail to load. These failures are silently ignored by the Common Library during normal usage, but can result in warning messages when running client applications in a debugger.

Name holidays – prime/nonprime table for the accounting system

Synopsis /etc/acct/holidays

Description The /etc/acct/holidays file specifies prime time hours and holidays. Holidays and weekends are considered non-prime time hours.

/etc/acct/holidays is used by the accounting system.

All lines beginning with an * are comments.

The /etc/acct/holidays file consists of two sections. The first non-comment line defines the current year and the start time of prime and non-prime time hours, in the form of:

```
current_year prime_start non_prime_start
```

Specify *prime_start* and *non_prime_start* times in the range of 0000 to 2400.

The remaining non-comment lines define the holidays in the form of:

```
month/day company_holiday
```

Of these two fields, only the *month/day* is actually used by the accounting system programs.

The /etc/acct/holidays file must be updated every year.

Examples EXAMPLE 1 An Example of the /etc/acct/holidays File

The following is an example of the /etc/acct/holidays file:

```
* Prime/Nonprime Table for the accounting system
*
* Curr      Prime      Non-Prime
* Year      Start      Start
*
  1991      0830      1800
*
* only the first column (month/day) is significant.
*
* month/day  Company Holiday
*
  1/1        New Years Day
  5/30       Memorial Day
  7/4        Indep. Day
  9/5        Labor Day
  11/24      Thanksgiving Day
  11/25      day after Thanksgiving
  12/25      Christmas
  12/26      day after Christmas
```


See Also [acct\(1M\)](#)

Name hosts – host name database

Synopsis /etc/inet/hosts

/etc/hosts

/etc/inet/ipnodes

Description The `hosts` file is a local database that associates the names of hosts with their Internet Protocol (IP) addresses. An IP address can be in either IPv4 or IPv6 format. The `hosts` file can be used in conjunction with, or instead of, other hosts databases, including the Domain Name System (DNS), the NIS `hosts map`, or information from an LDAP server. Programs use library interfaces to access information in the `hosts` file.

Note that `/etc/hosts` and `/etc/inet/ipnodes` are symbolic links to `/etc/inet/hosts`.

The `hosts` file has one entry for each IP address of each host. If a host has more than one IP address, it will have one entry for each, on consecutive lines. The format of each line is:

IP-address official-host-name nicknames . . .

Items are separated by any number of SPACE and/or TAB characters. The first item on a line is the host's IP address. The second entry is the host's official name. Subsequent entries on the same line are alternative names for the same machine, or “nicknames.” Nicknames are optional.

For a host with more than one IP address, consecutive entries for these addresses may contain the same or differing nicknames. Different nicknames are useful for assigning distinct names to different addresses.

A call to `gethostbyname(3NSL)` returns a `hostent` structure containing the union of all IPv4 addresses and nicknames from each line containing a matching official name or nickname. A call to `getipnodebyname(3SOCKET)` is similar, but is capable of returning `hostent` structures containing IPv4 and IPv6 addresses. Applications might prefer to use the address-family independent `getaddrinfo(3SOCKET)` API for name-to-address lookups.

A `#` indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search the file.

Network addresses are written in one of two ways:

- The conventional “decimal dot” notation and interpreted using the `inet_addr` routine from the Internet address manipulation library, `inet(3SOCKET)`.
- The IP Version 6 protocol [IPV6], defined in RFC 1884 and interpreted using the `inet_pton()` routine from the Internet address manipulation library. See `inet(3SOCKET)`.

This interface supports node names as defined in Internet RFC 952, which states:

A “name” (Net, Host, Gateway, or Domain name) is a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus sign (-), and period (.). Note that periods are only allowed when they serve to delimit components of “domain style names”. (See RFC 921, “Domain Name System Implementation Schedule,” for background). No blank or space characters are permitted as part of a name. No distinction is made between uppercase and lowercase. The first character must be an alpha character [or a digit. (RFC 1123 relaxed RFC 952’s limitation of the first character to only alpha characters.)] The last character must not be a minus sign or period.

Host names must not consist of numbers only. A host name must contain at least one alphabetical or special character.

Although the interface accepts host names longer than 24 characters for the host portion (exclusive of the domain component), choosing names for hosts that adhere to the 24 character restriction will insure maximum interoperability on the Internet.

A host which serves as a GATEWAY should have “-GATEWAY” or “-GW” as part of its name. Hosts which do not serve as Internet gateways should not use “-GATEWAY” and “-GW” as part of their names. A host which is a TAC should have “-TAC” as the last part of its host name, if it is a DoD host. Single character names or nicknames are not allowed.

Examples EXAMPLE 1 Example hosts File Entry

The following is a typical line from the hosts file:

```
192.9.1.20      gaia                # John Smith
```

EXAMPLE 2 Example IPv6 Address Entry

The following is an example of an IPv6 hosts entry:

```
2001:0db8:3c4d:55:a00:20ff:fe8e:f3ad myhost # John Smith
```

See Also [gethostbyname\(3NSL\)](#), [getipnodebyname\(3SOCKET\)](#), [inet\(3SOCKET\)](#), [nsswitch.conf\(4\)](#), [resolv.conf\(4\)](#)

Braden, B., editor, RFC 1123, *Requirements for Internet Hosts - Application and Support*, Network Working Group, October, 1989.

Harrenstien, K., Stahl, M., and Feinler, E., RFC 952, *DOD Internet Host Table Specification*, Network Working Group, October 1985.

Hinden, R., and Deering, S., editors, RFC 1884, *IP Version 6 Addressing Architecture*, Network Working Group, December, 1995.

Postel, Jon, RFC 921, *Domain Name System Implementation Schedule (Revised)*, Network Working Group, October 1984.

Notes `/etc/inet/hosts` is the official SVR4 name of the `hosts` file. The symbolic link `/etc/hosts` exists for BSD compatibility.

The symbolic link `/etc/net/ipnodes` exists for backwards compatibility with previous Solaris releases.

Name hosts.equiv, rhosts – trusted remote hosts and users

Description The `/etc/hosts.equiv` and `.rhosts` files provide the “remote authentication” database for `rlogin(1)`, `rsh(1)`, `rcp(1)`, and `rcmd(3SOCKET)`. The files specify remote hosts and users that are considered “trusted”. Trusted users are allowed to access the local system without supplying a password. The library routine `ruserok()` (see `rcmd(3SOCKET)`) performs the authentication procedure for programs by using the `/etc/hosts.equiv` and `.rhosts` files. The `/etc/hosts.equiv` file applies to the entire system, while individual users can maintain their own `.rhosts` files in their home directories.

These files bypass the standard password-based user authentication mechanism. To maintain system security, care must be taken in creating and maintaining these files.

The remote authentication procedure determines whether a user from a remote host should be allowed to access the local system with the identity of a local user. This procedure first checks the `/etc/hosts.equiv` file and then checks the `.rhosts` file in the home directory of the local user who is requesting access. Entries in these files can be of two forms. Positive entries allow access, while negative entries deny access. The authentication succeeds when a matching positive entry is found. The procedure fails when the first matching negative entry is found, or if no matching entries are found in either file. The order of entries is important. If the files contain both positive and negative entries, the entry that appears first will prevail. The `rsh(1)` and `rcp(1)` programs fail if the remote authentication procedure fails. The `rlogin` program falls back to the standard password-based login procedure if the remote authentication fails.

Both files are formatted as a list of one-line entries. Each entry has the form:

```
hostname [username]
```

Hostnames must be the official name of the host, not one of its nicknames.

Negative entries are differentiated from positive entries by a ‘-’ character preceding either the *hostname* or *username* field.

Positive Entries If the form:

```
hostname
```

is used, then users from the named host are trusted. That is, they may access the system with the same user name as they have on the remote system. This form may be used in both the `/etc/hosts.equiv` and `.rhosts` files.

If the line is in the form:

```
hostname username
```

then the named user from the named host can access the system. This form may be used in individual `.rhosts` files to allow remote users to access the system *as a different local user*. If this form is used in the `/etc/hosts.equiv` file, the named remote user will be allowed to access the system as *any* local user.

`netgroup(4)` can be used in either the *hostname* or *username* fields to match a number of hosts or users in one entry. The form:

`+@netgroup`

allows access from all hosts in the named netgroup. When used in the *username* field, netgroups allow a group of remote users to access the system as a particular local user. The form:

`hostname +@netgroup`

allows all of the users in the named netgroup from the named host to access the system as the local user. The form:

`+@netgroup1 +@netgroup2`

allows the users in *netgroup2* from the hosts in *netgroup1* to access the system as the local user.

The special character '+' can be used in place of either *hostname* or *username* to match any host or user. For example, the entry

`+`

will allow a user from any remote host to access the system with the same username. The entry

`+ username`

will allow the named user from any remote host to access the system. The entry

`hostname +`

will allow any user from the named host to access the system as the local user.

Negative Entries Negative entries are preceded by a '-' sign. The form:

`-hostname`

will disallow all access from the named host. The form:

`-@netgroup`

means that access is explicitly disallowed from all hosts in the named netgroup. The form:

`hostname -username`

disallows access by the named user only from the named host, while the form:

`+ -@netgroup`

will disallow access by all of the users in the named netgroup from all hosts.

Search Sequence To help maintain system security, the `/etc/hosts.equiv` file is not checked when access is being attempted for superuser. If the user attempting access is not the superuser, `/etc/hosts.equiv` is searched for lines of the form described above.

Checks are made for lines in this file in the following order:

1. `+`
2. `+%netgroup`
3. `-%netgroup`
4. `-%hostname`
5. `hostname`

The user is granted access if a positive match occurs. Negative entries apply only to `/etc/hosts.equiv` and may be overridden by subsequent `.rhosts` entries.

If no positive match occurred, the `.rhosts` file is then searched if the user attempting access maintains such a file. This file is searched whether or not the user attempting access is the superuser. As a security feature, the `.rhosts` file must be owned by the user who is attempting access.

Checks are made for lines in `.rhosts` in the following order:

1. `+`
2. `+%netgroup`
3. `-%netgroup`
4. `-%hostname`
5. `hostname`

Files

<code>/etc/hosts.equiv</code>	system trusted hosts and users
<code>~/.rhosts</code>	user's trusted hosts and users

See Also [rcp\(1\)](#), [rlogin\(1\)](#), [rsh\(1\)](#), [rcmd\(3SOCKET\)](#), [hosts\(4\)](#), [netgroup\(4\)](#), [passwd\(4\)](#)

Warnings Positive entries in `/etc/hosts.equiv` that include a *username* field (either an individual named user, a netgroup, or '+' sign) should be used with extreme caution. Because `/etc/hosts.equiv` applies system-wide, these entries allow one, or a group of, remote users to access the system *as any local user*. This can be a security hole. For example, because of the search sequence, an `/etc/hosts.equiv` file consisting of the entries

```
+
-hostxxx
```

will not deny access to "hostxxx".

Name `ib` – InfiniBand device driver configuration files

Description The InfiniBand (IB) bus is an I/O transport based on switched fabrics. IB devices are managed by the [ib\(7D\)](#) nexus driver. There are three categories of InfiniBand devices:

- IB port/IB VPPA/IB HCA_SVC devices
- IB IOC devices
- IB Pseudo devices

The IB port/IB VPPA/IB HCA_SVC devices are enumerated by way of the `ib.conf` file. See [ib\(7D\)](#).

The IB IOC devices are enumerated using the InfiniBand Device management class. See [ibdm\(7D\)](#).

For devices not in these two categories, most notably IB Pseudo devices, the driver must provide configuration files to inform the system of the IB devices to be created. Configuration parameters are represented in the form of name value pairs you can retrieve using the DDI property interfaces. See [ddi_prop_op\(9F\)](#) for details.

Configuration files for IB device drivers must identify the parent driver explicitly as `ib`, and must create a string array property called `unit-address` which is unique to this entry in the configuration file. Drivers name `ibport` and `ioc` are reserved by [ib\(7D\)](#) and should not be used.

Each entry in the configuration file creates a prototype `devinfo` node. Each node is assigned a unit address which is determined by the value of the `unit-address` property. This property is only applicable to children of the IB parent and is required. See [driver.conf\(4\)](#) for further details on configuration file syntax.

Examples Example 1: Sample configuration file

Here is a configuration file called `ibgen.conf` for an IB device driver that implements a generic IB driver. This file creates a node called `ibgen`.

```
#
# Copyright 2002-2003 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
#ident "@(#)ibgen.conf      1.3      03/05/01 SMI"
name="ibgen" parent="ib" unit-address="0";
```

See Also [driver.conf\(4\)](#), [ib\(7D\)](#), [ibtl\(7D\)](#), [ddi_prop_op\(9F\)](#)

Name idnkit.pc – meta information data file for libidnkit

Synopsis /usr/lib/pkgconfig/idnkit.pc

Description idnkit.pc is the meta information data file for [libidnkit\(3LIB\)](#). Use pkg-config(1) to retrieve the defined values such as compile and link flags for the library.

Examples EXAMPLE 1 Using idnkit.pc through pkg-config

The following command yields compile and link flags, if any, for [libidnkit\(3LIB\)](#):

```
example% pkg-config --cflags --libs idnkit
```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	library/idnkit/header-idnkit
Interface Stability	Volatile

See Also [idn_decodename\(3EXT\)](#), [idn_decodename\(3EXT\)](#), [idn_decodename\(3EXT\)](#), [libidnkit\(3LIB\)](#), [attributes\(5\)](#), [environ\(5\)](#), [iconv\(5\)](#)

Name ike.config – configuration file for IKE policy

Synopsis /etc/inet/ike/config

Description The /etc/inet/ike/config file contains rules for matching inbound IKE requests. It also contains rules for preparing outbound IKE requests.

You can test the syntactic correctness of an /etc/inet/ike/config file by using the -c or -f options of `in.iked(1M)`. You must use the -c option to test a config file. You might need to use the -f option if it is not in /etc/inet/ike/config.

Lexical Components On any line, an unquoted # character introduces a comment. The remainder of that line is ignored. Additionally, on any line, an unquoted // sequence introduces a comment. The remainder of that line is ignored.

There are several types of lexical tokens in the `ike.config` file:

num

A decimal, hex, or octal number representation is as in C.

IPaddr/prefix/range

An IPv4 or IPv6 address with an optional /*NNN* suffix, (where *NNN* is a *num*) that indicates an address (CIDR) prefix (for example, `10.1.2.0/24`). An optional /*ADDR* suffix (where *ADDR* is a second IP address) indicates an address/mask pair (for example, `10.1.2.0/255.255.255.0`). An optional -*ADDR* suffix (where *ADDR* is a second IPv4 address) indicates an inclusive range of addresses (for example, `10.1.2.0-10.1.2.255`). The / or - can be surrounded by an arbitrary amount of white space.

XXX | YYY | ZZZ

Either the words *XXX*, *YYY*, or *ZZZ*, for example, {yes,no}.

p1-id-type

An IKE phase 1 identity type. IKE phase 1 identity types include:

dn, DN
dns, DNS
fqdn, FQDN
gn, GN
ip, IP
ipv4
ipv4_prefix
ipv4_range
ipv6
ipv6_prefix
ipv6_range
mbox, MBOX
user_fqdn

Not all phase 1 identity types are supported.

"string"

A quoted string.

Examples include: "Label foo", or "C=US, OU=Sun Microsystems\\, Inc., N=olemcd@eng.example.com"

A backslash (\) is an escape character. If the string needs an actual backslash, two must be specified.

cert-sel

A certificate selector, a *string* which specifies the identities of zero or more certificates. The specifiers can conform to X.509 naming conventions.

A *cert-sel* can also use various shortcuts to match either subject alternative names, the filename or slot of a certificate in /etc/inet/ike/publickeys, or even the ISSUER. For example:

```
"SLOT=0"
"EMAIL=postmaster@domain.org"
"webmaster@domain.org" # Some just work w/o TYPE=
"IP=10.0.0.1"
"10.21.11.11"          # Some just work w/o TYPE=
"DNS=www.domain.org"
"mailhost.domain.org" # Some just work w/o TYPE=
"ISSUER=C=US, O=Sun Microsystems\\, Inc., CN=Sun CA"
```

Any *cert-sel* preceded by the character ! indicates a negative match, that is, not matching this specifier. These are the same kind of strings used in [ikecert\(1M\)](#).

ldap-list

A quoted, comma-separated list of LDAP servers and ports.

For example, "ldap1.example.com", "ldap1.example.com:389", "ldap1.example.com:389,ldap2.example.com".

The default port for LDAP is 389.

parameter-list

A list of parameters.

label

A sensitivity label, either as a quoted string containing a human-readable label or as a hexadecimal format internal text label. See [labels\(5\)](#) for more information.

For example, PUBLIC, 0x0002-08-08.

File Body Entries There are four main types of entries:

- global parameters
- IKE phase 1 transform defaults
- IKE rule defaults
- IKE rules

The global parameter entries are as follows:

`cert_root` *cert-sel*

The X.509 distinguished name of a certificate that is a trusted root CA certificate. It must be encoded in a file in the `/etc/inet/ike/publickeys` directory. It must have a CRL in `/etc/inet/ike/crls`. Multiple `cert_root` parameters aggregate.

`cert_trust` *cert-sel*

Specifies an X.509 distinguished name of a certificate that is self-signed, or has otherwise been verified as trustworthy for signing IKE exchanges. It must be encoded in a file in `/etc/inet/ike/publickeys`. Multiple `cert_trust` parameters aggregate.

`expire_timer` *integer*

The number of seconds to let a not-yet-complete IKE Phase I (Main Mode) negotiation linger before deleting it. Default value: 300 seconds.

`ignore_crls`

If this keyword is present in the file, `in.iked(1M)` ignores Certificate Revocation Lists (CRLs) for root CAs (as given in `cert_root`)

`label_aware`

This keyword can only be used on systems where Trusted Extensions are enabled. If this keyword is present in the file, `in.iked(1M)` attaches sensitivity label extensions to security associations, consults the `tnrhdb` for information about the clearances of peers, and negotiates labels with label-aware peers. Several additional keywords modify the behavior of `in.iked` in label-aware mode.

`ldap_server` *ldap-list*

A list of LDAP servers to query for certificates. The list can be additive.

`pkcs11_path` *string*

The string that follows is a name of a shared object (`.so`) that implements the PKCS#11 standard. The name is passed directly into `dlopen(3C)` for linking, with all of the semantics of that library call. By default, `in.iked(1M)` runs the same ISA as the running kernel, so a library specified using `pkcs11_path` and an absolute pathname *must* match the same ISA as the kernel. One can use the `start/exec` SMF property (see `svccfg(1M)`) to change `in.iked`'s ISA, but it is not recommended.

If this setting is not present, the default value is set to `libpkcs11.so`. Most cryptographic providers go through the default library, and this parameter should only be used if a specialized provider of IKE-useful cryptographic services cannot interface with the Solaris Cryptographic Framework. See `cryptoadm(1M)`.

This option is now deprecated, and might be removed in a future release.

proxy string

The string following this keyword must be a URL for an HTTP proxy, for example, `http://proxy:8080`.

retry_limit integer

The number of retransmits before any IKE negotiation or Dead Peer Detection (DPD) process is aborted. Default value: 5 times.

retry_timer_init integer or float

The initial interval (in seconds) between retransmits. This interval is doubled until the `retry_timer_max` value (see below) is reached. Default value: 0.5 seconds.

retry_timer_max integer or float

The maximum interval (in seconds) between retransmits. Used for both IKE and Dead Peer Detection (DPD). The doubling retransmit interval stops growing at this limit. Default value: 30 seconds.

This value is never reached with the default configuration. The longest interval is $8 (0.5 * 2^{(5 - 1)})$ seconds.

socks string

The string following this keyword must be a URL for a SOCKS proxy, for example, `socks://socks-proxy`.

use_http

If this keyword is present in the file, `in.iked(1M)` uses HTTP to retrieve Certificate Revocation Lists (CRLs).

wire_label inner wire_label label wire_label none label

This keyword can only be used if `label_aware` mode is selected and defines how IKE communicates with label-aware peers. `wire_label inner` reuses the inner label, and sends key management traffic as `admin_low`. `wire_label label` uses the specified label for key management traffic and uses that label as the outer label for all encrypted traffic. The label is attached to each packet as a CIPSO label. `wire_label none label` does not attach a CIPSO label to either key management traffic or traffic sent as a given SA, but otherwise treats the traffic as if it had the given label.

The following IKE phase 1 transform parameters can be prefigured using file-level defaults. Values specified within any given transform override these defaults.

The IKE phase 1 transform defaults are as follows:

p1_lifetime_secs num

The proposed default lifetime, in seconds, of an IKE phase 1 security association (SA).

p1_nonce_len num

The length in bytes of the phase 1 (main mode) nonce data. This cannot be specified on a per-rule basis.

The following IKE rule parameters can be prefigured using file-level defaults. Values specified within any given rule override these defaults, unless a rule cannot.

p2_lifetime_secs num

The proposed default lifetime, in seconds, of an IKE phase 2 security association (SA). This value is optional. If omitted, a default value is used.

p2_softlife_secs num

The soft lifetime of a phase 2 SA, in seconds. If this value is specified, the SA soft expires after the number of seconds specified by `p2_softlife_secs`. This causes `in.iked` to renegotiate a new phase 2 SA before the original SA expires.

This value is optional, if omitted soft expiry occurs after 90% of the lifetime specified by `p2_lifetime_secs`. The value specified by `p2_softlife_secs` is ignored if `p2_lifetime_secs` is not specified.

Setting `p2_softlife_secs` to the same value as `p2_lifetime_secs` disables soft expires.

p2_idletime_secs num

The idle lifetime of a phase 2 SA, in seconds. If the value is specified, the value specifies the lifetime of the SA, if the security association is not used before the SA is revalidated.

p2_lifetime_kb num

The lifetime of an SA can optionally be specified in kilobytes. This parameter specifies the default value. If lifetimes are specified in both seconds and kilobytes, the SA expires when either the seconds or kilobyte thresholds are passed.

p2_softlife_kb num

This value is the number of kilobytes that can be protected by an SA before a soft expire occurs (see `p2_softlife_secs`, above).

This value is optional. If omitted, soft expiry occurs after 90% of the lifetime specified by `p2_lifetime_kb`. The value specified by `p2_softlife_kb` is ignored if `p2_lifetime_kb` is not specified.

p2_nonce_len num

The length in bytes of the phase 2 (quick mode) nonce data. This cannot be specified on a per-rule basis.

local_id_type p1-id-type

The local identity for IKE requires a type. This identity type is reflected in the IKE exchange. It is needed because a single certificate can contain multiple values for use in IKE phase 1. The type can be one of the following:

- an IP address (for example, `10.1.1.2`)
- DNS name, also known as FQDN (for example, `test.domain.com`)
- MBOX, also known as `USER_FQDN` or RFC 822 name (for example, `root@domain.com`)
- DN-A X.509 distinguished name (for example, `C=US, O=Sun Microsystems\, Inc., CN=Sun Test cert`)

`p1_xform` '{ parameter-list }

A phase 1 transform specifies a method for protecting an IKE phase 1 exchange. An initiator offers up lists of phase 1 transforms, and a receiver is expected to only accept such an entry if it matches one in a phase 1 rule. There can be several of these, and they are additive. There must be either at least one phase 1 transform in a rule or a global default phase 1 transform list. In a configuration file without a global default phase 1 transform list *and* a rule without a phase, transform list is an invalid file. Unless specified as optional, elements in the parameter-list must occur exactly once within a given transform's parameter-list:

`oakley_group` *number*

The Oakley Diffie-Hellman group used for IKE SA key derivation. The group numbers are defined in RFC 2409, Appendix A, RFC 3526, RFC 4753, and RFC 5114. Acceptable values are currently:

1 (768-bit)
 2 (1024-bit)
 5 (1536-bit)
 14 (2048-bit)
 15 (3072-bit)
 16 (4096-bit)
 19 (256-bit ECP defined by RFC4753)
 20 (384-bit ECP defined by RFC4753)
 21 (521-bit ECP defined by RFC4753)
 22 (1024-bit MODP defined by RFC5114)
 23 (2048-bit MODP defined by RFC5114)
 24 (2048-bit MODP defined by RFC5114)
 25 (192-bit ECP defined by RFC5114)
 26 (224-bit ECP defined by RFC5114)

`encr_alg` {3des, 3des-cbc, blowfish, blowfish-cdc, des, des-cbc, aes, aes-cbc}
 An encryption algorithm.

The algorithm names without mode specification are all synonyms for the CBC modes, for example, aes - cbc and aes are the same.

Only aes/aes - cbc allows optional key-size setting, using the (low . . high) syntax, the same as specified in `ipsecconf(1M)` for the `keylen` specifier. To specify a single AES key size, the low value must equal the high value or single number must be used. If no range is specified, all three AES key sizes are allowed.

`auth_alg` {md5, sha, sha1, sha256, sha384, sha512}
 An authentication algorithm.

`auth_method` {preshared, rsa_sig, rsa_encrypt, dss_sig}
 The authentication method used for IKE phase 1.

p1_lifetime_secs num

Optional. The lifetime for a phase 1 SA.

p2_lifetime_secs num

If configuring the kernel defaults is not sufficient for different tasks, this parameter can be used on a per-rule basis to set the IPsec SA lifetimes in seconds.

p2_pfs num

Use perfect forward secrecy for phase 2 (quick mode). If selected, the oakley group specified is used for phase 2 PFS. Acceptable values are the same as for the oakley_group parameter listed above.

An IKE rule starts with a right-curly-brace (`{`), ends with a left-curly-brace (`}`), and has the following parameters in between:

label string

Required parameter. The administrative interface to `in.ikeyd` looks up phase 1 policy rules with the label as the search string. The administrative interface also converts the label into an index, suitable for an extended ACQUIRE message from PF_KEY - effectively tying IPsec policy to IKE policy in the case of a node initiating traffic. Only one `label` parameter is allowed per rule.

local_addr <IPaddr/prefix/range>

Required parameter. The local address, address prefix, or address range for this phase 1 rule. Multiple `local_addr` parameters accumulate within a given rule.

remote_addr <IPaddr/prefix/range>

Required parameter. The remote address, address prefix, or address range for this phase 1 rule. Multiple `remote_addr` parameters accumulate within a given rule.

local_id_type p1-id-type

Which phase 1 identity type to use for this rule. The supported p1-id-types are described in section for the global parameter `local_id_type`. Within a given rule, all phase 1 transforms must either use preshared or non-preshared authentication (they can not be mixed).

For rules with preshared authentication, the `local_id_type` parameter is optional, and defaults to IP. For rules which use non-preshared authentication, the `local_id_type` preshared authentication, the `local_id_type` parameter parameter is required. Multiple `local_id_type` parameters within a rule are not allowed.

For rules with preshared authentication, the `local_id_type` parameter is optional, and defaults to IP. For rules which use non-preshared authentication, the `local_id_type` parameter is required. Multiple `local_id_type` parameters within a rule are not allowed.

local_id cert-sel

Disallowed for preshared authentication method; required parameter for non-preshared authentication method. The local identity string or certificate selector. Only one local identity per rule is used, the first one stated.

remote_id cert-sel

Disallowed for preshared authentication method; required parameter for non-preshared authentication method. Selector for which remote phase 1 identities are allowed by this rule. Multiple *remote_id* parameters accumulate within a given rule. If a single empty string ("") is given, then this accepts any remote ID for phase 1. It is recommended that certificate trust chains or address enforcement be configured strictly to prevent a breakdown in security if this value for *remote_id* is used.

p2_lifetime_secs num

If configuring the kernel defaults is not sufficient for different tasks, this parameter can be used on a per-rule basis to set the IPsec SA lifetimes in seconds.

p2_pfs num

Use perfect forward secrecy for phase 2 (quick mode). If selected, the oakley group specified is used for phase 2 PFS. Acceptable values are:

- 0 (do not use Perfect Forward Secrecy for IPsec SAs)
- 1 (768-bit)
- 2 (1024-bit)
- 5 (1536-bit)
- 14 (2048-bit)
- 15 (3072-bit)
- 16 (4096-bit)

p1_xform { parameter-list }

A phase 1 transform specifies a method for protecting an IKE phase 1 exchange. An initiator offers up lists of phase 1 transforms, and a receiver is expected to only accept such an entry if it matches one in a phase 1 rule. There can be several of these, and they are additive. There must be either at least one phase 1 transform in a rule or a global default phase 1 transform list. A *ike.config* file without a global default phase 1 transform list *and* a rule without a phase 1 transform list is an invalid file. Elements within the *parameter-list*; unless specified as optional, must occur exactly once within a given transform's *parameter-list*:

oakley_group number

The Oakley Diffie-Hellman group used for IKE SA key derivation. Acceptable values are currently:

- 1 (768-bit)
- 2 (1024-bit)
- 5 (1536-bit)
- 14 (2048-bit)
- 15 (3072-bit)
- 16 (4096-bit)

`encr_alg` {3des, 3des-cbc, blowfish, blowfish-cdc, des, des-cbc, aes, aes-cbc}
An encryption algorithm, as in [ipsecconf\(1M\)](#). However, of the ciphers listed above, only aes and aes-cbc allow optional key-size setting, using the “low value-to-high value” syntax. To specify a single AES key size, the low value must equal the high value. If no range is specified, all three AES key sizes are allowed.

`auth_alg` {md5, sha, sha1}
An authentication algorithm, as specified in [ipseckey\(1M\)](#).

`auth_method` {preshared, rsa_sig, rsa_encrypt, dss_sig}
The authentication method used for IKE phase 1.

`multi_label`
Optional. Useful only on systems with Trusted Extensions enabled. Override `tnrhdh` and assume peer is label-aware.

`p1_lifetime_secs` *num*
Optional. The lifetime for a phase 1 SA.

`single_label`
Optional. Useful only on systems with Trusted Extensions enabled. Override `tnrhdh` and assume peer is not label-aware.

Examples

EXAMPLE 1 A Sample `ike.config` File
The following is an example of an `ike.config` file:

```
### BEGINNING OF FILE

### First some global parameters...

### certificate parameters...

# Root certificates. I SHOULD use a full Distinguished Name.
# I must have this certificate in my local filesystem, see ikecert(1m).
cert_root "C=US, O=Sun Microsystems\\, Inc., CN=Sun CA"

# Explicitly trusted certs that need no signatures, or perhaps
# self-signed ones. Like root certificates, use full DNS for them
# for now.
cert_trust "EMAIL=root@domain.org"

# Where do I send LDAP requests?
ldap_server "ldap1.domain.org,ldap2.domain.org:389"

## phase 1 transform defaults...

p1_lifetime_secs 14400
p1_nonce_len 20
```

EXAMPLE 1 A Sample ike.config File (Continued)

```

## Parameters that might also show up in rules.

p1_xform { auth_method preshared oakley_group 5 auth_alg sha
          encr_alg 3des }
p2_pfs 2

### Now some rules...

{
  label "simple inheritor"
  local_id_type ip
  local_addr 10.1.1.1
  remote_addr 10.1.1.2
}
{
  label "simple inheritor IPv6"
  local_id_type ipv6
  local_addr fe80::a00:20ff:fe7d:6
  remote_addr fe80::a00:20ff:fe7d:3780
}

{
  # an index-only rule.  If I'm a receiver, and all I
  # have are index-only rules, what do I do about inbound IKE requests?
  # Answer:  Take them all!

  label "default rule"
  # Use whatever "host" (e.g. IP address) identity is appropriate
  local_id_type ipv4

  local_addr 0.0.0.0/0
  remote_addr 0.0.0.0/0

  p2_pfs 5

  # Now I'm going to have the p1_xforms
  p1_xform
  {auth_method preshared oakley_group 5 auth_alg md5 encr_alg \
    blowfish } p1_xform
  {auth_method preshared oakley_group 5 auth_alg md5 encr_alg 3des }

  # After said list, another keyword (or a ')') stops xform
  # parsing.

```

EXAMPLE 1 A Sample ike.config File (Continued)

```
}

{
  # Let's try something a little more conventional.

  label "host to .80 subnet"
  local_id_type ip
  local_id "10.1.86.51"

  remote_id "" # Take any, use remote_addr for access control.

  local_addr 10.1.86.51
  remote_addr 10.1.80.0/24

  p1_xform
  { auth_method rsa_sig oakley_group 5 auth_alg md5 encr_alg 3des }
  p1_xform
  { auth_method rsa_sig oakley_group 5 auth_alg md5 encr_alg \
    blowfish }
  p1_xform
  { auth_method rsa_sig oakley_group 5 auth_alg sha1 encr_alg 3des }
  p1_xform
  { auth_method rsa_sig oakley_group 5 auth_alg sha1 encr_alg \
    blowfish }
}

{
  # Let's try something a little more conventional, but with ipv6.

  label "host to fe80::/10 subnet"
  local_id_type ip
  local_id "fe80::a00:20ff:fe7d:6"

  remote_id "" # Take any, use remote_addr for access control.

  local_addr fe80::a00:20ff:fe7d:6
  remote_addr fe80::/10

  p1_xform
  { auth_method rsa_sig oakley_group 5 auth_alg md5 encr_alg 3des }
  p1_xform
  { auth_method rsa_sig oakley_group 5 auth_alg md5 encr_alg \
    blowfish }
  p1_xform
  { auth_method rsa_sig oakley_group 5 auth_alg sha1 encr_alg \
    3des }
```

EXAMPLE 1 A Sample ike.config File (Continued)

```

    p1_xform
    { auth_method rsa_sig oakley_group 5 auth_alg sha1 encr_alg \
      blowfish }
}

{
    # How 'bout something with a different cert type and name?

    label "punchin-point"
    local_id_type mbox
    local_id "ipsec-wizard@domain.org"

    remote_id "10.5.5.128"

    local_addr 0.0.0.0/0
    remote_addr 10.5.5.128

    p1_xform
    { auth_method rsa_sig oakley_group 5 auth_alg md5 encr_alg \
      blowfish }
}

{
    label "receiver side"

    remote_id "ipsec-wizard@domain.org"

    local_id_type ip
    local_id "10.5.5.128"

    local_addr 10.5.5.128
    remote_addr 0.0.0.0/0

    p1_xform
    { auth_method rsa_sig oakley_group 5 auth_alg md5 encr_alg blowfish }
    # NOTE: Specifying preshared null-and-voids the remote_id/local_id
    #        fields.
    p1_xform
    { auth_method preshared oakley_group 5 auth_alg md5 encr_alg \
      blowfish}
}

{
    # Only allow AES with 256-bit keys in Phase 1

```

EXAMPLE 1 A Sample ike.config File (Continued)

```

label "keylen-restricted"

local_id_type ip
local_id "10.1.1.3"
local_addr 10.1.1.3
remote_addr 10.1.1.4

p1_xform
{auth_method rsa_sig oakley_group 5 auth_alg md5 encr_alg \
 aes(256)}
}

```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	Committed

See Also [cryptoadm\(1M\)](#), [ikeadm\(1M\)](#), [in.iked\(1M\)](#), [ikecert\(1M\)](#), [ipseckey\(1M\)](#), [ipsecalgs\(1M\)](#), [ipseconf\(1M\)](#), [svccfg\(1M\)](#), [dlopen\(3C\)](#), [attributes\(5\)](#), [labels\(5\)](#), [random\(7D\)](#)

Harkins, Dan and Carrel, Dave. *RFC 2409, Internet Key Exchange (IKE)*. Cisco Systems, November 1998.

Maughan, Douglas et. al. *RFC 2408, Internet Security Association and Key Management Protocol (ISAKMP)*. National Security Agency, Ft. Meade, MD. November 1998.

Piper, Derrell. *RFC 2407, The Internet IP Security Domain of Interpretation for ISAKMP*. Network Alchemy. Santa Cruz, California. November 1998.

Kivinen, T. *RFC 3526, More Modular Exponential (MODP) Diffie-Hellman Groups for Internet Key Exchange (IKE)*. The Internet Society, Network Working Group. May 2003.

Name ike.preshared – pre-shared keys file for IKE

Synopsis /etc/inet/secret/ike.preshared

Description The /etc/inet/secret/ike.preshared file contains secret keying material that two IKE instances can use to authenticate each other. Because of the sensitive nature of this data, it is kept in the /etc/inet/secret directory, which is only accessible by root.

Pre-shared keys are delimited by open-curly-brace ({) and close-curly-brace (}) characters. There are five name-value pairs required inside a pre-shared key:

Name	Value	Example
localidtype	IP	localidtype IP
remoteidtype	IP	remoteidtype IP
localid	IP-address	localid 10.1.1.2
	Subnet/Prefix	localid 10.1.1.0/24
remoteid	IP-address	remoteid 10.1.1.3
	Subnet/Prefix	remoteid 10.1.1.0/24
key	hex-string	1234567890abcdef
	hex-string	1234567890abcdef
	hex-string	0x1234567890abcdef
	ASCII-string	"This is my preshared key"

Comment lines with # appearing in the first column are also legal.

An ASCII-string can consist of any valid ASCII character except for NEWLINE. A backslash (\) is considered an escape character when it precedes a double quote or itself. Otherwise a backslash is taken literally.

Files in this format can also be used by the `ikeadm(1M)` command to load additional pre-shared keys into a running an `in.iked(1M)` process.

Examples EXAMPLE 1 A Sample ike.preshared File

The following is an example of an ike.preshared file:

```
#
# Two pre-shared keys between myself, 10.1.1.2, and two remote
# hosts. Note that names are not allowed for IP addresses.
#
# A decent hex string can be obtained by performing:
```

EXAMPLE 1 A Sample `ike.preshared` File (Continued)

```
#          od -x </dev/random | head
#
{
  localidtype IP
  localid 10.1.1.2
  remoteidtype IP
  remoteid 10.21.12.4
  key 4b656265207761732068657265210c0a
}
{
  localidtype IP
  localid 10.1.1.2
  remoteidtype IP
  remoteid 10.21.13.0/24
  key "string of my ch01c3"
}
{
  localidtype IP
  localid 10.1.1.2
  remoteidtype IP
  remoteid 10.9.1.25
  key 536f20776572652042696c6c2c2052656e65652c20616e642043687269732e0a
}
}
```

Security If this file is compromised, all IPsec security associations derived from secrets in this file will be compromised as well. The default permissions on `ike.preshared` are `0600`. They should stay this way.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os

See Also [od\(1\)](#), [ikeadm\(1M\)](#), [in.iked\(1M\)](#), [ipseckey\(1M\)](#), [attributes\(5\)](#), [random\(7D\)](#)

Name inetd.conf – Internet servers database

Synopsis /etc/inet/inetd.conf

/etc/inetd.conf

Description In the current release of the Solaris operating system, the `inetd.conf` file is no longer directly used to configure `inetd`. The Solaris services which were formerly configured using this file are now configured in the Service Management Facility (see [smf\(5\)](#)) using [inetadm\(1M\)](#). Any records remaining in this file after installation or upgrade, or later created by installing additional software, must be converted to [smf\(5\)](#) services and imported into the SMF repository using [inetconv\(1M\)](#), otherwise the service will not be available.

For Solaris operating system releases prior to the current release (such as Solaris 9), the `inetd.conf` file contains the list of servers that [inetd\(1M\)](#) invokes when it receives an Internet request over a socket. Each server entry is composed of a single line of the form:

```
service-name endpoint-type protocol wait-status uid server-program \
server-arguments
```

Fields are separated by either SPACE or TAB characters. A '#' (number sign) indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines that search this file.

service-name The name of a valid service listed in the `services` file. For RPC services, the value of the *service-name* field consists of the RPC service name or program number, followed by a '/' (slash) and either a version number or a range of version numbers, for example, `rstatd/2-4`.

endpoint-type Can be one of:

<code>stream</code>	for a stream socket
<code>dgram</code>	for a datagram socket
<code>raw</code>	for a raw socket
<code>seqpacket</code>	for a sequenced packet socket
<code>tli</code>	for all TLI endpoints

protocol A recognized protocol listed in the file `/etc/inet/protocols`. For servers capable of supporting TCP and UDP over IPv6, the following protocol types are also recognized:

```
tcp6
udp6
```

`tcp6` and `udp6` are not official protocols; accordingly, they are not listed in the `/etc/inet/protocols` file.

Here the `inetd` program uses an `AF_INET6` type socket endpoint. These servers can also handle incoming IPv4 client requests in addition to IPv6 client requests.

For RPC services, the field consists of the string `rpc` followed by a `'/` (slash) and either a `'*` (asterisk), one or more nettypes, one or more netids, or a combination of nettypes and netids. Whatever the value, it is first treated as a nettype. If it is not a valid nettype, then it is treated as a netid. For example, `rpc/*` for an RPC service using all the transports supported by the system (the list can be found in the `/etc/netconfig` file), equivalent to saying `rpc/visible rpc/ticots` for an RPC service using the Connection-Oriented Transport Service.

<i>wait-status</i>	This field has values <code>wait</code> or <code>nowait</code> . This entry specifies whether the server that is invoked by <code>inetd</code> will take over the listening socket associated with the service, and whether once launched, <code>inetd</code> will <code>wait</code> for that server to exit, if ever, before it resumes listening for new service requests. The <i>wait-status</i> for datagram servers must be set to <code>wait</code> , as they are always invoked with the original datagram socket that will participate in delivering the service bound to the specified service. They do not have separate listening and accepting sockets. Accordingly, do not configure UDP services as <code>nowait</code> . This causes a race condition by which the <code>inetd</code> program selects on the socket and the server program reads from the socket. Many server programs will be forked, and performance will be severely compromised. Connection-oriented services such as TCP stream services can be designed to be either <code>wait</code> or <code>nowait</code> status.
<i>uid</i>	The user ID under which the server should run. This allows servers to run with access privileges other than those for root.
<i>server-program</i>	Either the pathname of a server program to be invoked by <code>inetd</code> to perform the requested service, or the value <code>internal</code> if <code>inetd</code> itself provides the service.
<i>server-arguments</i>	If a server must be invoked with command line arguments, the entire command line (including argument 0) must appear in this field (which consists of all remaining words in the entry). If the server expects <code>inetd</code> to pass it the address of its peer, for compatibility with 4.2BSD executable daemons, then the first argument to the command should be specified as <code>%A</code> . No more than 20 arguments are allowed in this field. The <code>%A</code> argument is implemented only for services whose <i>wait-status</i> value is <code>nowait</code> .

Files /etc/netconfig network configuration file
 /etc/inet/protocols Internet protocols
 /etc/inet/services Internet network services

See Also [rlogin\(1\)](#), [rsh\(1\)](#), [in.tftpd\(1M\)](#), [inetadm\(1M\)](#), [inetconv\(1M\)](#), [inetd\(1M\)](#), [services\(4\)](#), [smf\(5\)](#)

Notes /etc/inet/inetd.conf is the official SVR4 name of the inetd.conf file. The symbolic link /etc/inetd.conf exists for BSD compatibility.

This man page describes inetd.conf as it was supported in Solaris operating system releases prior to the current release. The services that were configured by means of inetd.conf are now configured in the Service Management Facility (see [smf\(5\)](#)) using [inetadm\(1M\)](#).

Name inet_type – default Internet protocol type

Synopsis /etc/default/inet_type

Description The `inet_type` file defines the default IP protocol to use. Currently this file is only used by the `ifconfig(1M)` and `netstat(1M)` commands.

The `inet_type` file can contain a number of `<variable>=<value>` lines. Currently, the only variable defined is `DEFAULT_IP`, which can be assigned a value of `IP_VERSION4`, `IP_VERSION6`, or `BOTH`.

The output displayed by the `ifconfig` and `netstat` commands can be controlled by the value of `DEFAULT_IP` set in `inet_type` file. By default, both commands display the IPv4 and IPv6 information available on the system. The user can choose to suppress display of IPv6 information by setting the value of `DEFAULT_IP`. The following shows the possible values for `DEFAULT_IP` and the resulting `ifconfig` and `netstat` output that will be displayed:

<code>IP_VERSION4</code>	Displays only IPv4 related information. The output displayed is backward compatible with older versions of the <code>ifconfig(1M)</code> and <code>netstat(1M)</code> commands.
<code>IP_VERSION6</code>	Displays both IPv4 and IPv6 related information for <code>ifconfig</code> and <code>netstat</code> .
<code>BOTH</code>	Displays both IPv4 and IPv6 related information for <code>ifconfig</code> and <code>netstat</code> .

The command-line options to the `ifconfig` and `netstat` commands override the effect of `DEFAULT_IP` as set in the `inet_type` file. For example, even if the value of `DEFAULT_IP` is `IP_VERSION4`, the command

```
example% ifconfig -a6
```

will display all IPv6 interfaces.

Examples EXAMPLE 1 Suppressing IPv6 Related Output

This is what the `inet_type` file must contain if you want to suppress IPv6 related output:

```
DEFAULT_IP=IP_VERSION4
```

See Also `ifconfig(1M)`, `netstat(1M)`

Name `init.d` – initialization and termination scripts for changing init states

Synopsis `/etc/init.d`

Description `/etc/init.d` is a directory containing initialization and termination scripts for changing init states. These scripts are linked when appropriate to files in the `rc?.d` directories, where ‘?’ is a single character corresponding to the init state. See [init\(1M\)](#) for definitions of the states.

The service management facility (see [smf\(5\)](#)) is the preferred mechanism for service initiation and termination. The `init.d` and `rc?.d` directories are obsolete, and are provided for compatibility purposes only. Applications launched from these directories by [svc.startd\(1M\)](#) are incomplete services, and are not restarted on failure.

File names in `rc?.d` directories are of the form `[SK]nn<init.d filename>`, where *S* means start this job, *K* means kill this job, and *nn* is the relative sequence number for killing or starting the job.

When entering a state (init *S*,0,2,3,etc.) the `rc[S0-6]` script executes those scripts in `/etc/rc[S0-6].d` that are prefixed with *K* followed by those scripts prefixed with *S*. When executing each script in one of the `/etc/rc[S0-6]` directories, the `/usr/sbin/rc[S0-6]` script passes a single argument. It passes the argument ‘stop’ for scripts prefixed with *K* and the argument ‘start’ for scripts prefixed with *S*. There is no harm in applying the same sequence number to multiple scripts. In this case the order of execution is deterministic but unspecified.

Guidelines for selecting sequence numbers are provided in README files located in the directory associated with that target state. For example, `/etc/rc[S0-6].d/README`. Absence of a README file indicates that there are currently no established guidelines.

Do not put `/etc/init.d` in your `$PATH`. Having this directory in your `$PATH` can cause unexpected behavior. The programs in `/etc/init.d` are associated with init state changes and, under normal circumstances, are not intended to be invoked from a command line.

Examples **EXAMPLE 1** Example of `/usr/sbin/rc2`.

When changing to init state 2 (multi-user mode, network resources not exported), `/sbin/rc2` is initiated by the [svc.startd\(1M\)](#) process. The following steps are performed by `/usr/sbin/rc2`.

1. In the directory `/etc/rc2.d` are files used to stop processes that should not be running in state 2. The filenames are prefixed with *K*. Each *K* file in the directory is executed (by `/usr/sbin/rc2`) in alphanumeric order when the system enters init state 2. See example below.
2. Also in the `rc2.d` directory are files used to start processes that should be running in state 2. As in Step 1, each *S* file is executed.

Assume the file `/etc/init.d/netdaemon` is a script that initiates networking daemons when given the argument ‘start’, and terminates the daemons if given the argument ‘stop’. It is linked

EXAMPLE 1 Example of `/usr/sbin/rc2`. (Continued)

to `/etc/rc2.d/S68netdaemon`, and to `/etc/rc0.d/K67netdaemon`. The file is executed by `/etc/rc2.d/S68netdaemon start` when init state 2 is entered and by `/etc/rc0.d/K67netdaemon stop` when shutting the system down.

See Also [svcs\(1\)](#), [init\(1M\)](#), [svc.startd\(1M\)](#), [svccfg\(1M\)](#), [smf\(5\)](#)

Notes Solaris now provides an expanded mechanism, which includes automated restart, for applications historically started using the init script mechanism. The Service Management Facility (introduced in [smf\(5\)](#)) is the preferred delivery mechanism for persistently running applications. Existing `init.d` scripts continue to be executed according to the rules in this manual page. The details of execution in relation to managed services are available in [svc.startd\(1M\)](#).

On earlier Solaris releases, a script named with a suffix of `.sh` would be sourced, allowing scripts to modify the environment of other scripts executed later. This behavior is no longer supported; for altering the environment in which services are run, see the `setenv` subcommand in [svccfg\(1M\)](#).

`/usr/sbin/rc2` has references to the obsolescent `rc.d` directory. These references are for compatibility with old `INSTALL` scripts. New `INSTALL` scripts should use the `init.d` directory for related executables. The same is true for the `shutdown.d` directory.

Name inittab – script for init

Description The `/etc/inittab` file controls process dispatching by `init`. The processes most typically dispatched by `init` are daemons.

It is no longer necessary to edit the `/etc/inittab` file directly. Administrators should use the Solaris Service Management Facility (SMF) to define services instead. Refer to [smf\(5\)](#) and the *Oracle Solaris Administration: Common Tasks* for more information on SMF.

To modify parameters passed to `ttymon(1M)`, use `svccfg(1M)` to modify the SMF repository. See [ttymon\(1M\)](#) for details on the available SMF properties.

The `inittab` file is composed of entries that are position dependent and have the following format:

id:*rstate*:*action*:*process*

Each entry is delimited by a newline; however, a backslash (\) preceding a newline indicates a continuation of the entry. Up to 512 characters for each entry are permitted. Comments may be inserted in the *process* field using the convention for comments described in [sh\(1\)](#). There are no limits (other than maximum entry size) imposed on the number of entries in the `inittab` file. The entry fields are:

id One to four characters used to uniquely identify an entry. Do not use the characters “r” or “t” as the first or only character in this field. These characters are reserved for the use of [rlogin\(1\)](#) and [telnet\(1\)](#).

rstate Define the run level in which this entry is to be processed. Run-levels effectively correspond to a configuration of processes in the system. That is, each process spawned by `init` is assigned a run level(s) in which it is allowed to exist. The run levels are represented by a number ranging from 0 through 6. For example, if the system is in run level 1, only those entries having a 1 in the *rstate* field are processed.

When `init` is requested to change run levels, all processes that do not have an entry in the *rstate* field for the target run level are sent the warning signal SIGTERM and allowed a 5-second grace period before being forcibly terminated by the kill signal SIGKILL. The *rstate* field can define multiple run levels for a process by selecting more than one run level in any combination from 0 through 6. If no run level is specified, then the process is assumed to be valid at all run levels 0 through 6.

There are three other values, a, b and c, which can appear in the *rstate* field, even though they are not true run levels. Entries which have these characters in the *rstate* field are processed only when an `init` or `telinit` process requests them to be run (regardless of the current run level of the system). See [init\(1M\)](#). These

differ from run levels in that `init` can never enter run level `a`, `b` or `c`. Also, a request for the execution of any of these processes does not change the current run level. Furthermore, a process started by an `a`, `b` or `c` command is not killed when `init` changes levels. They are killed only if their line in `inittab` is marked `off` in the *action* field, their line is deleted entirely from `inittab`, or `init` goes into single-user state.

action Key words in this field tell `init` how to treat the process specified in the *process* field. The actions recognized by `init` are as follows:

- | | |
|------------------------|---|
| <code>respawn</code> | If the process does not exist, then start the process; do not wait for its termination (continue scanning the <code>inittab</code> file), and when the process dies, restart the process. If the process currently exists, do nothing and continue scanning the <code>inittab</code> file. |
| <code>wait</code> | When <code>init</code> enters the run level that matches the entry's <i>rstate</i> , start the process and wait for its termination. All subsequent reads of the <code>inittab</code> file while <code>init</code> is in the same run level cause <code>init</code> to ignore this entry. |
| <code>once</code> | When <code>init</code> enters a run level that matches the entry's <i>rstate</i> , start the process, do not wait for its termination. When it dies, do not restart the process. If <code>init</code> enters a new run level and the process is still running from a previous run level change, the program is not restarted. |
| <code>boot</code> | The entry is to be processed only at <code>init</code> 's boot-time read of the <code>inittab</code> file. <code>init</code> is to start the process and not wait for its termination; when it dies, it does not restart the process. In order for this instruction to be meaningful, the <i>rstate</i> should be the default or it must match <code>init</code> 's run level at boot time. This action is useful for an initialization function following a hardware reboot of the system. |
| <code>bootwait</code> | The entry is to be processed the first time <code>init</code> goes from single-user to multi-user state after the system is booted. <code>init</code> starts the process, waits for its termination and, when it dies, does not restart the process. |
| <code>powerfail</code> | Execute the process associated with this entry only when <code>init</code> receives a power fail signal, <code>SIGPWR</code> (see signal(3C)). |
| <code>powerwait</code> | Execute the process associated with this entry only when <code>init</code> receives a power fail signal, <code>SIGPWR</code> , and wait until it terminates before continuing any processing of <code>inittab</code> . |

<code>off</code>	If the process associated with this entry is currently running, send the warning signal <code>SIGTERM</code> and wait 5 seconds before forcibly terminating the process with the kill signal <code>SIGKILL</code> . If the process is nonexistent, ignore the entry.
<code>ondemand</code>	This instruction is really a synonym for the <code>respawn</code> action. It is functionally identical to <code>respawn</code> but is given a different keyword in order to divorce its association with run levels. This instruction is used only with the <code>a</code> , <code>b</code> or <code>c</code> values described in the <code>rstate</code> field.
<code>sysinit</code>	Entries of this type are executed before <code>init</code> tries to access the console (that is, before the <code>Console Login:</code> prompt). It is expected that this entry will be used only to initialize devices that <code>init</code> might try to ask the run level question. These entries are executed and <code>init</code> waits for their completion before continuing.
<code>process</code>	Specify a command to be executed. The entire process field is prefixed with <code>exec</code> and passed to a forked <code>sh</code> as <code>sh -c 'exec command'</code> . For this reason, any legal <code>sh</code> syntax can appear in the <code>process</code> field.

See Also [sh\(1\)](#), [who\(1\)](#), [init\(1M\)](#), [svcadm\(1M\)](#), [svc.startd\(1M\)](#), [ttymon\(1M\)](#), [exec\(2\)](#), [open\(2\)](#), [signal\(3C\)](#), [smf\(5\)](#)

Oracle Solaris Administration: Common Tasks

Notes With the introduction of the service management facility, the system-provided `/etc/inittab` file is greatly reduced from previous releases.

The `initdefault` entry is not recognized in Solaris 10. See [smf\(5\)](#) for information on SMF milestones, and [svcadm\(1M\)](#), which describes the “`svcadm milestone -d`” command; this provides similar functionality to modifying the `initdefault` entry in previous versions of the Solaris OS.

Name ipaddrsel.conf – IPv6 default address selection policy

Synopsis /etc/inet/ipaddrsel.conf

Description The `ipaddrsel.conf` file contains the IPv6 default address selection policy table used for IPv6 source address selection and the sorting of `AF_INET6` addresses returned from name to address resolution. The mechanism for loading the file, the file format, and the meaning of the contents are described in [ipaddrsel\(1M\)](#).

Examples EXAMPLE 1 Default/etc/inet/ipaddrsel.conf File

The following is the default `/etc/inet/ipaddrsel.conf` file:

```
#
#ident      "@(#)ipvdas.conf    1.1    02/07/28 SMI"
#
# Copyright 2002 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
# Prefix                Precedence Label
::1/128                 50      0
::/0                    40      1
2002::/16               30      2
::/96                   20      3
::ffff:0.0.0.0/96      10      4
```

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTETYPE	ATTRIBUTEVALUE
Availability	system/core-os
Interface Stability	Committed

See Also [ipaddrsel\(1M\)](#), [attributes\(5\)](#)

Name ipf, ipf.conf – IP packet filter rule syntax

Description The `ipf` or `ipf.conf` configuration files are associated with the Solaris IP Filter feature. See [ipfilter\(5\)](#).

A rule file for `ipf(1M)` can have any name or can be `stdin`. You can use `ipfstat(1M)` output as input to `ipf(1M)`. `ipfstat` outputs parseable rules, suitable for input to `ipf`, when displaying the internal kernel filter lists. Thus, for example, to remove all filters on input packets, you can enter:

```
# ipfstat -i | ipf -rf -
```

Grammar The IP filter feature uses the grammar shown below to construct filtering rules. The syntax is simplified for readability. Note that some combinations that match this grammar are disallowed by the software because they do not make sense (for example, `tcp` flags for non-TCP packets).

```
filter-rule = [ insert ] action in-out [ options ] [ tos ] [ ttl ]
             [ proto ] ip [ group ].

insert      = "@" decnumber .
action      = block | "pass" | log | "count" | skip | auth .
in-out      = "in" | "out" .
options     = [ log ] [ tag ] [ "quick" ] [ "on" interface-name
             [ dup ] [ frouete ] [ replyto ]
tos         = "tos" decnumber | "tos" hexnumber .
ttl         = "ttl" decnumber .
proto       = "proto" protocol .
ip          = srcdst [ flags ] [ icmp ] [ with withopt ] [ keep ] .
group       = [ "head" decnumber ] [ "group" decnumber ] .

block       = "block" [ return-icmp[return-code] | "return-rst" ] .
auth        = "auth"
log         = "log" [ "body" ] [ "first" ] [ "or-block" ] [ "level" loglevel ] .
tag         = "set-tag" tagid
skip        = "skip" decnumber .
dup         = "dup-to" interface-name[":ipaddr"] .
frouete     = "fastroute" | "to" interface-name [ ":" ipaddr ].
replyto     = "reply-to" interface-name [ ":" ipaddr ].
protocol    = "tcp/udp" | "udp" | "tcp" | "icmp" | decnumber .
srcdst      = "all" | fromto .
fromto      = "from" [ "!" ] object "to" [ "!" ] object .

return-icmp = "return-icmp" | "return-icmp-as-dest" .
object      = addr [ port-comp | port-range ] .
addr        = "any" | nummask | host-name [ "mask" ipaddr | "mask" hexnumber ] .
addr        = "any" | "<thishost>" | nummask |
             host-name [ "mask" ipaddr | "mask" hexnumber ] .
port-comp   = "port" compare port-num .
```

```

port-range = "port" port-num range port-num .
flags      = "flags" flag { flag } [ "/" flag { flag } ] .
with       = "with" | "and" .
icmp       = "icmp-type" icmp-type [ "code" decnumber ] .
return-code = ("icmp-code") .
keep       = "keep" "state" | "keep" "frags" | "keep" "state"
           "keep" "frags" | "keep" "frags" | "keep" "state".
loglevel   = facility"."priority | priority .
nummask    = host-name [ "/" decnumber ] .
host-name  = ipaddr | hostname | "any" .
ipaddr     = host-num "." host-num "." host-num "." host-num | ipv6addr .
host-num   = digit [ digit [ digit ] ] .
port-num   = service-name | decnumber .

withopt    = [ "not" | "no" ] opttype [ withopt ] v6hdrs [ ipv6hdr ] .
opttype    = "ipopts" | "short" | "frag" | "frag-body" | "frags" |
           "opt" optname | "nat" | "multicast" |
           "bcast" | "mcast" | "state" | "bad-nat" | "bad" | "oow" |
           "lowttl" | "bad-src" optname .
optname    = ipopts [ "," optname ] .
ipopts    = optlist | "sec-class" [ secname ] .
ipv6hdr    = "ah" | "esp" | "dstopts" | "hopopts" | "ipv6" | "none" |
           "routing" | "frag"
secname    = seclvl [ "," secname ] .
seclvl    = "unclass" | "confid" | "reserv-1" | "reserv-2" | "reserv-3" |
           "reserv-4" | "secret" | "topsecret" .
icmp-type  = "unreach" | "echo" | "echorep" | "squelch" | "redir" |
           "timex" | "paramprob" | "timest" | "timestrep" | "inforeq" |
           "inforep" | "maskreq" | "maskrep" | decnumber .
icmp-code  = decumber | "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" .
optlist    = "nop" | "rr" | "zsu" | "mtup" | "mtur" | "encode" | "ts" |
           "tr" | "sec" | "lsrr" | "e-sec" | "cipso" | "satid" | "ssrr" |
           "addext" | "visa" | "imitd" | "eip" | "finn" .
facility    = "kern" | "user" | "mail" | "daemon" | "auth" | "syslog" |
           "lpr" | "news" | "uucp" | "cron" | "ftp" | "authpriv" |
           "audit" | "logalert" | "local0" | "local1" | "local2" |
           "local3" | "local4" | "local5" | "local6" | "local7" .
priority   = "emerg" | "alert" | "crit" | "err" | "warn" | "notice" |
           "info" | "debug" .

hexnumber  = "0" "x" hexstring .
hexstring  = hexdigit [ hexstring ] .
decnumber  = digit [ decnumber ] .

compare    = "=" | "!=" | "<" | ">" | "<=" | ">=" | "eq" | "ne" | "lt" |

```

```

    "gt" | "le" | "ge" .
range      = "<" | ">" .
hexdigit = digit | "a" | "b" | "c" | "d" | "e" | "f" .
digit     = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" .
flag     = "F" | "S" | "R" | "P" | "A" | "U" .

```

Filter Rules Filter rules are checked in order, with the last matching rule determining the treatment of the packet. An exception to this is the quick option, which is discussed below.

By default, filters are installed at the end of the kernel's filter lists. Prepending a rule with `@<num>` causes it to be inserted as the `<num>`th entry in the current list. This is especially useful when modifying and testing active filter rule sets. See [ipf\(1M\)](#) for more information.

The simplest valid rules are:

```

block in all
pass in all
log out all
count in all

```

These rules do not have an effect on filtering, but are listed here to illustrate the grammar.

Actions Each rule *must* have an action. The action indicates what to do with the packet if it matches the filter rule. The IP filter feature recognizes the following actions:

block Indicates that a packet should be flagged to be dropped. In response to blocking a packet, the filter can be instructed to send a reply packet, either an ICMP packet (`return-icmp`), an ICMP packet that fakes being from the original packet's destination (`return-icmp-as-dest`), or a TCP reset (`return-rst`). An ICMP packet can be generated in response to any IP packet and its type can optionally be specified, but a TCP reset can only be used with a rule that is being applied to TCP packets. When using `return-icmp` or `return-icmp-as-dest`, it is possible to specify the actual unreachable type. That is, whether it is a network unreachable, port unreachable, or even administratively prohibited. You do this by enclosing the ICMP code associated with the action in parentheses directly following `return-icmp` or `return-icmp-as-dest`. For example:

```
block return-icmp(11) ...
```

The preceding entry causes a return of a Type-Of-Service (TOS) ICMP unreachable error.

pass Flag the packet to be let through the filter without any action being taken.

log Causes the packet to be logged (as described in the LOGGING section, below) and has no effect on whether the packet will be allowed through the filter.

count	Causes the packet to be included in the accounting statistics kept by the filter and has no effect on whether the packet will be allowed through the filter. These statistics are viewable with <code>ipfstat(1M)</code> .
skip <num>	Causes the filter to skip over the next <num> filter rules. If a rule is inserted or deleted inside the region being skipped over, then the value of <num> is adjusted appropriately.
auth	Allows authentication to be performed by a user-space program running and waiting for packet information to validate. The packet is held for a period of time in an internal buffer while it waits for the program to return to the kernel the “real” flags for whether it should be allowed through. Such a program might look at the source address and request some sort of authentication from the user (such as a password) before allowing the packet through or telling the kernel to drop it if the packet is from an unrecognized source.

The word following the action keyword must be either `in` or `out`. Each packet moving through the kernel is either inbound or outbound. “Inbound” means that a packet has just been received on an interface and is moving towards the kernel’s protocol processing. “Outbound” means that a packet has been transmitted or forwarded by the stack and is on its way to an interface. There is a requirement that each filter rule explicitly state on which side of the I/O it is to be used.

Options The currently supported options are listed below. Where you use options, you must use them in the order shown here.

log If this is the last matching rule, the packet header is written to the `ipf log`, as described in the LOGGING section below.

quick Allows short-cut rules to speed up the filter or override later rules. If a packet matches a filter rule that is marked as quick, this rule will be the last rule checked, allowing a “short-circuit” path to avoid processing later rules for this packet. The current status of the packet (after any effects of the current rule) determine whether it is passed or blocked.

If the `quick` option is missing, the rule is taken to be a “fall-through” rule, meaning that the result of the match (block or pass) is saved and that processing will continue to see if there are any more matches.

on Allows an interface name to be incorporated into the matching procedure. Interface names are as displayed by `netstat -i`. If this option is used, the rule matches only if the packet is going through that interface in the specified direction (`in` or `out`). If this option is absent, the rule is applied to a packet regardless of the interface it is present on (that is, on all interfaces). Filter rule sets are common to all interfaces, rather than having a filter list for each interface.

This option is especially useful for simple IP-spoofing protection: packets should be allowed to pass inbound only on the interface from which the specified source address would be expected. Others can be logged, or logged and dropped.

- dup - to Causes the packet to be copied, with the duplicate packet sent outbound on a specified interface, optionally with the destination IP address changed to that specified. This is useful for off-host logging, using a network sniffer.
- to Causes the packet to be moved to the outbound queue on the specified interface. This can be used to circumvent kernel routing decisions, and, if applied to an inbound rule, even to bypass the rest of the kernel processing of the packet. It is thus possible to construct a firewall that behaves transparently, like a filtering hub or switch, rather than a router. The `fast route` keyword is a synonym for this option.

Matching Parameters The keywords described in this section are used to describe attributes of the packet to be used when determining whether rules do or do not match. The following general-purpose attributes are provided for matching and must be used in the order shown below.

- tos Packets with different Type-Of-Service values can be filtered. Individual service levels or combinations can be filtered upon. The value for the TOS mask can be represented either as a hexadecimal or decimal integer.
- ttl Packets can also be selected by their Time-To-Live value. The value given in the filter rule must exactly match that in the packet for a match to occur. This value can be given only as a decimal integer.
- proto Allows a specific protocol to be matched against. All protocol names found in `/etc/protocols` are recognized and can be used. However, the protocol can also be given as a decimal number, allowing for rules to match your own protocols and for new protocols.

The special protocol keyword `tcp/udp` can be used to match either a TCP or a UDP packet and has been added as a convenience to save duplication of otherwise-identical rules.

IP addresses can be specified in one of two ways: as a numerical address/mask, or as a hostname mask/netmask. The hostname can be either of the dotted numeric form or a valid hostname, from the `hosts` file or DNS (depending on your configuration and library). There is no special designation for networks, but network names are recognized. Note that having your filter rules depend on DNS results can introduce an avenue of attack and is discouraged.

There is a special case for the hostname `any`, which is taken to be `0.0.0.0/0` (mask syntax is discussed below) and matches all IP addresses. Only the presence of `any` has an implied mask. In all other situations, a hostname *must* be accompanied by a mask. It is possible to give any a hostmask, but in the context of this language, it would accomplish nothing.

The numerical format x/y indicates that a mask of y consecutive 1 bits set is generated, starting with the MSB, so that a y value of 16 would result in `0xffff0000`. The symbolic x mask y indicates that the mask y is in dotted IP notation or a hexadecimal number of the form `0x12345678`. Note that all the bits of the IP address indicated by the bitmask must match the address on the packet exactly; there is currently not a way to invert the sense of the match or to match ranges of IP addresses that do not express themselves easily as bitmasks.

If a port match is included, for either or both of source and destination, then it is only applied to TCP and UDP packets. This is equivalent to `proto tcp/udp`. When composing port comparisons, either the service name or an integer port number can be used. Port comparisons can be done in a number of forms, with a number of comparison operators, or you can specify port ranges. When the port appears as part of the `from` object, it matches the source port number. When it appears as part of the `to` object, it matches the destination port number. See EXAMPLES.

The `all` keyword is essentially a synonym for “from any to any” with no other match parameters.

Following the source and destination matching parameters, you can use the following additional parameters:

with Used to match irregular attributes that some packets might have associated with them. To match the presence of IP options in general, use `with ipopts`. To match packets that are too short to contain a complete header, use `with short`. To match fragmented packets, use `with frag`. For more specific filtering on IP options, you can list individual options.

Before any parameter used after the `with` keyword, you can insert the word `not` or `no` to cause the filter rule to match only if the option(s) is not present.

Multiple consecutive `with` clauses are allowed. Alternatively, you can use the `keyword` and `in place` of `with`. This alternative is provided to make the rules more readable (“`with ... and ...`”). When multiple clauses are listed, all clauses must match to cause a match of the rule.

flags Effective only for TCP filtering. Each of the letters possible represents one of the possible flags that can be set in the TCP header. The association is as follows:

F - FIN
S - SYN
R - RST
P - PUSH
A - ACK
U - URG

The various flag symbols can be used in combination, so that SA matches a SYN-ACK combination in a packet. There is nothing preventing the specification of combinations, such as SFR, that would not normally be generated by fully conformant TCP implementations. However, to guard against unpredictable behavior, it is necessary to state which flags you are filtering against. To allow this, it is possible to set a mask indicating against which TCP flags you wish to compare (that is, those you deem significant). This is done by appending */<flags>* to the set of TCP flags you wish to match against, for example:

```
... flags S      Becomes flags S/AUPRFS and matches packets with only
                  the SYN flag set.

... flags SA     Becomes flags SA/AUPRFS and matches any packet with
                  only the SYN and ACK flags set.

... flags S/SA   Matches any packet with just the SYN flag set out of the
                  SYN-ACK pair, which is the common establish
                  keyword action. S/SA will not match a packet with both
                  SYN and ACK set, but will match SFP.
```

`icmp-type` Effective only when used with `proto icmp` and must *not* be used in conjunction with `flags`. There are a number of types, which can be referred to by an abbreviation recognized by this language or by the numbers with which they are associated. The most important type from a security point of view is the ICMP redirect.

`Keep History` The penultimate parameter that can be set for a filter rule is whether or not to record historical information for a packet, and what sort to keep. The following information can be kept:

`state` Keeps information about the flow of a communication session. State can be kept for TCP, UDP, and ICMP packets.

`frags` Keeps information on fragmented packets, to be applied to later fragments.

Presence of these parameters allows matching packets to flow straight through, rather than going through the access control list.

`Groups` The last pair of parameters control filter rule grouping. By default, all filter rules are placed in group 0 if no other group is specified. To add a rule to a non-default group, the group must first be started by creating a group *head*. If a packet matches a rule which is the head of a group, the filter processing then switches to the group, using that rule as the default for the group. If `quick` is used with a head rule, rule processing is not stopped until it has returned from processing the group.

A rule can be both the head for a new group and a member of a non-default group (head and group can be used together in a rule).

- `head <n>` Indicates that a new group (number <n>) should be created.
- `group <n>` Indicates that the rule should be put in group (number <n>) rather than group 0.

Logging When a packet is logged, by means of either the `log` action or `log` option, the headers of the packet are written to the `ipl` packet logging pseudo-device. Immediately following the `log` keyword, you can use the following qualifiers in the order listed below:

- `body` Indicates that the first 128 bytes of the packet contents will be logged after the headers.
- `first` If `log` is being used in conjunction with a `keep` option, it is recommended that you also apply this option so that only the triggering packet is logged and not every packet which thereafter matches state information.
- `or-block` Indicates that, if for some reason, the filter is unable to log the packet (such as the log reader being too slow), then the rule should be interpreted as if the action was `block` for this packet.
- `level loglevel` Indicates what logging facility and priority (or, if the default facility is used, priority only) will be used to log information about this packet using `ipmon`'s `-s` option.

You can use `ipmon(1M)` to read and format the log.

Loopback Filtering (Inter-Zone) By default, the IP Filter feature will *not* filter or intercept any packets that are local to the machine. This includes traffic to or from the loopback addresses (127.0.0.1, and so forth), traffic between sockets that are on the same host (for example, from `eri0` to `eri1`), and traffic between zones.

To enable loopback or zone filtering, you must add the following line to `ipf.conf` file:

```
set intercept_loopback true;
```

This line must be placed before any `block` or `pass` rules in this file or, put another way, it must be the first non-comment line in `ipf.conf`.

When you enable filtering of packets in any one of the scenarios described above, you enabling filtering for all them. That is, when you enable the IP Filter feature to intercept packets between zones, you also cause it to receive packets that are involved in loopback traffic.

Examples **EXAMPLE 1** Using the `quick` Option

The `quick` option works well for rules such as:

```
block in quick from any to any with ipopts
```

This rule matches any packet with a non-standard header length (IP options present) and aborts further processing of later rules, recording a match and also indicating that the packet

EXAMPLE 1 Using the quick Option (Continued)

should be blocked.

EXAMPLE 2 Using the Fall-through Nature of Rule Parsing

The “fall-through” rule parsing allows for effects such as the following:

```
block in from any to any port < 6000
pass in from any to any port >= 6000
block in from any to any port > 6003
```

These rules set up the range 6000-6003 as being permitted and all others being denied. Note that the effect of the first rule is overridden by subsequent rules. Another (easier) way to do the same is:

```
block in from any to any port 6000 <> 6003
pass in from any to any port 5999 >> 6004
```

Note that both the `block` and `pass` are needed here to effect a result, because a failed match on the `block` action does not imply a `pass`. It implies only that the rule has not taken effect. To then allow ports lower than 1024, a rule such as:

```
pass in quick from any to any port < 1024
```

...would be needed before the first `block`. To create a new group for processing all inbound packets on `le0/le1/lo0`, with the default being to block all inbound packets, you would use a rule such as:

```
block in all
block in quick on le0 all head 100
block in quick on le1 all head 200
block in quick on lo0 all head 300
```

and to then allow ICMP packets in on `le0` only, you would use:

```
pass in proto icmp all group 100
```

Note that because only inbound packets on `le0` are processed by group 100, there is no need to respecify the interface name. Likewise, you could further breakup processing of TCP as follows:

```
block in proto tcp all head 110 group 100
pass in from any to any port = 23 group 110
```

...and so on. The last line, if written without the groups, would be:

```
pass in on le0 proto tcp from any to any port = telnet
```

EXAMPLE 2 Using the Fall-through Nature of Rule Parsing (Continued)

Note, that if you wanted to specify `port = telnet`, you would need to specify `proto tcp`, because the parser interprets each rule on its own and qualifies all service and port names with the protocol specified.

Files `/etc/ipf/ipf.conf` Location of rules file that is read upon startup of IP Filter feature.

- `/dev/ipauth`
- `/dev/ipl`
- `/dev/ipstate`
- `/etc/hosts`
- `/etc/services`

Attributes See [attributes\(5\)](#) for a description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Committed

See Also [ipf\(1M\)](#), [ipfstat\(1M\)](#), [ipmon\(1M\)](#), [svc.ipfd\(1M\)](#), [attributes\(5\)](#), [ipfilter\(5\)](#)

Oracle Solaris Administration: IP Services

Notes See [svc.ipfd\(1M\)](#) for information regarding custom policies.

Name ipnat, ipnat.conf – IP NAT file format

Synopsis ipnat.conf

Description The ipnat or ipnat.conf configuration files are associated with the Solaris IP Filter feature. See [ipfilter\(5\)](#).

The format for files accepted by ipnat is described by the following grammar:

```

ipmap ::= mapblock | redir | map .

map ::= mapit ifname ipmask "->" dstipmask [ mapport | mapproxy ] \
      mapoptions.
map ::= mapit ifname fromto "->" dstipmask [ mapport ] mapoptions.
mapblock ::= "map-block" ifname ipmask "->" ipmask [ ports ] \
            mapoptions.
redir ::= "rdr" ifname ipmask dport "->" ip [ "," ip ] rdrport \
        rdoptions .

dport ::= "port" port-number [ "-" port-number ] .
ports ::= "ports" numports | "auto" .
rdrport ::= "port" port-number .
mapit ::= "map" | "bimap" .
fromto ::= "from" object "to" object .
ipmask ::= ip "/" bits | ip "/" mask | ip "netmask" mask .
dstipmask ::= ipmask | "range" ip "-" ip .
mapport ::= "portmap" tcpudp portspec .
mapoptions ::= [ tcpudp ] [ "frag" ] [ age ] [ clamp ] [ mapproxy ] .
rdoptions ::= rdrproto [ rr ] [ "frag" ] [ age ] [ clamp ] \
            [ rdrproxy ] .

object ::= addr [ port-comp | port-range ] .
addr ::= "any" | nummask | host-name [ "mask" ipaddr | "mask" \
        hexnumber ] .
port-comp ::= "port" compare port-number .
port-range ::= "port" port-number range port-number .
rdrproto ::= tcpudp | protocol .

rr ::= "round-robin" .
age ::= "age" decnumber [ "/" decnumber ] .
clamp ::= "mssclamp" decnumber .
tcpudp ::= "tcp/udp" | protocol .
mapproxy ::= "proxy" "port" port proxy-name '/' protocol
rdrproxy ::= "proxy" proxy-name .

protocol ::= protocol-name | decnumber .
nummask ::= host-name [ "/" decnumber ] .
portspec ::= "auto" | port-number ":" port-number .
port ::= port-number | port-name .

```

```
port-number ::= number { numbers } .
ifname ::= 'A' - 'Z' { 'A' - 'Z' } numbers .

numbers ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' .
```

For standard NAT functionality, a rule should start with `map` and then proceed to specify the interface for which outgoing packets will have their source address rewritten.

Packets that will be rewritten can only be selected by matching the original source address. When specifying an address for matching, a netmask must be specified with the IP address.

The address selected for replacing the original is chosen from an IP address/netmask pair. A netmask of all 1's, indicating a hostname, is valid. A netmask of thirty-one 1's (255.255.255.254) is considered invalid, because there is no space for allocating host IP addresses after consideration for broadcast and network addresses.

When remapping TCP and UDP packets, it is also possible to change the source port number. Either TCP or UDP or both can be selected by each rule, with a range of port numbers to remap into given as *port-number:port-number*.

Commands The following commands are recognized by IP Filter's NAT code:

<code>map</code>	Used for mapping one address or network to another in an unregulated round-robin fashion.
<code>rdr</code>	Used for redirecting packets to one IP address and port pair to another.
<code>bimap</code>	Used for setting up bidirectional NAT between an external IP address and an internal IP address.
<code>map-block</code>	Sets up static IP-address-based translation, based on an algorithm to squeeze the addresses to be translated into the destination range.

Matching For basic NAT and redirection of packets, the address subject to change is used along with its protocol to check if a packet should be altered. The packet *matching* part of the rule is to the left of the symbol `→` in each rule.

The IPFilter software allows for complex matching of packets. In place of the address which is to be translated, an IP address and port number comparison can be made using the same expressions available with `ipf`. A simple NAT rule could be written as:

```
map de0 10.1.0.0/16 -> 201.2.3.4/32
map de0 fec0:1::/64 -> fec0:2::2/128

or as

map de0 from 10.1.0.0/16 to any -> 201.2.3.4/32
map de0 from fec0:1::/64 to any -> fec0:2::2/128
```

As is true of all NAT rules, you can compare against only IP address and port numbers. In addition, you cannot specify both IPv4 and IPv6 addresses in the same rule.

Translation To the right of the \rightarrow is the address and port specification that will be written into the packet, provided it has already successfully matched the prior constraints. The case of redirections (`rdr`) is the simplest: the new destination address is that specified in the rule. For map rules, the destination address will be one for which the tuple combining the new source and destination is known to be unique.

If the packet is either a TCP or UDP packet, the destination and source ports enter into the comparison also. If the tuple already exists, the IP Filter software increments the port number first, within the available range specified by `portmap`, and, if there is no unique tuple, the source address is incremented within the specified netmask. If a unique tuple cannot be determined, then the packet will not be translated.

The map-block is more limited in how it searches for a new, free, and unique tuple, in that it will use an algorithm to determine what the new source address should be, staying within the range of available ports. The IP address is never changed, nor does the port number ever exceed its allotted range.

ICMPIDMAP Feature ICMP messages can be divided into two groups, errors and queries. ICMP errors are generated as a response to another IP packet. IP Filter will take care that ICMP errors that are the response of a NAT-ed IP packet are handled properly.

For four types of ICMP queries (echo request, timestamp request, information request and address mask request), IP Filter supports an additional mapping called “ICMP id mapping”. These four types of ICMP queries use a unique identifier called the ICMP id. This id is set by the process sending the ICMP query and is usually equal to the process id. The receiver of the ICMP query will use the same id in its response, thus enabling the sender to recognize that the incoming ICMP reply is intended for him and is an answer to a query that he made. The ICMP id mapping feature modifies these ICMP ids in a way identical to the modification performed by `portmap` for TCP or UDP.

When using the ICMP id mapping feature, you do not need an IP address per host behind the NAT box that wants to perform ICMP queries. The two numbers that follow the `icmpidmap` keyword are the first and the last `icmp id` numbers that can be used. There is one important caveat: if you map to an IP address that belongs to the NAT box itself (notably if you have only a single public IP address), then you must ensure that the NAT box does not use the `icmpidmap` range that you specified in the map rule. Since the ICMP id is usually the process id, it is wise to restrict the largest permissible process id (PID) on your operating system to a value such as 63999 and use the range 64000:65535 for ICMP id mapping.

Kernel Proxies The IP Filter software comes with a few, simple, proxies built into the code that is loaded into the kernel to allow secondary channels to be opened without forcing the packets through a user program. Kernel proxies are not supported for IPv6 NAT-ing.

Transparent Proxies True transparent proxying should be performed using the redirect (`rdr`) rules directing ports to `localhost` (127.0.0.1), with the proxy program doing a lookup through `/dev/ipnat` to determine the real source and address of the connection.

Load Balancing Two options for use with `rdr` are available to support primitive, round-robin-based load balancing. The first option allows for a `rdr` to specify a second destination, as follows:

```
rdr le0 203.1.2.3/32 port 80 -> 203.1.2.3,203.1.2.4 port 80 tcp
```

The preceding would send alternate connections to either 203.1.2.3 or 203.1.2.4. In scenarios where the load is being spread among a larger set of servers, you can use:

```
rdr le0 203.1.2.3/32 port 80 -> 203.1.2.3,203.1.2.4 port 80 tcp \
round-robin
```

```
rdr le0 203.1.2.3/32 port 80 -> 203.1.2.5 port 80 tcp round-robin
```

In this case, a connection will be redirected to 203.1.2.3, then 203.1.2.4, and then 203.1.2.5 before going back to 203.1.2.3. In accomplishing this, the rule is removed from the top of the list and added to the end, automatically, as required. This will not effect the display of rules using `ipnat -l`, only the internal application order.

Examples **EXAMPLE 1** Using the map Command

The following are variations of the map command.

To change IP addresses used internally from network 10 into an ISP-provided 8-bit subnet at 209.1.2.0 through the `ppp0` interface, use the following:

```
map ppp0 10.0.0.0/8 -> 209.1.2.0/24
```

An obvious problem is that you are trying to squeeze over sixteen million IP addresses into a 254-address space. To increase the scope, remapping for TCP and/or UDP, port remapping can be used, as follows:

```
map ppp0 10.0.0.0/8 -> 209.1.2.0/24 portmap tcp/udp 1025:65000
```

The preceding falls only 527,566 addresses short of the space available in network 10. If we combine these rules, they would need to be specified as follows:

```
map ppp0 10.0.0.0/8 -> 209.1.2.0/24 portmap tcp/udp 1025:65000
map ppp0 10.0.0.0/8 -> 209.1.2.0/24
```

...so that all TCP/UDP packets were port mapped and only other protocols, such as ICMP, have their IP address changed. In some instances, it is more appropriate to use the keyword `auto` in place of an actual range of port numbers if you want to guarantee simultaneous access to all within the given range. However, in the preceding case, it would default to one port per IP address, because you need to squeeze 24 bits of address space into eight bits. A good example of how `auto` is used is:

EXAMPLE 1 Using the map Command *(Continued)*

```
map ppp0 172.192.0.0/16 -> 209.1.2.0/24 portmap tcp/udp auto
```

This would result in each IP address being given a small range of ports to use (252). The problem here is that the map directive tells the NAT code to use the next address/port pair available for an outgoing connection, resulting in no easily discernible relation between external addresses/ports and internal ones. This is overcome by using map-block as follows:

```
map-block ppp0 172.192.0.0/16 -> 209.1.2.0/24 ports auto
```

For example, this would result in 172.192.0.0/24 being mapped to 209.1.2.0/32 with each address, from 172.192.0.0 to 172.192.0.255 having 252 ports of its own. As distinguished from the preceding use of map, if, for some reason, the user of (say) 172.192.0.2 wanted 260 simultaneous connections going out, he would be limited to 252 with map-block but would just move on to the next IP address with the map command.

EXAMPLE 2 Mapping from Class B Network to Single Address

The following directive maps from a class B network to a single address.

```
map de0 10.1.0.0/16 -> 201.2.3.4/32
```

An equivalent directive is:

```
map de0 from 10.1.0.0/16 to any -> 201.2.3.4/32
```

Files /etc/ipf/ipnat.conf Location of rules file that is read upon startup of IP Filter feature.

- /dev/ipnat
- /etc/services
- /etc/hosts

Attributes See [attributes\(5\)](#) for a description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Committed

See Also [ipf\(1M\)](#), [ipnat\(1M\)](#), [ipf\(4\)](#), [hosts\(4\)](#), [attributes\(5\)](#), [ipfilter\(5\)](#)

Oracle Solaris Administration: IP Services

Name ipnodes – symbolic link to hosts database

Synopsis /etc/inet/ipnodes

Description The ipnodes file is now a symbolic link to the /etc/hosts file. See [hosts\(4\)](#). In prior releases of the Solaris operating system, ipnodes was a local database distinct from hosts. The man page for a given Solaris release describes the ipnodes file for that release.

See Also [hosts\(4\)](#)

Name ippool, ippool.conf – IP pool file format

Synopsis ippool.conf

Description The format for files accepted by `ippool(1M)` is described by the following grammar:

```

line ::= table | groupmap .
table ::= "table" role tabletype .
groupmap ::= "group-map" inout role number ipfgroup
tabletype ::= ipftree | ipfhash .

role ::= "role" "=" "ipf" .
inout ::= "in" | "out" .

ipftree ::= "type" "=" "tree" number "{" addrlist "}" .
ipfhash ::= "type" "=" "hash" number hashopts "{" hashlist "}" .

ipfgroup ::= setgroup hashopts "{" grouplist "}" |
             hashopts "{" setgrouplist "}" .
setgroup ::= "group" "=" groupname .

hashopts ::= size [ seed ] | seed .

size ::= "size" "=" number .
seed ::= "seed" "=" number .

addrlist ::= range [ "," addrlist ] .
grouplist ::= groupentry [ ";" grouplist ] | groupentry ";" |
             addrmask ";" | addrmask ";" [ grouplist ] .

setgrouplist ::= groupentry ";" [ setgrouplist ] .

groupentry ::= addrmask "," setgroup .

range ::= addrmask | "!" addrmask .

hashlist ::= hashentry ";" [ hashlist ] .
hashentry ::= addrmask .

addrmask ::= ipaddr | ipaddr "/" mask .

mask ::= number | ipaddr .

groupname ::= number | name .

number ::= digit { digit } .

ipaddr = host-num "." host-num "." host-num "." host-num | ipv6addr .

```

```
host-num = digit [ digit [ digit ] ] .  
  
digit ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" .  
  
name ::= letter { letter | digit } .
```

The IP pool configuration file is used for defining a single object that contains a reference to multiple IP address/netmask pairs. A pool can consist of a mixture of netmask sizes, from 0 to 32.

In the current release, only IPv4 addressing is supported in IP pools.

The IP pool configuration file provides for defining two different mechanisms for improving speed in matching IP addresses with rules. The first, `table`, defines a lookup table to provide a single reference in a filter rule to multiple targets. The second mechanism, `group-map`, provides a mechanism to target multiple groups from a single filter line.

The `group-map` command can be used only with filter rules that use the `call` command to invoke either `fr_srcgrpmmap` or `fr_dstgrpmmap`, to use the source or destination address, respectively, for determining which filter group to jump to next for continuation of filter packet processing.

Pool Types Two storage formats are provided: hash tables and tree structure. The hash table is intended for use with objects that all contain the same netmask or a few, different sized-netmasks of non-overlapping address space. The tree is designed for supporting exceptions to a covering mask, in addition to normal searching as you would do with a table. It is not possible to use the tree data storage type with `group-map` configuration entries.

Pool Roles When a pool is defined in the configuration file, it must have an associated role. At present the only supported role is `ipf`. Future development might see further expansion of the use of roles by other sections of IPFilter code.

Examples The following examples show how the pool configuration file is used with the `ipf` configuration file to enhance the succinctness of the latter file's entries.

EXAMPLE 1 Referencing Specific Pool

The following example shows how a filter rule makes reference to a specific pool for matching of the source address.

```
pass in from pool/100 to any
```

The following pool configuration matches IP addresses 1.1.1.1 and any in 2.2.0.0/16, except for those in 2.2.2.0/24.

```
table role = ipf type = tree number = 100  
    { 1.1.1.1/32, 2.2.0.0/16, !2.2.2.0/24 };
```

EXAMPLE 2 ipf Configuration Entry

The following `ipf.conf` excerpt uses the `fr_srcgrpmmap/fr_dstgrpmmap` lookups to use the group-map facility to look up the next group to use for filter processing, providing the call filter rule is matched.

```
call now fr_srcgrpmmap/1010 in all
call now fr_dstgrpmmap/2010 out all
pass in all group 1020
block in all group 1030
pass out all group 2020
block out all group 2040
```

An `ippool` configuration to work with the preceding `ipf.conf` segment might look like the following:

```
group-map in role = ipf number = 1010
    { 1.1.1.1/32, group = 1020; 3.3.0.0/16, group = 1030; };
group-map out role = ipf number = 2010 group = 2020
    { 2.2.2.2/32; 4.4.0.0/16; 5.0.0.0/8, group = 2040; };
```

- Files**
- `/dev/ippool`
 - `/etc/ipf/ippool.conf`
 - `/etc/hosts`

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	network/ipfilter
Interface Stability	Committed

See Also [ipf\(1M\)](#), [ipnat\(1M\)](#), [ippool\(1M\)](#), [ipf\(4\)](#), [attributes\(5\)](#), [hosts\(4\)](#)

Name issue – issue identification file

Description The file `/etc/issue` contains the issue or project identification to be printed as a login prompt. `issue` is an ASCII file that is read by program `ttymon` and then written to any terminal spawned or respawned, prior to the normal prompt.

Files `/etc/issue`

See Also [login\(1\)](#), [ttymon\(1M\)](#)

Name kadm5.acl – Kerberos access control list (ACL) file

Synopsis /etc/krb5/kadm5.acl

Description The ACL file is used by the `kadmin(1M)` command to determine which principals are allowed to perform Kerberos administration actions. For operations that affect principals, the ACL file also controls which principals can operate on which other principals. The location of the ACL file is determined by the `acl_file` configuration variable in the `kdc.conf(4)` file. The default location is `/etc/krb5/kadm5.acl`.

For incremental propagation, see `kadmin(1M)`. The ACL file must contain the `kiprop` service principal with propagation privileges in order for the slave KDC to pull updates from the master's principal database. Refer to the EXAMPLES section for this case.

The ACL file can contain comment lines, null lines, or lines that contain ACL entries. Comment lines start with the pound sign (`#`) and continue until the end of the line.

The order of entries is significant. The first matching entry specifies the principal on which the control access applies, whether it is on just the principal or on the principal when it operates on a target principal.

Lines containing ACL entries must have the following format:

principal operation-mask [operation-target]

principal Specifies the principal on which the *operation-mask* applies. Can specify either a partially or fully qualified Kerberos principal name. Each component of the name can be substituted with a wildcard, using the asterisk (`*`) character.

operation-mask Specifies what operations can or cannot be performed by a principal matching a particular entry. Specify *operation-mask* as one or more *privileges*.

A *privilege* is a string of one or more of the following characters: a, A, c, C, d, D, i, I, l, L, m, M, p, P, u, U, x, or *. Generally, if the character is lowercase, the privilege is allowed and if the character is uppercase, the operation is disallowed. The x and * characters are exceptions to the uppercase convention.

The following *privileges* are supported:

- a Allows the addition of principals or policies in the database.
- A Disallows the addition of principals or policies in the database.
- c Allows the changing of passwords for principals in the database.
- C Disallows the changing of passwords for principals in the database.

d	Allows the deletion of principals or policies in the database.
D	Disallows the deletion of principals or policies in the database.
i	Allows inquiries to the database.
I	Disallows inquiries to the database.
l	Allows the listing of principals or policies in the database.
L	Disallows the listing of principals or policies in the database.
m	Allows the modification of principals or policies in the database.
M	Disallows the modification of principals or policies in the database.
p	Allow the propagation of the principal database.
P	Disallow the propagation of the principal database.
u	Allows the creation of one-component user principals whose password can be validated with PAM.
U	Negates the u privilege.
x	Short for specifying privileges a, d,m,c,i, and l. The same as *.
*	Short for specifying privileges a, d,m,c,i, and l. The same as x.
<i>operation-target</i>	Optional. When specified, the <i>privileges</i> apply to the <i>principal</i> when it operates on the <i>operation-target</i> . For the <i>operation-target</i> , you can specify a partially or fully qualified Kerberos principal name. Each component of the name can be substituted by a wildcard, using the asterisk (*) character.

Examples EXAMPLE 1 Specifying a Standard, Fully Qualified Name

The following ACL entry specifies a standard, fully qualified name:

```
user/instance@realm adm
```

The *operation-mask* applies only to the `user/instance@realm` principal and specifies that the principal can add, delete, or modify principals and policies, but it cannot change passwords.

EXAMPLE 2 Specifying a Standard Fully Qualified Name and Target

The following ACL entry specifies a standard, fully qualified name:

```
user/instance@realm cim service/instance@realm
```

The *operation-mask* applies only to the `user/instance@realm` principal operating on the `service/instance@realm` target, and specifies that the principal can change the target's password, request information about the target, and modify it.

EXAMPLE 3 Specifying a Name Using a Wildcard

The following ACL entry specifies a name using a wildcard:

```
user/*@realm ac
```

The *operation-mask* applies to all principals in realm `realm` whose first component is `user` and specifies that the principals can add principals and change passwords.

EXAMPLE 4 Specifying a Name Using a Wildcard and a Target

The following ACL entry specifies a name using a wildcard and a target:

```
user/*@realm i */instance@realm
```

The *operation-mask* applies to all principals in realm `realm` whose first component is `user` and specifies that the principals can perform inquiries on principals whose second component is `instance` and realm is `realm`.

EXAMPLE 5 Specifying Incremental Propagation Privileges

The following ACL entry specifies propagation privileges for the `kiprop` service principal:

```
kiprop/slavehost@realm p
```

The *operation-mask* applies to the `kiprop` service principal for the specified slave host `slavehost` in realm `realm`. This specifies that the associated `kiprop` service principal can receive incremental principal updates.

Files `/etc/krb5/kdc.conf` KDC configuration information.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5
Interface Stability	Committed

See Also [kpasswd\(1\)](#), [gkadmin\(1M\)](#), [kadmind\(1M\)](#), [kadmin.local\(1M\)](#), [kdb5_util\(1M\)](#), [kdc.conf\(4\)](#), [attributes\(5\)](#), [kerberos\(5\)](#), [pam_krb5_migrate\(5\)](#)

Name kdc.conf – Key Distribution Center (KDC) configuration file

Synopsis /etc/krb5/kdc.conf

Description The `kdc.conf` file contains KDC configuration information, including defaults used when issuing Kerberos tickets. This file must reside on all KDC servers. After you make any changes to the `kdc.conf` file, stop and restart the `krb5kdc` daemon on the KDC for the changes to take effect.

The format of the `kdc.conf` consists of section headings in square brackets (`[]`). Each section contains zero or more configuration variables (called relations), of the form of:

relation = relation-value

or

```
relation-subsection = {  
    relation = relation-value  
    relation = relation-value  
}
```

The `kdc.conf` file contains one of more of the following three sections:

`kdcdefaults`

Contains default values for overall behavior of the KDC.

`realms`

Contains subsections for Kerberos realms, where *relation-subsection* is the name of a realm. Each subsection contains relations that define KDC properties for that particular realm, including where to find the Kerberos servers for that realm.

`logging`

Contains relations that determine how Kerberos programs perform logging.

The `kdcdefaults` Section The following relation can be defined in the `[kdcdefaults]` section:

`kdc_ports`

This relation lists the UDP ports on which the Kerberos server should listen by default.

This list is a comma-separated list of integers. If the assigned value is 0, the Kerberos server does not listen on any UDP port. If this relation is not specified, the Kerberos server listens on port 750 and port 88.

`kdc_tcp_ports`

This relation lists the TCP ports on which the Kerberos server should listen by default. This list is a comma-separated list of integers. If the assigned value is 0, the Kerberos server does not listen on any TCP port. If this relation is not specified, the Kerberos server listens on the `kdc` TCP port specified in `/etc/services`. If this port is not found in `/etc/services` the Kerberos server defaults to listen on TCP port 88.

`kdc_max_tcp_connections`

This relation controls the maximum number of TCP connections the KDC allows. The minimum value is 10. If this relation is not specified, the Kerberos server allows a maximum of 30 TCP connections.

The `realms` Section This section contains subsections for Kerberos realms, where *relation-subsection* is the name of a realm. Each subsection contains relations that define KDC properties for that particular realm.

The following relations can be specified in each subsection:

`acl_file`

(string) Location of the Kerberos V5 access control list (ACL) file that `kadmin` uses to determine the privileges allowed to each principal on the database. The default location is `/etc/krb5/kadm5.acl`.

`admin_keytab`

(string) Location of the keytab file that `kadmin` uses to authenticate to the database. The default location is `/etc/krb5/kadm5.keytab`.

`database_name`

(string) Location of the Kerberos database for this realm. The default location is `/var/krb5/principal`.

`default_principal_expiration`

(absolute time string) The default expiration date of principals created in this realm. See the Time Format section in [kinit\(1\)](#) for the valid absolute time formats you can use for `default_principal_expiration`.

`default_principal_flags`

(flag string) The default attributes of principals created in this realm. Some of these flags are better to set on an individual principal basis through the use of the attribute modifiers when using the `kadmin` command to create and modify principals. However, some of these options can be applied to all principals in the realm by adding them to the list of flags associated with this relation.

A “flag string” is a list of one or more of the flags listed below preceded by a minus (-) or a plus (+) character, indicating that the option that follows should be enabled or disabled.

Flags below marked with an asterisk (*) are flags that are best applied on an individual principal basis through the `kadmin` or `gkadmin` interface rather than as a blanket attribute to be applied to all principals.

`postdateable`

Create postdateable tickets.

`forwardable`

Create forwardable tickets.

tgt-based

Allow TGT-based requests.

renewable

Create Renewable tickets.

proxiabile

Create Proxiabile tickets.

dup-skey

Allow DUP_SKEY requests, this enables user-to-user authentication.

preauth

Require the use of pre-authentication data whenever principals request TGTs.

hwauth

Require the use of hardware-based pre-authentication data whenever principals request TGTs.

*** allow-tickets**

Allow tickets to be issued for all principals.

*** pwdchange**

Require principal's to change their password.

*** service**

Enable or disable a service.

*** pwservice**

Mark principals as password changing principals.

An example of `default_principal_flags` is shown in EXAMPLES, below.

dict_file

(string) Location of the dictionary file containing strings that are not allowed as passwords. A principal with any password policy is not allowed to select a password in the dictionary. The default location is `/var/krb5/kadm5.dict`.

kadmind_port

(port number) The port that the kadmind daemon is to listen on for this realm. The assigned port for kadmind is 749.

key_stash_file

(string) Location where the master key has been stored (by `kdb5_util stash`). The default location is `/var/krb5/.k5.realm`, where *realm* is the Kerberos realm.

kdc_ports

(string) The list of UDP ports that the KDC listens on for this realm. By default, the value of `kdc_ports` as specified in the `[kdcdefaults]` section is used.

kdc_tcp_ports

(string) The list of TCP ports that the KDC listens on (in addition to the UDP ports specified by `kdc_ports`) for this realm. By default, the value of `kdc_tcp_ports` as specified in the `[kdcdefaults]` section is used.

master_key_name

(string) The name of the master key.

master_key_type

(key type string) The master key's key type. This is used to determine the type of encryption that encrypts the entries in the principal db. `des-cbc-crc`, `des3-cbc-md5`, `des3-cbc-sha1-kd`, `arcfour-hmac-md5`, `arcfour-hmac-md5-exp`, `aes128-cts-hmac-sha1-96`, and `aes256-cts-hmac-sha1-96` are supported at this time (`aes256-cts-hmac-sha1-96` is the default). If you set this to `des3-cbc-sha1-kd` all systems that receive copies of the principal db, such as those running slave KDC's, must support `des3-cbc-sha1-kd`.

`arcfour-hmac-md5-exp`, `des-cbc-md5`, and `des-cbc-crc` are considered weak encryption types. See the `allow_weak_crypto` relation in the [krb5.conf\(4\)](#) manual page for more information.

max_life

(delta time string) The maximum time period for which a ticket is valid in this realm. See the Time Format section in [kinit\(1\)](#) for the valid time duration formats you can use for `max_life`.

max_renewable_life

(delta time string) The maximum time period during which a valid ticket can be renewed in this realm. See the Time Format section in [kinit\(1\)](#) for the valid time duration formats you can use for `max_renewable_life`.

sunw_dbprop_enable = [true | false]

Enable or disable incremental database propagation. Default is `false`.

sunw_dbprop_master_ologsize = N

Specifies the maximum number 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.

sunw_dbprop_slave_poll = N[s, m, h]

Specifies how often the slave KDC polls for new updates that the master might have. Default is `2m` (two minutes).

supported_ectypes

List of `key/salt` strings. The default `key/salt` combinations of principals for this realm. The `key` is separated from the `salt` by a colon (`:`) or period (`.`). Multiple `key/salt` strings can be used by separating each string with a space. The `salt` is additional information

encoded within the key that tells what kind of key it is. Only the *normal* salt is supported at this time, for example, `des-cbc-crc:normal`. If this relation is not specified, the default setting is:

```
aes256-cts-hmac-sha1-96:normal \  
aes128-cts-hmac-sha1-96:normal \  
des3-cbc-sha1-kd:normal \  
arcfour-hmac-md5:normal \  
des-cbc-md5:normal \  
arcfour-hmac-md5:normal
```

`arcfour-hmac-md5-exp`, `des-cbc-md5`, and `des-cbc-crc` are considered weak encryption types. See the `allow_weak_crypto` relation in the [krb5.conf\(4\)](#) manual page for more information.

`reject_bad_transit`

This boolean specifies whether the list of transited realms for cross-realm tickets should be checked against the transit path computed from the realm names and the `[capaths]` section of its [krb5.conf\(4\)](#) file.

The default for `reject_bad_transit` is `true`.

The logging Section This section indicates how Kerberos programs perform logging. The same relation can be repeated if you want to assign it multiple logging methods.

There are similar logging and relations defined in [krb5.conf\(4\)](#). The relations defined in [krb5.conf\(4\)](#) supercede the same relations defined in this section.

The following relations can be defined in the `[logging]` section:

`kdc`

Specifies how the KDC is to perform its logging. The default is `FILE:/var/krb5/kdc.log`.

`admin_server`

Specifies how the administration server is to perform its logging. The default is `FILE:/var/krb5/kadmin.log`.

`default`

Specifies how to perform logging in the absence of explicit specifications.

The `[logging]` relations can have the following values:

`FILE:filename`

or

`FILE=filename`

This value causes the entity's logging messages to go to the specified file. If the '=' form is used, the file is overwritten. If the ':' form is used, the file is appended to.

STDERR

This value sends the entity's logging messages to its standard error stream.

CONSOLE

This value sends the entity's logging messages to the console, if the system supports it.

DEVICE=*devicename*

This sends the entity's logging messages to the specified device.

SYSLOG[:*severity* [:*facility*]]

This sends the entity's logging messages to the system log.

The *severity* argument specifies the default severity of system log messages. This default can be any of the following severities supported by the `syslog(3C)` call, minus the `LOG_` prefix: `LOG_EMERG`, `LOG_ALERT`, `LOG_CRIT`, `LOG_ERR`, `LOG_WARNING`, `LOG_NOTICE`, `LOG_INFO`, and `LOG_DEBUG`. For example, a value of `CRIT` would specify `LOG_CRIT` severity.

The *facility* argument specifies the facility under which the messages are logged. This can be any of the following facilities supported by the `syslog(3C)` call minus the `LOG_` prefix: `LOG_KERN`, `LOG_USER`, `LOG_MAIL`, `LOG_DAEMON`, `LOG_AUTH`, `LOG_LPR`, `LOG_NEWS`, `LOG_UUCP`, `LOG_CRON`, and `LOG_LOCAL0` through `LOG_LOCAL7`.

If no severity is specified, the default is `ERR`. If no facility is specified, the default is `AUTH`.

In the following example, the logging messages from the KDC go to the console and to the system log under the facility `LOG_DAEMON` with default severity of `LOG_INFO`; the logging messages from the administration server are appended to the `/var/krb5/kadmin.log` file and sent to the `/dev/tty04` device.

```
[logging]
kdc = CONSOLE
kdc = SYSLOG:INFO:DAEMON
admin_server = FILE:/export/logging/kadmin.log
admin_server = DEVICE=/dev/tty04
```

PKINIT-specific Options The following are `pkinit`-specific options. These values can be specified in `[kdcdefaults]` as global defaults, or within a realm-specific subsection of `[realms]`. A realm-specific value overrides, does not add to, a generic `[kdcdefaults]` specification. The search order is

1. realm-specific subsection of `[realms]`

`[realms]`

`[realms]`

```
EXAMPLE.COM = {
    pkinit_anchors = FILE:/usr/local/example.com.crt
}
```

2. generic value in the `[kdcdefaults]` section

[kdcdefaults]	
pkinit_anchors = DIR:/usr/local/generic_trusted_cas/	
pkinit_identity = <i>URI</i>	Specifies the location of the KDC's X.509 identity information. This option is required if <code>pkinit</code> is supported by the KDC. Valid <i>URI</i> types are FILE, DIR, PKCS11, PKCS12, and ENV. See the PKINIT URI Types section for more details.
pkinit_anchors = <i>URI</i>	Specifies the location of trusted anchor (root) certificates which the KDC trusts to sign client certificates. This option is required if <code>pkinit</code> is supported by the KDC. This option can be specified multiple times. Valid <i>URI</i> types are FILE and DIR. See the PKINIT URI Types section for details.
pkinit_pool	Specifies the location of intermediate certificates which can be used by the KDC to complete the trust chain between a client's certificate and a trusted anchor. This option can be specified multiple times. Valid <i>URI</i> types are FILE and DIR. See the PKINIT URI Types section for more details.
pkinit_revoke	Specifies the location of Certificate Revocation List (CRL) information to be used by the KDC when verifying the validity of client certificates. This option can be specified multiple times. The default certificate verification process always checks the available revocation information to see if a certificate has been revoked. If a match is found for the certificate in a CRL, verification fails. If the certificate being verified is not listed in a CRL, or there is no CRL present for its issuing CA, and <code>pkinit_require_crl_checking</code> is <code>false</code> , then verification succeeds. The only valid <i>URI</i> types is DIR. See the PKINIT URI Types section for more details. If <code>pkinit_require_crl_checking</code> is <code>true</code> and there is no CRL information available for the issuing CA, verification fails. <code>pkinit_require_crl_checking</code> should be set to <code>true</code> if the policy is such that up-to-date CRLs must be present for every CA.
pkinit_dh_min_bits	Specifies the minimum number of bits the KDC is willing to accept for a client's Diffie-Hellman key.
pkinit_allow_upn	Specifies that the KDC is willing to accept client certificates with the Microsoft UserPrincipalName (UPN) Subject Alternative Name (SAN). This means the KDC accepts the binding of the UPN in the certificate to the Kerberos principal name. The default is <code>false</code> .

		Without this option, the KDC only accepts certificates with the <code>id-pkinit-san</code> as defined in RFC4556. There is currently no option to disable SAN checking in the KDC.
<code>pkinit_eku_checking</code>		This option specifies what Extended Key Usage (EKU) values the KDC is willing to accept in client certificates. The values recognized in the <code>kdc.conf</code> file are:
	<code>kpClientAuth</code>	This is the default value and specifies that client certificates must have the <code>id-pkinit-KPClientAuth</code> EKU as defined in RFC4556.
	<code>scLogin</code>	If <code>scLogin</code> is specified, client certificates with the Microsoft Smart Card Login EKU (<code>id-ms-kp-sc-logon</code>) is accepted.
PKINIT URI Types	<code>FILE:file-name[,key-file-name]</code>	
		This option has context-specific behavior.
	<code>pkinit_identity</code>	<i>file-name</i> specifies the name of a PEM-format file containing the user's certificate. If <i>key-file-name</i> is not specified, the user's private key is expected to be in <i>file-name</i> as well. Otherwise, <i>key-file-name</i> is the name of the file containing the private key.
	<code>pkinit_anchors</code> <code>pkinit_pool</code>	<i>file-name</i> is assumed to be the name of an OpenSSL-style ca-bundle file. The <i>ca-bundle</i> file should be base-64 encoded.
	<code>DIR:directory-name</code>	
		This option has context-specific behavior.
	<code>pkinit_identity</code>	<i>directory-name</i> specifies a directory with files named <code>*.crt</code> and <code>*.key</code> , where the first part of the file name is the same for matching pairs of certificate and private key files. When a file with a name ending with <code>.crt</code> is found, a matching file ending with <code>.key</code> is assumed to contain the private key. If no such file is found, then the certificate in the <code>.crt</code> is not used.
	<code>pkinit_anchors</code> <code>pkinit_pool</code>	<i>directory-name</i> is assumed to be an OpenSSL-style hashed CA directory where each CA cert is stored in a file named <code>hash-of-ca-cert.#</code> . This infrastructure is encouraged, but all files in the directory is examined and if they contain certificates (in PEM format), they are used.
	<code>pkinit_revoke</code>	<i>directory-name</i> is assumed to be an OpenSSL-style hashed CA directory where each revocation list is stored in a file named <code>hash-of-ca-cert.r#</code> . This infrastructure is encouraged, but all files in

the directory is examined and if they contain a revocation list (in PEM format), they are used.

PKCS12:pkcs12-file-name

`pkcs12-file-name` is the name of a PKCS #12 format file, containing the user's certificate and private key.

PKCS11:[slotid=*slot-id*][:token=*token-label*][:certid=*cert-id*][:certlabel=*cert-label*]

All keyword/values are optional. PKCS11 modules (for example, `opensc-pkcs11.so`) must be installed as a crypto provider under `libpkcs11(3LIB)`. `slotid=` and/or `token=` can be specified to force the use of a particular smart card reader or token if there is more than one available. `certid=` and/or `certlabel=` can be specified to force the selection of a particular certificate on the device. See the `pkinit_cert_match` configuration option for more ways to select a particular certificate to use for `pkinit`.

ENV:*environment-variable-name*

environment-variable-name specifies the name of an environment variable which has been set to a value conforming to one of the previous values. For example, `ENV:X509_PROXY`, where environment variable `X509_PROXY` has been set to `FILE:/tmp/my_proxy.pem`.

Examples EXAMPLE 1 Sample `kdc.conf` File

The following is an example of a `kdc.conf` file:

```
[kdcdefaults]
    kdc_ports = 88

[realms]
    ATHENA.MIT.EDU = {
        kadmind_port = 749
        max_life = 10h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        default_principal_flags = +preauth,+forwardable,-postdateable
        master_key_type = aes256-cts-hmac-sha1-96:normal
        supported_encetypes = aes256-cts-hmac-sha1-96:normal
    }

[logging]
    kdc = FILE:/export/logging/kdc.log
    admin_server = FILE:/export/logging/kadmin.log
```

Files `/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`.

`/var/krb5/principal`

Kerberos principal database.

`/var/krb5/principal.ulog`

The update log file for incremental propagation.

`/var/krb5/kadm5.dict`

Dictionary of strings explicitly disallowed as passwords.

`/var/krb5/kdc.log`

KDC logging file.

`/var/krb5/kadmin.log`

Kerberos administration server logging file.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/security/kerberos-5
Interface Stability	See below.

All of the keywords, except for the PKINIT keywords are Committed. The PKINIT keywords are Volatile.

See Also [kpasswd\(1\)](#), [gkadmin\(1M\)](#), [kadmin\(1M\)](#), [kadmin.local\(1M\)](#), [kdb5_util\(1M\)](#), [kpropd\(1M\)](#), [libpkcs11\(3LIB\)](#), [syslog\(3C\)](#), [kadm5.acl\(4\)](#), [krb5.conf\(4\)](#), [attributes\(5\)](#), [kerberos\(5\)](#)

Name keytables – keyboard table descriptions for loadkeys and dumpkeys

Description These files are used by [loadkeys\(1\)](#) to modify the translation tables used by the keyboard streams module and generated from those translation tables. See [loadkeys\(1\)](#).

Any line in the file beginning with # is a comment, and is ignored. # is treated specially only at the beginning of a line.

Other lines specify the values to load into the tables for a particular keystation. The format is either:

key number list_of_entries

or

swap number1 with number2

or

key number1 same as number2

or a blank line, which is ignored.

key number list_of_entries

sets the entries for keystation *number* from the list given. An entry in that list is of the form

tablename code

where *tablename* is the name of a particular translation table, or a ll. The translation tables are:

<i>base</i>	entry when no shifts are active
<i>shift</i>	entry when "Shift" key is down
<i>caps</i>	entry when "Caps Lock" is in effect
<i>ctrl</i>	entry when "Control" is down
<i>altg</i>	entry when "Alt Graph" is down
<i>numl</i>	entry when "Num Lock" is in effect
<i>up</i>	entry when a key goes up

All tables other than *up* refer to the action generated when a key goes down. Entries in the *up* table are used only for shift keys, since the shift in question goes away when the key goes up, except for keys such as "Caps Lock" or "Num Lock"; the keyboard streams module makes the key look as if it were a latching key.

A table name of *all* indicates that the entry for all tables should be set to the specified value, with the following exception: for entries with a value other than *hole*, the entry for the *numl* table should be set to *nonl*, and the entry for the *up* table should be set to *nop*.

The *code* specifies the effect of the key in question when the specified shift key is down. A *code* consists of either:

- A character, which indicates that the key should generate the given character. The character can either be a single character, a single character preceded by ^ which refers to a "control character" (for instance, ^c is control-C), or a C-style character constant enclosed in single quote characters ('), which can be expressed with C-style escape sequences such as \r for RETURN or \000 for the null character. Note that the single character may be any character in an 8-bit character set, such as ISO 8859/1.
- A string, consisting of a list of characters enclosed in double quote characters ("). Note that the use of the double quote character means that a *code* of double quote must be enclosed in single quotes.

- One of the following expressions:

<code>shiftkeys+leftshift</code>	the key is to be the left-hand "Shift" key
<code>shiftkeys+rightshift</code>	the key is to be the right-hand "Shift" key
<code>shiftkeys+leftctrl</code>	the key is to be the left-hand "Control" key
<code>shiftkeys+rightctrl</code>	the key is to be the right-hand "Control" key
<code>shiftkeys+alt</code>	the key is to be the "Alt" shift key
<code>shiftkeys+altgraph</code>	the key is to be the "Alt Graph" shift key
<code>shiftkeys+capslock</code>	the key is to be the "Caps Lock" key
<code>shiftkeys+shiftlock</code>	the key is to be the "Shift Lock" key
<code>shiftkeys+numlock</code>	the key is to be the "Num Lock" key
<code>buckybits+systembit</code>	the key is to be the "Stop" key in SunView; this is normally the L1 key, or the SETUP key on the VT100 keyboard
<code>buckybits+metabit</code>	the key is to be the "meta" key. That is, the "Left" or "Right" key on a Sun-2 or Sun-3 keyboard or the "diamond" key on a Sun-4 keyboard
<code>compose</code>	the key is to be the "Compose" key
<code>ctrlq</code>	on the "VT100" keyboard, the key is to transmit the control-Q character (this would be the entry for the "Q" key in the <code>ctrl</code> table)
<code>ctrls</code>	on the "VT100" keyboard, the key is to transmit the control-S character (this would be the entry for the "S" key in the <code>ctrl</code> table)

<code>noscroll</code>	on the "VT100" keyboard, the key is to be the "No Scroll" key
<code>string+uparrow</code>	the key is to be the "up arrow" key
<code>string+downarrow</code>	the key is to be the "down arrow" key
<code>string+leftarrow</code>	the key is to be the "left arrow" key
<code>string+rightarrow</code>	the key is to be the "right arrow" key
<code>string+homearrow</code>	the key is to be the "home" key
<code>fa_acute</code>	the key is to be the acute accent "floating accent" key
<code>fa_cedilla</code>	the key is to be the cedilla "floating accent" key
<code>fa_cflex</code>	the key is to be the circumflex "floating accent" key
<code>fa_grave</code>	the key is to be the grave accent "floating accent" key
<code>fa_tilde</code>	the key is to be the tilde "floating accent" key
<code>fa_umlaut</code>	the key is to be the umlaut "floating accent" key
<code>nonl</code>	this is used only in the Num Lock table; the key is not to be affected by the state of Num Lock
<code>pad0</code>	the key is to be the "0" key on the numeric keypad
<code>pad1</code>	the key is to be the "1" key on the numeric keypad
<code>pad2</code>	the key is to be the "2" key on the numeric keypad
<code>pad3</code>	the key is to be the "3" key on the numeric keypad
<code>pad4</code>	the key is to be the "4" key on the numeric keypad
<code>pad5</code>	the key is to be the "5" key on the numeric keypad
<code>pad6</code>	the key is to be the "6" key on the numeric keypad
<code>pad7</code>	the key is to be the "7" key on the numeric keypad
<code>pad8</code>	the key is to be the "8" key on the numeric keypad
<code>pad9</code>	the key is to be the "9" key on the numeric keypad
<code>paddot</code>	the key is to be the "." key on the numeric keypad
<code>padenter</code>	the key is to be the "Enter" key on the numeric keypad
<code>padplus</code>	the key is to be the "+" key on the numeric keypad
<code>padminus</code>	the key is to be the "-" key on the numeric keypad
<code>padstar</code>	the key is to be the "*" key on the numeric keypad

<code>padslash</code>	the key is to be the "/" key on the numeric keypad
<code>padequal</code>	the key is to be the "=" key on the numeric keypad
<code>padsep</code>	the key is to be the "," (separator) key on the numeric keypad
<code>lf(<i>n</i>)</code>	the key is to be the left-hand function key <i>n</i>
<code>rf(<i>n</i>)</code>	the key is to be the right-hand function key <i>n</i>
<code>tf(<i>n</i>)</code>	the key is to be the top function key <i>n</i>
<code>bf(<i>n</i>)</code>	the key is to be the "bottom" function key <i>n</i>
<code>nop</code>	the key is to do nothing
<code>error</code>	this code indicates an internal error; to be used only for keystation 126, and must be used there
<code>idle</code>	this code indicates that the keyboard is idle (that is, has no keys down); to be used only for all entries other than the <code>numl</code> and up table entries for keystation 127, and must be used there
<code>oops</code>	this key exists, but its action is not defined; it has the same effect as <code>nop</code>
<code>reset</code>	this code indicates that the keyboard has just been reset; to be used only for the up table entry for keystation 127, and must be used there.
<code>swap <i>number1</i> with <i>number2</i></code>	exchanges the entries for keystations <i>number1</i> and <i>number2</i> .
<code>key <i>number1</i> same as <i>number2</i></code>	sets the entries for keystation <i>number1</i> to be the same as those for keystation <i>number2</i> . If the file does not specify entries for keystation <i>number2</i> , the entries currently in the translation table are used; if the file does specify entries for keystation <i>number2</i> , those entries are used.

Examples EXAMPLE 1 Example of setting multiple keystations.

The following entry sets keystation 15 to be a "hole" (that is, an entry indicating that there is no keystation 15); sets keystation 30 to do nothing when Alt Graph is down, generate "!" when Shift is down, and generate "1" under all other circumstances; and sets keystation 76 to be the left-hand Control key.

```
key 15  all hole
key 30  base 1 shift ! caps 1 ctrl 1 altg nop
```

EXAMPLE 1 Example of setting multiple keystations. *(Continued)*

```
key 76  all shiftkeys+leftctrl up shiftkeys+leftctrl
```

EXAMPLE 2 Exchange DELETE and BACKSPACE keys

The following entry exchanges the Delete and Back Space keys on the Type 4 keyboard:

```
swap 43 with 66
```

Keystation 43 is normally the Back Space key, and keystation 66 is normally the Delete key.

EXAMPLE 3 Disable CAPS LOCK key

The following entry disables the Caps Lock key on the Type 3 and U.S. Type 4 keyboards:

```
key 119 all nop
```

EXAMPLE 4 Standard translation tables for the U.S. Type 4 keyboard

The following specifies the standard translation tables for the U.S. Type 4 keyboard:

```
key 0   all hole
key 1   all buckybits+systembit up buckybits+systembit
key 2   all hole
key 3   all lf(2)
key 4   all hole
key 5   all tf(1)
key 6   all tf(2)
key 7   all tf(10)
key 8   all tf(3)
key 9   all tf(11)
key 10  all tf(4)
key 11  all tf(12)
key 12  all tf(5)
key 13  all shiftkeys+altgraph up shiftkeys+altgraph
key 14  all tf(6)
key 15  all hole
key 16  all tf(7)
key 17  all tf(8)
key 18  all tf(9)
key 19  all shiftkeys+alt up shiftkeys+alt
key 20  all hole
key 21  all rf(1)
key 22  all rf(2)
key 23  all rf(3)
key 24  all hole
key 25  all lf(3)
key 26  all lf(4)
key 27  all hole
```


EXAMPLE 4 Standard translation tables for the U.S. Type 4 keyboard (Continued)

```

key 28  all hole
key 29  all ^[
key 30  base 1 shift ! caps 1 ctrl 1 altg nop
key 31  base 2 shift @ caps 2 ctrl 2 altg nop
key 32  base 3 shift # caps 3 ctrl 3 altg nop
key 33  base 4 shift $ caps 4 ctrl 4 altg nop
key 34  base 5 shift % caps 5 ctrl 5 altg nop
key 35  base 6 shift ^ caps 6 ctrl 6 altg nop
key 36  base 7 shift & caps 7 ctrl 7 altg nop
key 37  base 8 shift * caps 8 ctrl 8 altg nop
key 38  base 9 shift ( caps 9 ctrl 9 altg nop
key 39  base 0 shift ) caps 0 ctrl 0 altg nop
key 40  base - shift _ caps - ctrl ^_ altg nop
key 41  base = shift + caps = ctrl = altg nop
key 42  base ' shift ~ caps ' ctrl ^^ altg nop
key 43  all '\b'
key 44  all hole
key 45  all rf(4) numl padequal
key 46  all rf(5) numl padslash
key 47  all rf(6) numl padstar
key 48  all bf(13)
key 49  all lf(5)
key 50  all bf(10) numl padequal
key 51  all lf(6)
key 52  all hole
key 53  all '\t'
key 54  base q shift Q caps Q ctrl ^Q altg nop
key 55  base w shift W caps W ctrl ^W altg nop
key 56  base e shift E caps E ctrl ^E altg nop
key 57  base r shift R caps R ctrl ^R altg nop
key 58  base t shift T caps T ctrl ^T altg nop
key 59  base y shift Y caps Y ctrl ^Y altg nop
key 60  base u shift U caps U ctrl ^U altg nop
key 61  base i shift I caps I ctrl '\t' altg nop
key 62  base o shift O caps O ctrl ^O altg nop
key 63  base p shift P caps P ctrl ^P altg nop
key 64  base [ shift { caps [ ctrl ^[ altg nop
key 65  base ] shift } caps ] ctrl ^] altg nop
key 66  all '\177'
key 67  all compose
key 68  all rf(7) numl pad7
key 69  all rf(8) numl pad8
key 70  all rf(9) numl pad9
key 71  all bf(15) numl padminus
key 72  all lf(7)

```

EXAMPLE 4 Standard translation tables for the U.S. Type 4 keyboard (Continued)

```

key 73  all lf(8)
key 74  all hole
key 75  all hole
key 76  all shiftkeys+leftctrl up shiftkeys+leftctrl
key 77  base a shift A caps A ctrl ^A altg nop
key 78  base s shift S caps S ctrl ^S altg nop
key 79  base d shift D caps D ctrl ^D altg nop
key 80  base f shift F caps F ctrl ^F altg nop
key 81  base g shift G caps G ctrl ^G altg nop
key 82  base h shift H caps H ctrl '\b' altg nop
key 83  base j shift J caps J ctrl '\n' altg nop
key 84  base k shift K caps K ctrl '\v' altg nop
key 85  base l shift L caps L ctrl ^L altg nop
key 86  base ; shift : caps ; ctrl ; altg nop
key 87  base '\'' shift '"' caps '\'' ctrl '\'' altg nop
key 88  base '\\\'' shift | caps '\\\'' ctrl ^\ altg nop
key 89  all '\r'
key 90  all bf(11) numl padenter
key 91  all rf(10) numl pad4
key 92  all rf(11) numl pad5
key 93  all rf(12) numl pad6
key 94  all bf(8) numl pad0
key 95  all lf(9)
key 96  all hole
key 97  all lf(10)
key 98  all shiftkeys+numlock
key 99  all shiftkeys+leftshift up shiftkeys+leftshift
key 100 base z shift Z caps Z ctrl ^Z altg nop
key 101 base x shift X caps X ctrl ^X altg nop
key 102 base c shift C caps C ctrl ^C altg nop
key 103 base v shift V caps V ctrl ^V altg nop
key 104 base b shift B caps B ctrl ^B altg nop
key 105 base n shift N caps N ctrl ^N altg nop
key 106 base m shift M caps M ctrl '\r' altg nop
key 107 base , shift < caps , ctrl , altg nop
key 108 base . shift > caps . ctrl . altg nop
key 109 base / shift ? caps / ctrl ^_ altg nop
key 110 all shiftkeys+rightshift up shiftkeys+rightshift
key 111 all '\n'
key 112 all rf(13) numl pad1
key 113 all rf(14) numl pad2
key 114 all rf(15) numl pad3
key 115 all hole
key 116 all hole
key 117 all hole

```

EXAMPLE 4 Standard translation tables for the U.S. Type 4 keyboard *(Continued)*

```
key 118 all lf(16)
key 119 all shiftkeys+capslock
key 120 all buckybits+metabit up buckybits+metabit
key 121 base ' ' shift ' ' caps ' ' ctrl ^@ altg ' '
key 122 all buckybits+metabit up buckybits+metabit
key 123 all hole
key 124 all hole
key 125 all bf(14) numl padplus
key 126 all error numl error up hole
key 127 all idle numl idle up reset
```

See Also [loadkeys\(1\)](#)

Name krb5.conf – Kerberos configuration file

Synopsis /etc/krb5/krb5.conf

Description The `krb5.conf` file contains Kerberos configuration information, including the locations of KDCs and administration daemons for the Kerberos realms of interest, defaults for the current realm and for Kerberos applications, and mappings of host names onto Kerberos realms. This file must reside on all Kerberos clients.

The format of the `krb5.conf` consists of sections headings in square brackets. Each section can contain zero or more configuration variables (called *relations*), of the form:

relation= *relation-value*

or

relation-subsection = {

relation= *relation-value*

relation= *relation-value*

}

The `krb5.conf` file can contain any or all of the following sections:

`libdefaults`

Contains default values used by the Kerberos V5 library.

`appdefaults`

Contains subsections for Kerberos V5 applications, where *relation-subsection* is the name of an application. Each subsection describes application-specific defaults.

`realms`

Contains subsections for Kerberos realms, where *relation-subsection* is the name of a realm. Each subsection contains relations that define the properties for that particular realm.

`domain_realm`

Contains relations which map domain names and subdomains onto Kerberos realm names. This is used by programs to determine what realm a host should be in, given its fully qualified domain name.

`logging`

Contains relations which determine how Kerberos programs are to perform logging.

`capaths`

Contains the authentication paths used with direct (nonhierarchical) cross-realm authentication. Entries in this section are used by the client to determine the intermediate realms which can be used in cross-realm authentication. It is also used by the end-service when checking the transited field for trusted intermediate realms.

dbmodules

Contains relations for Kerberos database plug-in-specific configuration information.

kdc

For a Key Distribution Center (KDC), can contain the location of the `kdc.conf` file.

The `[libdefaults]` Section The `[libdefaults]` section can contain any of the following relations:

database_module

Selects the `dbmodule` section entry to use to access the Kerberos database. If this parameter is not present the code uses the standard db2-based Kerberos database.

default_keytab_name

Specifies the default keytab name to be used by application servers such as `telnetd` and `rlogind`. The default is `/etc/krb5/krb5.keytab`.

default_realm

Identifies the default Kerberos realm for the client. Set its value to your Kerberos realm.

default_tgs_etypes

Identifies the supported list of session key encryption types that should be returned by the KDC. The list can be delimited with commas or whitespace. The supported encryption types by default are `aes256-cts-hmac-sha1-96`, `aes128-cts-hmac-sha1-96`, `des3-cbc-sha1-kd`, and `arcfour-hmac-md5`.

`arcfour-hmac-md5-exp`, `des-cbc-md5`, and `des-cbc-crc` are considered weak encryption types. See the `allow_weak_crypto` relation in this manual page for more information.

default_tkt_etypes

Identifies the supported list of session key encryption types that should be requested by the client. The format is the same as for `default_tgs_etypes`. The supported encryption types by default are `aes256-cts-hmac-sha1-96`, `aes128-cts-hmac-sha1-96`, `des3-cbc-sha1-kd`, and `arcfour-hmac-md5`.

`arcfour-hmac-md5-exp`, `des-cbc-md5`, and `des-cbc-crc` are considered weak encryption types. See the `allow_weak_crypto` relation in this manual page for more information.

permitted_etypes

This relation controls the encryption types for session keys permitted by server applications that use Kerberos for authentication. In addition, it controls the encryption types of keys added to a keytab by means of the `kadmin(1M) ktadd` command. The default is: `aes256-cts-hmac-sha1-96`, `aes128-cts-hmac-sha1-96`, `des3-hmac-sha1-kd` and `arcfour-hmac-md5`.

`arcfour-hmac-md5-exp`, `des-cbc-md5`, and `des-cbc-crc` are considered weak encryption types. See the `allow_weak_crypto` relation in this manual page for more information.

`allow_weak_crypto = [true | false]`

If this is set to false, then weak encryption types are filtered out of the previous three lists.

The default value for this tag is false, which can cause authentication failures in existing Kerberos infrastructures that do not support strong crypto. Users in affected environments should set this tag to true until their infrastructure adopts stronger ciphers.

`clockskew`

Sets the maximum allowable amount of clock skew in seconds that the library tolerates before assuming that a Kerberos message is invalid. The default value is 300 seconds, or five minutes.

`forwardable = [true | false]`

Sets the “forwardable” flag in all tickets. This allows users to transfer their credentials from one host to another without reauthenticating. This option can also be set in the `[appdefaults]` or `[realms]` section (see below) to limit its use in particular applications or just to a specific realm.

`proxiable = [true | false]`

Sets the `proxiable` flag in all tickets. This allows users to create a proxy ticket that can be transferred to a kerberized service to allow that service to perform some function on behalf of the original user. This option can also be set in the `[appdefaults]` or `[realms]` section (see below) to limit its use in particular applications or just to a specific realm.

`renew_lifetime =lifetime`

Requests renewable tickets, with a total lifetime of *lifetime*. The value for *lifetime* must be followed immediately by one of the following delimiters:

s
seconds

m
minutes

h
hours

d
days

Example:

`renew_lifetime = 90m`

Do not mix units. A value of “3h30m” results in an error.

`dns_lookup_kdc`

Indicates whether DNS SRV records need to be used to locate the KDCs and the other servers for a realm, if they have not already been listed in the `[realms]` section. This option makes the machine vulnerable to a certain type of DoS attack if someone spoofs the DNS records and does a redirect to another server. This is, however, no worse than a DoS, since

the bogus KDC is unable to decode anything sent (excepting the initial ticket request, which has no encrypted data). Also, anything the fake KDC sends out is not trusted without verification (the local machine is unaware of the secret key to be used). If `dns_lookup_kdc` is not specified but `dns_fallback` is, then that value is used instead. In either case, values (if present) in the `[realms]` section override DNS. `dns_lookup_kdc` is enabled by default.

`dns_lookup_realm`

Indicates whether DNS TXT records need to be used to determine the Kerberos realm information and/or the host/domain name-to-realm mapping of a host, if this information is not already present in the `krb5.conf` file. Enabling this option might make the host vulnerable to a redirection attack, wherein spoofed DNS replies persuade a client to authenticate to the wrong realm. In a realm with no cross-realm trusts, this is a DoS attack. If `dns_lookup_realm` is not specified but `dns_fallback` is, then that value is used instead. In either case, values (if present) in the `[libdefaults]` and `[domain_realm]` sections override DNS.

`dns_fallback`

Generic flag controlling the use of DNS for retrieval of information about Kerberos servers and host/domain name-to-realm mapping. If both `dns_lookup_kdc` and `dns_lookup_realm` have been specified, this option has no effect.

`verify_ap_req_nofail [true | false]`

If `true`, the local keytab file (`/etc/krb5/krb5.keytab`) must contain an entry for the local host principal, for example, `host/foo.bar.com@FOO.COM`. This entry is needed to verify that the TGT requested was issued by the same KDC that issued the key for the host principal. If undefined, the behavior is as if this option were set to `true`. Setting this value to `false` leaves the system vulnerable to DNS spoofing attacks. This parameter can be in the `[realms]` section to set it on a per-realm basis, or it can be in the `[libdefaults]` section to make it a network-wide setting for all realms.

`ticket_lifetime`

The value of this tag is the default lifetime for initial tickets. The default value for the tag is 1 day (1d).

The `[appdefaults]` Section

This section contains subsections for Kerberos V5 applications, where *relation-subsection* is the name of an application. Each subsection contains relations that define the default behaviors for that application.

The following relations can be found in the `[appdefaults]` section, though not all relations are recognized by all kerberized applications. Some are specific to particular applications.

`auto_login = [true | false]`

Forces the application to attempt automatic login by presenting Kerberos credentials. This is valid for the following applications: `rlogin`, `rsh`, `rcp`, and `telnet`.

`encrypt = [true | false]`

Forces applications to use encryption by default (after authentication) to protect the privacy of the sessions. This is valid for the following applications: `rlogin`, `rsh`, `rcp`, and `telnet`.

`forward = [true | false]`

Forces applications to forward the user's credentials (after authentication) to the remote server. This is valid for the following applications: `rlogin`, `rsh`, `rcp`, and `telnet`.

`forwardable = [true | false]`

See the description in the `[libdefaults]` section above. This is used by any application that creates a ticket granting ticket and also by applications that can forward tickets to a remote server.

`proxiabile = [true | false]`

See the description in the `[libdefaults]` section above. This is used by any application that creates a ticket granting ticket.

`renewable = [true | false]`

Creates a TGT that can be renewed (prior to the ticket expiration time). This is used by any application that creates a ticket granting ticket.

`no_addresses = [true | false]`

Creates tickets with no address bindings. This is to allow tickets to be used across a NAT boundary or when using multi-homed systems. This option is valid in the `kinit [appdefault]` section only.

`rcmd_protocol = [rcmdv1 | rcmdv2]`

Specifies which Kerberized “`rcmd`” protocol to use when using the Kerberized `rlogin(1)`, `rsh(1)`, or `rcp(1)` programs. The default is to use `rcmdv2` by default, as this is the more secure and more recent update of the protocol. However, when talking to older MIT or SEAM-based “`rcmd`” servers, it can be necessary to force the new clients to use the older `rcmdv1` protocol. This option is valid only for the following applications: `rlogin`, `rcp`, and `rsh`.

The following application defaults can be set to `true` or `false`:

`kinit`

```
forwardable = true
proxiabile = true
renewable = true
no_addresses = true
```

See `kinit(1)` for the valid time duration formats you can specify for `delta_time`.

In the following example, `kinit` gets forwardable tickets by default and `telnet` has three default behaviors specified:

```
[appdefaults]
kinit = {
```



```

        forwardable = true
    }

    telnet = {
        forward = true
        encrypt = true
        autologin = true
    }

```

The application defaults specified here are overridden by those specified in the `[realms]` section.

The `[realms]` Section This section contains subsections for Kerberos realms, where *relation-subsection* is the name of a realm. Each subsection contains relations that define the properties for that particular realm. The following relations can be specified in each `[realms]` subsection:

`admin_server`

Identifies the host where the Kerberos administration daemon (`kadmind`) is running. Typically, this is the master KDC.

application defaults

Application defaults that are specific to a particular realm can be specified within a `[realms]` subsection. Realm-specific application defaults override the global defaults specified in the `[appdefaults]` section.

`auth_to_local_realm`

For use in the default realm, non-default realms can be equated with the default realm for authenticated name-to-local name mapping.

`auth_to_local_names`

This subsection allows you to set explicit mappings from principal names to local user names. The tag is the mapping name and the value is the corresponding local user name.

`auth_to_local`

This tag allows you to set a general rule for mapping principal names to local user names. It is used if there is not an explicit mapping for the principal name that is being translated. The possible values are:

```
RULE: [ncomps]:<format>(<regex>)s/<regex>/<text>/
```

Each rule has three parts:

First part—Formulate the string on which to perform operations:

If not present then the string defaults to the fully flattened principal minus the realm name. Otherwise the syntax is as follows:

```
"[" <ncomps> ":" <format> "]"
```

Where:

`<ncomps>` is the number of expected components for this rule. If the particular principal does not have this number of components, then this rule does not apply.

`<format>` is a string of `<component>` or verbatim characters to be inserted.

`<component>` is of the form “\$”`<number>` to select the `<number>`th component. `<number>` begins from 1.

Second part—select rule validity:

If not present, this rule can apply to all selections. Otherwise the syntax is as follows:

```
"(" <regex> ")"
```

Where:

`<regex>` is a selector regular expression. If this regular expression matches the whole pattern generated from the first part, then this rule still applies.

Third part—Transform rule:

If not present, then the selection string is passed verbatim and is matched. Otherwise, the syntax is as follows:

```
<rule> . . .
```

Where:

`<rule>` is of the form:

```
"s/" <regex> "/" <text> "/" ["g"]
```

Regular expressions are defined in [regex\(5\)](#).

For example:

```
auth_to_local = RULE:[1:$1@$0](.*@.*ACME\.COM)s/@.*//
```

The preceding maps `username@ACME.COM` and all sub-realms of `ACME.COM` to `username`.

DEFAULT

The principal name is used as the local name. If the principal has more than one component or is not in the default realm, this rule is not applicable and the conversion fails.

database_module

Selects the `dbmodule` section entry to use to access the Kerberos database.

extra_addresses...

This allows a computer to use multiple local addresses, to allow Kerberos to work in a network that uses NATs. The addresses should be in a comma-separated list.

kdc

The name of a host running a KDC for that realm. An optional port number (separated from the hostname by a colon) can be included.

kpasswd_server

Identifies the host where the Kerberos password-changing server is running. Typically, this is the same as host indicated in the `admin_server`. If this parameter is omitted, the host in `admin_server` is used. You can also specify a port number if the server indicated by `kpasswd_server` runs on a port other than 464 (the default). The format of this parameter is: `hostname[:port]`.

kpasswd_protocol

Identifies the protocol to be used when communicating with the server indicated by `kpasswd_server`. By default, this parameter is defined to be `RPCSEC_GSS`, which is the protocol used by Solaris-based administration servers. To be able to change a principal's password stored on non-Solaris Kerberos server, such as Microsoft Active Directory or MIT Kerberos, this value should be `SET_CHANGE`. This indicates that a non-RPC-based protocol is used to communicate the password change request to the server in the `kpasswd_server` entry.

udp_preference_limit

When sending a message to the KDC, the library tries using TCP before UDP if the size of the message is above `udp_preference_limit`. If the message is smaller than `udp_preference_limit`, then UDP is tried before TCP. Regardless of the size, both protocols are tried if the first attempt fails.

verify_ap_req_nofail [true | false]

If `true`, the local keytab file (`/etc/krb5/krb5.keytab`) must contain an entry for the local host principal, for example, `host/foo.bar.com@FOO.COM`. This entry is needed to verify that the TGT requested was issued by the same KDC that issued the key for the host principal. If undefined, the behavior is as if this option were set to `true`. Setting this value to `false` leaves the system vulnerable to DNS spoofing attacks. This parameter might be in the `[realms]` section to set it on a per-realm basis, or it might be in the `[libdefaults]` section to make it a network-wide setting for all realms.

The parameters “forwardable”, “proxiabile”, and “renew_lifetime” as described in the `[libdefaults]` section (see above) are also valid in the `[realms]` section.

Notice that `kpasswd_server` and `kpasswd_protocol` are realm-specific parameters. Most often, you need to specify them only when using a non-Solaris-based Kerberos server. Otherwise, the change request is sent over `RPCSEC_GSS` to the Solaris Kerberos administration server.

The `[domain_realm]`
Section

This section provides a translation from a domain name or hostname to a Kerberos realm name. The *relation* can be a host name, or a domain name, where domain names are indicated by a period (‘.’) prefix. *relation-value* is the Kerberos realm name for that particular host or domain. Host names and domain names should be in lower case.

If no translation entry applies, the host's realm is considered to be the hostname's domain portion converted to upper case. For example, the following `[domain_realm]` section maps `crash.mit.edu` into the `TEST.ATHENA.MIT.EDU` realm:

```
[domain_realm]
    .mit.edu = ATHENA.MIT.EDU
    mit.edu = ATHENA.MIT.EDU
    crash.mit.edu = TEST.ATHENA.MIT.EDU
    .fubar.org = FUBAR.ORG
    fubar.org = FUBAR.ORG
```

All other hosts in the `mit.edu` domain maps by default to the `ATHENA.MIT.EDU` realm, and all hosts in the `fubar.org` domain maps by default into the `FUBAR.ORG` realm. The entries for the hosts `mit.edu` and `fubar.org`. Without these entries, these hosts would be mapped into the Kerberos realms `EDU` and `ORG`, respectively.

The `[logging]` Section This section indicates how Kerberos programs are to perform logging. There are two types of relations for this section: relations to specify how to log and a relation to specify how to rotate `kdc` log files.

The following relations can be defined to specify how to log. The same relation can be repeated if you want to assign it multiple logging methods.

`admin_server`

Specifies how to log the Kerberos administration daemon (`kadmin`). The default is `FILE:/var/krb5/kadmin.log`.

`default`

Specifies how to perform logging in the absence of explicit specifications otherwise.

`kdc`

Specifies how the KDC is to perform its logging. The default is `FILE:/var/krb5/kdc.log`.

The `admin_server`, `default`, and `kdc` relations can have the following values:

`FILE:filename`

`FILE=filename`

This value causes the entity's logging messages to go to the specified file. If the '=' form is used, the file is overwritten. If the ':' form is used, the file is appended to.

`STDERR`

This value causes the entity's logging messages to go to its standard error stream.

`CONSOLE`

This value causes the entity's logging messages to go to the console, if the system supports it.

`DEVICE=devicename`

This causes the entity's logging messages to go to the specified device.

`SYSLOG[:severity[:facility]]`

This causes the entity's logging messages to go to the system log.

The *severity* argument specifies the default severity of system log messages. This can be any of the following severities supported by the `syslog(3C)` call, minus the `LOG_` prefix: `LOG_EMERG`, `LOG_ALERT`, `LOG_CRIT`, `LOG_ERR`, `LOG_WARNING`, `LOG_NOTICE`, `LOG_INFO`, and `LOG_DEBUG`. For example, a value of `CRIT` would specify `LOG_CRIT` severity.

The *facility* argument specifies the facility under which the messages are logged. This can be any of the following facilities supported by the `syslog(3C)` call minus the `LOG_` prefix: `LOG_KERN`, `LOG_USER`, `LOG_MAIL`, `LOG_DAEMON`, `LOG_AUTH`, `LOG_LPR`, `LOG_NEWS`, `LOG_UUCP`, `LOG_CRON`, and `LOG_LOCAL0` through `LOG_LOCAL7`.

If no severity is specified, the default is `ERR`. If no facility is specified, the default is `AUTH`.

The following relation can be defined to specify how to rotate `kadmin` and `kdc` log files if the `FILE: value` is being used to log:

```
admin_server_rotate
```

```
kdc_rotate
```

A relation subsection that enables `kadmin` (`admin_server_rotate`) and/or `kdc` (`kdc_rotate`) logging to be rotated to multiple files based on a time interval. This can be used to avoid logging to one file, which might grow too large and bring the KDC to a halt.

The time interval for the rotation is specified by the `period` relation. The number of log files to be rotated is specified by the `versions` relation. Both the `period` and `versions` (described below) should be included in this subsection. And, this subsection applies only if the `kdc` relation has a `FILE: value`.

The following relations can be specified for the `kdc_rotate` relation subsection:

```
period=delta_time
```

Specifies the time interval before a new log file is created. See the `TimeFormats` section in `kinit(1)` for the valid time duration formats you can specify for *delta_time*. If `period` is not specified or set to `never`, no rotation occurs.

Specifying a time interval does not mean that the log files are rotated at the time interval based on real time. This is because the time interval is checked at each attempt to write a record to the log, or when logging is actually occurring. Therefore, rotation occurs only when logging has actually occurred for the specified time interval.

```
versions=number
```

Specifies how many previous versions are saved before the rotation begins. A number is appended to the log file, starting with 0 and ending with (*number* - 1). For example, if `versions` is set to 2, up to three logging files are created (*filename*, *filename.0*, and *filename.1*) before the first one is overwritten to begin the rotation.

Notice that if `versions` is not specified or set to 0, only one log file is created, but it is overwritten whenever the time interval is met.

In the following example, the logging messages from the Kerberos administration daemon goes to the console. The logging messages from the KDC is appended to the `/var/krb5/kdc.log`, which is rotated between twenty-one log files with a specified time interval of a day.

```
[logging]
  admin_server = CONSOLE
  kdc = FILE:/export/logging/kadmin.log
  kdc_rotate = {
    period = 1d
    versions = 20
  }
```

The `[capaths]` Section In order to perform direct (non-hierarchical) cross-realm authentication, a database is needed to construct the authentication paths between the realms. This section defines that database.

A client uses this section to find the authentication path between its realm and the realm of the server. The server uses this section to verify the authentication path used by the client, by checking the transited field of the received ticket.

There is a subsection for each participating realm, and each subsection has relations named for each of the realms. The *relation-value* is an intermediate realm which can participate in the cross-realm authentication. The relations can be repeated if there is more than one intermediate realm. A value of '.' means that the two realms share keys directly, and no intermediate realms should be allowed to participate.

There are $n \times 2$ possible entries in this table, but only those entries which is needed on the client or the server need to be present. The client needs a subsection named for its local realm, with relations named for all the realms of servers it needs to authenticate with. A server needs a subsection named for each realm of the clients it serves.

For example, ANL.GOV, PNL.GOV, and NERSC.GOV all wish to use the ES.NET realm as an intermediate realm. ANL has a sub realm of TEST.ANL.GOV, which authenticates with NERSC.GOV but not PNL.GOV. The `[capath]` section for ANL.GOV systems would look like this:

```
[capaths]
  ANL.GOV = {
    TEST.ANL.GOV = .
    PNL.GOV = ES.NET
    NERSC.GOV = ES.NET
    ES.NET = .
  }

  TEST.ANL.GOV = {
    ANL.GOV = .
  }

  PNL.GOV = {
```

```

    ANL.GOV = ES.NET
}

NERSC.GOV = {
    ANL.GOV = ES.NET
}

ES.NET = {
    ANL.GOV = .
}

```

The [capath] section of the configuration file used on NERSC.GOV systems would look like this:

```

[capaths]
  NERSC.GOV = {
    ANL.GOV = ES.NET
    TEST.ANL.GOV = ES.NET
    TEST.ANL.GOV = ANL.GOV
    PNL.GOV = ES.NET
    ES.NET = .
  }

  ANL.GOV = {
    NERSC.GOV = ES.NET
  }

  PNL.GOV = {
    NERSC.GOV = ES.NET
  }

  ES.NET = {
    NERSC.GOV = .
  }

  TEST.ANL.GOV = {
    NERSC.GOV = ANL.GOV
    NERSC.GOV = ES.NET
  }

```

In the above examples, the ordering is not important, except when the same relation is used more than once. The client uses this to determine the path. (It is not important to the server, since the transited field is not sorted.)

PKINIT-specific Options The following are `pkinit`-specific options. These values can be specified in [libdefaults] as global defaults, or within a realm-specific subsection of [libdefaults], or can be specified as realm-specific values in the [realms] section. A realm-specific value overrides, does not add to, a generic [libdefaults] specification.

The search order is:

1. realm-specific subsection of [libdefaults]

```
[libdefaults]
    EXAMPLE.COM = {
        pkinit_anchors = FILE:/usr/local/example.com.crt
```

2. realm-specific value in the [realms] section

```
[realms]
    OTHERREALM.ORG = {
        pkinit_anchors = FILE:/usr/local/otherrealm.org.crt
```

3. generic value in the [libdefaults] section

```
[libdefaults]
    pkinit_anchors = DIR:/usr/local/generic_trusted_cas/
```

The syntax for specifying Public Key identity, trust, and revocation information for pkinit is as follows:

`pkinit_identities = URI`

Specifies the location(s) to be used to find the user's X.509 identity information. This option can be specified multiple times. Each value is attempted in order until identity information is found and authentication is attempted. These values are not used if the user specifies `X509_user_identity` on the command line.

Valid *URI* types are FILE, DIR, PKCS11, PKCS12, and ENV. See the PKINIT URI Types section for more details.

`pkinit_anchors = URI`

Specifies the location of trusted anchor (root) certificates which the client trusts to sign KDC certificates. This option can be specified multiple times. These values from the config file are not used if the user specifies `X509_anchors` on the command line.

Valid *URI* types are FILE and DIR. See the PKINIT URI Types section for more details.

`pkinit_pool = URI`

Specifies the location of intermediate certificates which can be used by the client to complete the trust chain between a KDC certificate and a trusted anchor. This option can be specified multiple times.

<code>pkinit_revoke = URI</code>	<p>Valid <i>URI</i> types are FILE and DIR. See the PKINIT URI Types section for more details.</p> <p>Specifies the location of Certificate Revocation List (CRL) information to be used by the client when verifying the validity of the KDC certificate presented. This option can be specified multiple times.</p> <p>The only valid <i>URI</i> type is DIR. See the PKINIT URI Types section for more details.</p>
<code>pkinit_require_crl_checking = value</code>	<p>The default certificate verification process always checks the available revocation information to see if a certificate has been revoked. If a match is found for the certificate in a CRL, verification fails. If the certificate being verified is not listed in a CRL, or there is no CRL present for its issuing CA, and <code>pkinit_require_crl_checking</code> is <code>false</code>, then verification succeeds. However, if <code>pkinit_require_crl_checking</code> is <code>true</code> and there is no CRL information available for the issuing CA, then verification fails. <code>pkinit_require_crl_checking</code> should be set to <code>true</code> if the policy is such that up-to-date CRLs must be present for every CA.</p>
<code>pkinit_dh_min_bits = value</code>	<p>Specifies the size of the Diffie-Hellman key the client attempts to use. The acceptable values are currently 1024, 2048, and 4096. The default is 2048.</p>
<code>pkinit_win2k = value</code>	<p>This flag specifies whether the target realm is assumed to support only the old, pre-RFC version of the protocol. The default is <code>false</code>.</p>
<code>pkinit_win2k_require_binding = value</code>	<p>If this flag is set to <code>true</code>, it expects that the target KDC is patched to return a reply with a checksum rather than a nonce. The default is <code>false</code>.</p>
<code>pkinit_eku_checking = value</code>	<p>This option specifies what Extended Key Usage value the KDC certificate presented to the client must contain. If the KDC certificate has the <code>pkinit SubjectAlternativeName</code> encoded as the Kerberos TGS name, EKU checking is not</p>

necessary since the issuing CA has certified this as a KDC certificate. The values recognized in the `krb5.conf` file are:

<code>kpKDC</code>	This is the default value and specifies that the KDC must have the <code>id-pkinit-KPKdc</code> EKU as defined in RFC4556.
<code>kpServerAuth</code>	If <code>kpServerAuth</code> is specified, a KDC certificate with the <code>id-kp-serverAuth</code> EKU as used by Microsoft is accepted.
<code>none</code>	If <code>none</code> is specified, then the KDC certificate is not checked to verify it has an acceptable EKU. The use of this option is not recommended.

`pkinit_kdc_hostname = value`

The presence of this option indicates that the client is willing to accept a KDC certificate with a `dnsName SAN` (Subject Alternative Name) rather than requiring the `id-pkinit-san` as defined in RFC4556. This option can be specified multiple times. Its value should contain the acceptable hostname for the KDC (as contained in its certificate).

`pkinit_cert_match = rule`

Specifies matching rules that the client certificate must match before it is used to attempt `pkinit` authentication. If a user has multiple certificates available (on a smart card, or by way of another media), there must be exactly one certificate chosen before attempting `pkinit` authentication. This option can be specified multiple times. All the available certificates are checked against each rule in order until there is a match of exactly one certificate.

The Subject and Issuer comparison strings are the RFC2253 string representations from the certificate Subject DN and Issuer DN values.

The syntax of the matching rules is:

```
[relation-operator]component-rule '...'
```

where

relation-operator Specify *relation-operator* as &&, meaning all component rules must match, or ||, meaning only one component rule must match. If *relation-operator* is not specified, the default is &&.

component-rule There is no punctuation or white space between component rules. Specify *component-rule* as one of the following:

'<SUBJECT>' regular-expression

'<ISSUER>' regular-expression

'<SAN>' regular-expression

'<EKU>' extended-key-usage-list

where extended-key-usage-list is a comma-separated list of required Extended Key Usage values. All values in the list must be present in the certificate.

```
'pkinit'
'msScLogin'
'clientAuth'
'emailProtection'
```

'<KU>' key-usage-list

where key-usage-list is a comma-separated list of required Key Usage values. All values in the list must be present in the certificate.

```
'digitalSignature'
```

Examples:

```
pkinit_cert_match = ||<SUBJECT>.*DoE.*<SAN>.*@EXAMPLE.COM
pkinit_cert_match = &&<EKU>msScLogin,clientAuth<ISSUER>.*DoE.*
pkinit_cert_match = <EKU>msScLogin,clientAuth<KU>digitalSignature
```

PKINIT URI Types `FILE:file-name[,key-file-name]`

This option has context-specific behavior.

`pkinit_identities` *file-name* specifies the name of a PEM-format file containing the user's certificate. If *key-file-name* is not specified, the user's private key is expected to be in *file-name* as well. Otherwise, *key-file-name* is the name of the file containing the private key.

`pkinit_anchors`
`pkinit_pool` *file-name* is assumed to be the name of an OpenSSL-style ca-bundle file. The ca-bundle file should be base-64 encoded.

DIR:*directory-name*

This option has context-specific behavior.

`pkinit_identities` *directory-name* specifies a directory with files named *.crt and *.key, where the first part of the file name is the same for matching pairs of certificate and private key files. When a file with a name ending with .crt is found, a matching file ending with .key is assumed to contain the private key. If no such file is found, then the certificate in the .crt is not used.

`pkinit_anchors`
`pkinit_pool` *directory-name* is assumed to be an OpenSSL-style hashed CA directory where each CA cert is stored in a file named hash-of-ca-cert.#. This infrastructure is encouraged, but all files in the directory is examined and if they contain certificates (in PEM format), they are used.

PKCS12:*pkcs12-file-name*

pkcs12-file-name is the name of a PKCS #12 format file, containing the user's certificate and private key.

PKCS11: [slotid=*slot-id*] [:token=*token-label*] [:certid=*cert-id*] [:certlabel=*cert-label*]

All keyword/values are optional. PKCS11 modules (for example, `opensc-pkcs11.so`) must be installed as a crypto provider under `libpkcs11(3LIB)`. `slotid=` and/or `token=` can be specified to force the use of a particular smart card reader or token if there is more than one available. `certid=` and/or `certlabel=` can be specified to force the selection of a particular certificate on the device. See the `pkinit_cert_match` configuration option for more ways to select a particular certificate to use for `pkinit`.

ENV:*environment-variable-name*

environment-variable-name specifies the name of an environment variable which has been set to a value conforming to one of the previous values. For example, `ENV:X509_PROXY`, where environment variable `X509_PROXY` has been set to `FILE:/tmp/my_proxy.pem`.

The [dbmodules] Section This section consists of relations that provide configuration information for plug-in modules. In particular, the relations describe the configuration for LDAP KDB plug-in. Use of the db2 KDB plug-in is the default behavior and that this section does not need to be filled out in that case.

db_library

Name of the plug-in library. To use the LDAP KDB plug-in the name must be `kdb_ldap`. The default value is `db2`.

db_module_dir

Path to the plug-in libraries. The default is `/usr/lib/krb5`.

ldap_cert_path

Path to the Network Security Services (NSS) trusted database for an SSL connection. This is a required parameter when using the LDAP KDB plug-in.

ldap_conns_per_server

Number of connections per LDAP instance. The default is 5.

ldap_kadmin_dn

Bind DN for `kadmin`. This specifies the DN that the `kadmin` service uses when binding to the LDAP Directory Server. The password for this bind DN should be in the `ldap_service_password_file`.

ldap_kdc_dn

Bind DN for a Key Distribution Center (KDC). This specifies the DN that the `krb5kdc` service use when binding to the LDAP Directory Server. The password for this bind DN should be in the `ldap_service_password_file`.

ldap_servers

List of LDAP directory servers in URI format. Use of either of the following is acceptable.

```
ldap://<ds hostname>:<SSL port>
```

```
ldap://<ds hostname>
```

Each server URI should be separated by whitespace.

ldap_service_password_file

File containing stashed passwords used by the KDC when binding to the LDAP Directory Server. The default is `/var/krb5/service_passwd`. This file is created using `kdb5_ldap_util(1M)`.

ldap_ssl_port

Port number for SSL connection with directory server. The default is 389.

Examples EXAMPLE 1 Sample File

The following is an example of a generic `krb5.conf` file:

```
[libdefaults]
    default_realm = ATHENA.MIT.EDU
    default_tkt_enctypes = aes256-cts-hmac-sha1-96
```

EXAMPLE 1 SampleFile (Continued)

```
default_tgs_enctypes = aes256-cts-hmac-sha1-96

[realms]
  ATHENA.MIT.EDU = {
    kdc = kerberos.mit.edu
    kdc = kerberos-1.mit.edu
    kdc = kerberos-2.mit.edu
    admin_server = kerberos.mit.edu
    auth_to_local_realm = KRBDEV.ATHENA.MIT.EDU
  }

  FUBAR.ORG = {
    kdc = kerberos.fubar.org
    kdc = kerberos-1.fubar.org
    admin_server = kerberos.fubar.org
  }

[domain_realm]
  .mit.edu = ATHENA.MIT.EDU
  mit.edu = ATHENA.MIT.EDU
```

EXAMPLE 2 KDC Using the LDAP KDB plug-in, realms and dbmodules Sections

The following is an example of the realms and dbmodules sections of a Kerberos configuration file when the KDC is using the LDAP KDB plug-in.

```
[realms]
  SUN.COM = {
    kdc = kc-umpk-01.athena.mit.edu
    kdc = kc-umpk-02.athena.mit.edu
    admin_server = kc-umpk-01.athena.mit.edu
    database_module = LDAP
  }

[dbmodules]
  LDAP = {
    db_library = kdb_ldap
    ldap_kerberos_container_dn = "cn=krbcontainer,dc=mit,dc=edu"
    ldap_kdc_dn = "cn=kdc service,ou=profile,dc=mit,dc=edu"
    ldap_kadmin_dn = "cn=kadmin service,ou=profile,dc=mit,dc=edu"
    ldap_cert_path = /var/ldap
    ldap_servers = ldaps://ds.mit.edu
  }
```

Files /var/krb5/kdc.log
KDC logging file

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	See below.

All of the keywords are Committed, except for the PKINIT keywords, which are Volatile.

See Also [kinit\(1\)](#), [rcp\(1\)](#), [rlogin\(1\)](#), [rsh\(1\)](#), [telnet\(1\)](#), [syslog\(3C\)](#), [attributes\(5\)](#), [kerberos\(5\)](#), [regex\(5\)](#)

Notes If the `krb5.conf` file is not formatted properly, the `telnet` command fails. However, the `dtlogin` and `login` commands still succeed, even if the `krb5.conf` file is specified as required for the commands. If this occurs, the following error message is displayed:

```
Error initializing krb5: Improper format of item
```

To bypass any other problems that might occur, you should fix the file as soon as possible.

Name label_encodings – label encodings file

Synopsis /etc/security/tsol/label_encodings

Description The `label_encodings` file is a standard encodings file of security labels that are used to control the conversion of human-readable labels into an internal format, the conversion from the internal format to a human-readable canonical form, and the construction of banner pages for printed output. On a Solaris Trusted Extensions system, the `label_encodings` file is protected at the label `admin_high`. The file should be edited and checked by the security administrator using the Check Label Encodings action in the System_Admin folder in the Application Manager.

In addition to the required sections of the label encodings file that are described in [Compartmented Mode Workstation Labeling: Encodings Format](#), a Solaris Trusted Extensions system accepts optional local extensions. These extensions provide various translation options and an association between character-coded color names and sensitivity labels.

The optional local extensions section starts with the LOCAL DEFINITIONS: keyword and is followed by zero or more of the following unordered statements:

DEFAULT USER SENSITIVITY LABEL= *sensitivity label*

This option specifies the sensitivity label to use as the user's minimum sensitivity label if none is defined for the user in the administrative databases. The default value is the MINIMUM SENSITIVITY LABEL= value from the ACCREDITATION RANGE: section of the label encodings file.

DEFAULT USER CLEARANCE= *clearance*

This option specifies the clearance to use as the user's clearance if none is defined for the user in the administrative databases. The default value is the MINIMUM CLEARANCE= value from the ACCREDITATION RANGE: section of the label encodings file.

The final part of the LOCAL DEFINITIONS: section defines the character-coded color names to be associated with various words, sensitivity labels, or classifications. This section supports the [str_to_label\(3TSOL\)](#) function. It consists of the COLOR NAMES: keyword and is followed by zero or more color-to-label assignments. Each statement has one of the following two syntaxes:

`word= word value; color= color value;`

`label= label value; color= color value;`

where *color value* is a character-coded color name to be associated with the word *word value*, or with the sensitivity label *label value*, or with the classification *label value*.

The character-coded color name *color value* for a label is determined by the order of entries in the COLOR NAMES: section that make up the label. If a label contains a word *word value* that is specified in this section, the *color value* of the label is the one associated with the first *word value* specified. If no specified word *word value* is contained in the label, the *color value* is the

one associated with an exact match of a *label value*. If there is no exact match, the *color value* is the one associated with the first specified *label value* whose classification matches the classification of the label.

Examples EXAMPLE 1 A Sample LOCAL DEFINITIONS: Section

```
LOCAL DEFINITIONS:

DEFAULT USER SENSITIVITY LABEL= C A;
DEFAULT USER CLEARANCE LABEL= S ABLE;

COLOR NAMES:

label= Admin_Low;    color= Pale Blue;
label= unclassified; color= light grey;
word= Project A;    color= bright blue;
label= c;            color= sea foam green;
label= secret;      color= #ff0000;      * Hexadecimal RGB value
word= Hotel;        color= Lavender;
word= KelO;         color= red;
label= TS;          color= khaki;
label= TS Elephant; color= yellow;
label= Admin_High;  color= shocking pink;
```

Files /etc/security/tso1/label_encodings

The label encodings file contains the classification names, words, constraints, and values for the defined labels of this system. It is protected at the label admin_high.

Diagnostics The following diagnostics are in addition to those found in Appendix A of *Compartmented Mode Workstation Labeling: Encodings Format*:

Can't allocate NNN bytes for color names table.

The system cannot dynamically allocate the memory it needs to process the COLOR NAMES: section.

Can't allocate NNN bytes for color table entry.

The system cannot dynamically allocate the memory it needs to process a Color Table entry.

Can't allocate NNN bytes for color word entry.

The system cannot dynamically allocate the memory it needs to process a Color Word entry.

Can't allocate NNN bytes for DEFAULT USER CLEARANCE.

The system cannot dynamically allocate the memory it needs to process the DEFAULT USER CLEARANCE.

Can't allocate NNN bytes for DEFAULT USER SENSITIVITY LABEL.

The system cannot dynamically allocate the memory it needs to process the DEFAULT USER SENSITIVITY LABEL.

DEFAULT USER CLEARANCE= XXX is not in canonical form. Is YYY what is intended?

This error occurs if the clearance specified, while understood, is not in canonical form. This additional canonicalization check ensures that no errors are made in specifying the clearance.

DEFAULT USER SENSITIVITY LABEL= XXX is not in canonical form. Is YYY what is intended?

This error occurs if a sensitivity label specified, while understood, is not in canonical form. This additional canonicalization check ensures that no errors are made in specifying the sensitivity label.

Duplicate DEFAULT USER CLEARANCE= ignored.

More than one DEFAULT USER CLEARANCE= option was encountered. All but the first are ignored.

Duplicate DEFAULT USER SENSITIVITY LABEL= ignored.

More than one DEFAULT USER SENSITIVITY LABEL= option was encountered. All but the first are ignored.

End of File not found where expected. Found instead: XXX.

The noted extraneous text was found when the end of label encodings file was expected.

End of File or LOCAL DEFINITIONS: not found. Found instead: XXX.

The noted extraneous text was found when the LOCAL DEFINITIONS: section or end of label encodings file was expected.

Found color XXX without associated label.

The color XXX was found, however it had no label or word associated with it.

Invalid color label XXX.

The label XXX cannot be parsed.

Invalid DEFAULT USER CLEARANCE XXX.

The DEFAULT USER CLEARANCE XXX cannot be parsed.

Invalid DEFAULT USER SENSITIVITY LABEL XXX.

The DEFAULT USER SENSITIVITY LABEL XXX cannot be parsed.

Label preceding XXX did not have a color specification.

A label or word was found without a matching color name.

Word XXX not found as a valid Sensitivity Label word.

The word XXX was not found as a valid word for a sensitivity label.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/trusted/trusted-global-zone

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	See NOTES.

See Also `chk_encodings(1M)`, `label_to_str(3TSOL)`, `str_to_label(3TSOL)`, `attributes(5)`, `labels(5)`

Oracle Solaris Trusted Extensions Label Administration

Defense Intelligence Agency document DDS-2600-6216-93, *Compartmented Mode Workstation Labeling: Encodings Format*, September 1993.

Warnings Creation of and modification to the label encodings file should only be undertaken with a thorough understanding not only of the concepts in *Compartmented Mode Workstation Labeling: Encodings Format*, but also of the details of the local labeling requirements.

The following warnings are paraphrased from *Compartmented Mode Workstation Labeling: Encodings Format*.

Take extreme care when modifying a label encodings file that is already loaded and running on a Solaris Trusted Extensions system. Once the system runs with the label encodings file, many objects are labeled with sensitivity labels that are well formed with respect to the loaded label encodings file. If the label encodings file is subsequently changed, it is possible that the existing labels will no longer be well-formed. Changing the bit patterns associated with words causes existing objects whose labels contain the words to have possibly invalid labels. Raising the minimum classification or lowering the maximum classification that is associated with words will likely cause existing objects whose labels contain the words to no longer be well-formed.

Changes to a current encodings file that has already been used should be limited only to adding new classifications or words, changing the names of existing words, or modifying the local extensions. As described in *Compartmented Mode Workstation Labeling: Encodings Format*, it is important to reserve extra inverse bits when the label encodings file is first created to allow for later expansion of the label encodings file to incorporate new inverse words. If an inverse word is added that does not use reserved inverse bits, all existing objects on the system will erroneously have labels that include the new inverse word.

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 and might be meaningful only for the DIA MAC policy. This file might not be applicable to other Mandatory policies that might be developed for future releases of Solaris Trusted Extensions software. Parts of it are obsolete and retained for ease of porting. The obsolete parts might be removed in a future Solaris Trusted Extensions release.

Parts of the `label_encodings` file are considered standard and are controlled by Defense Intelligence Agency document DDS-2600-6216-93, *Compartmented Mode Workstation Labeling: Encodings Format*, September 1993. Of that standard, the parts that refer to the `INFORMATION LABELS:` and `NAME INFORMATION LABELS:` sections are Obsolete. However, the `INFORMATION LABELS:` section must be present and syntactically correct. It is ignored. The `NAME INFORMATION LABELS:` section is optional. If present, it is ignored but must be syntactically correct.

Defining the label encodings file is a three-step process. First, the set of human-readable labels to be represented must be identified and understood. The definition of this set includes the list of classifications and other words that are used in the human-readable labels, relations between and among the words, classification restrictions that are associated with use of each word, and intended use of the words in mandatory access control and labeling system output. Next, this definition is associated with an internal format of integers, bit patterns, and logical relationship statements. Finally, a label encodings file is created. The *Compartmented Mode Workstation Labeling: Encodings Format* document describes the second and third steps, and assumes that the first has already been performed.

The following values in the optional `LOCAL DEFINITIONS:` section are obsolete. These values might only affect the obsolete `bltos(3TSOL)` functions, and might be ignored by the `label_to_str(3TSOL)` replacement function:

```
ADMIN LOW NAME=  
ADMIN HIGH NAME=  
DEFAULT LABEL VIEW IS EXTERNAL  
DEFAULT LABEL VIEW IS INTERNAL  
DEFAULT FLAGS=  
FORCED FLAGS=  
CLASSIFICATION NAME=  
COMPARTMENTS NAME=
```

Name ldapfilter.conf – configuration file for LDAP filtering routines

Synopsis /etc/opt/SUNWconn/ldap/current/ldapfilter.conf

Description The `ldapfilter.conf` file contains information used by the LDAP filtering routines.

Blank lines and lines that begin with a hash character (#) are treated as comments and ignored. The configuration information consists of lines that contain one to five tokens. Tokens are separated by white space, and double quotes can be used to include white space inside a token.

The file consists of a sequence of one or more filter sets. A filter set begins with a line containing a single token called a *tag*.

The filter set consists of a sequence of one or more filter lists. The first line in a filter list must contain four or five tokens: the *value pattern*, the *delimiter list*, a *filtertemplate*, a *match description*, and an optional *search scope*. The *value pattern* is a regular expression that is matched against the value passed to the LDAP library call to select the filter list.

The *delimiter list* is a list of the characters (in the form of a single string) that can be used to break the *value* into distinct words.

The *filter template* is used to construct an LDAP filter (see description below)

The *match description* is returned to the caller along with a filter as a piece of text that can be used to describe the sort of LDAP search that took place. It should correctly compete both of the following phrases: "One *match description* match was found for ..." and "Three *match description* matches were found for...."

The *search scope* is optional, and should be one of `base`, `onelevel`, or `subtree`. If *search scope* is not provided, the default is `subtree`.

The remaining lines of the filter list should contain two or three tokens, a *filter template*, a *match description* and an optional *search scope*.

The *filter template* is similar in concept to a `printf(3C)` style format string. Everything is taken literally except for the character sequences:

- `%v` Substitute the entire *value* string in place of the `%v`.
- `%v$` Substitute the last word in this field.
- `%vN` Substitute word *N* in this field (where *N* is a single digit 1-9). Words are numbered from left to right within the *value* starting at 1.
- `%vM-N` Substitute the indicated sequence of words where *M* and *N* are both single digits 1-9.
- `%vN-` Substitute word *N* through the last word in *value* where *N* is again a single digit 1-9.

Examples EXAMPLE 1 An LDAP Filter Configuration File

The following LDAP filter configuration file contains two filter sets, `example1` and `example2 onelevel`, each of which contains four filter lists.

```
# ldap filter file
#
example1
"="          " "          "%v"          "arbitrary filter"
"[0-9][0-9-]*" " "          "(telephoneNumber=%v)" "phone number"

"@"          " "          "(mail=%v)"          "email address"

"^.[ _].*"   ". _"        "(cn=%v1* %v2-)"     "first initial"

".*[ _].$"   ". _"        "(cn=%v1-*)"         "last initial"

"[ _]"       ". _"        "(|(sn=%v1-)(cn=%v1-))" "exact"
              "(|(sn~=%v1-)(cn~=%v1-))" "approximate"

".*"         ". "         "(|(cn=%v1)(sn=%v1)(uid=%v1))" "exact"
              "(|(cn~=%v1)(sn~=%v1))"      "approximate"

"example2 onelevel"
"^..$"       " "         "(|(o=%v)(c=%v)(l=%v)(co=%v))" "exact" "onelevel"
              "(|(o~=%v)(c~=%v)(l~=%v)(co~=%v))" "approximate"

"onelevel"

" "          " "         "(|(o=%v)(l=%v)(co=%v)" "exact" "onelevel"
              "(|(o~=%v)(l~=%v)(co~=%v)" "approximate" "onelevel"

"."          " "         "(associatedDomain=%v)" "exact" "onelevel"

".*"         " "         "(|(o=%v)(l=%v)(co=%v)" "exact" "onelevel"
              "(|(o~=%v)(l~=%v)(co~=%v)" "approximate" "onelevel"
```

Attributes See [attributes\(5\)](#) for a description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	naming/ldap
Interface Stability	Committed

See Also [ldap_getfilter\(3LDAP\)](#), [ldap_ufn\(3LDAP\)](#), [attributes\(5\)](#)

Name ldapsearchprefs.conf – configuration file for LDAP search preference routines

Synopsis /etc/opt/SUNWconn/ldap/current/ldapsearchprefs.conf

Description The ldapsearchprefs.conf file contains information used by LDAP when searching the directory. Blank lines and lines that start with a hash (#) character are treated as comments and ignored. Non-comment lines contain one or more tokens. Tokens are separated by white space, and double quotes can be used to include white space inside a token.

Search preferences are typically used by LDAP-based client programs to specify what a user may search for, which attributes are searched, and which options are available to the user.

The first non-comment line specifies the version of the template information and must contain the token `Version` followed by an integer version number. For example:

```
Version 1
```

The current version is *1*, so the above example is always the correct opening line.

The remainder of the file consists of one or more search preference configurations. The first line of a search preference is a human-readable name for the type of object being searched for, for example `People` or `Organizations`. This name is stored in the `so_objtypeprompt` member of the `ldap_searchobj` structure (see [ldap_searchprefs\(3LDAP\)](#)). For example:

```
People
```

specifies a label for a search preference designed to find X.500 entries for people.

The next line specifies a list of options for this search object. The only option currently allowed is "internal" which means that this search object should not be presented directly to a user.

Options are placed in the `so_options` member of the `ldap_searchobj` structure and can be tested using the `LDAP_IS_SEARCHOBJ_OPTION_SET()` macro. Use "" if no special options are required.

The next line specifies a label to use for "Fewer Choices" searches. "Fewer Choices" searches are those where the user's input is fed to the `ldap_filter` routines to determine an appropriate filter to use. This contrasts with explicitly-constructed LDAP filters, or "More Choices" searches, where the user can explicitly construct an LDAP filter.

For example:

```
"Search For:"
```

can be used by LDAP client programs to label the field into which the user can type a "Fewer Choices" search.

The next line specifies an LDAP filter prefix to append to all "More Choices" searched. This is typically used to limit the types of entries returned to those containing a specific object class.

For example:

```
"(&(objectClass=person))"
```

would cause only entries containing the object class *person* to be returned by a search. Note that parentheses may be unbalanced here, since this is a filter prefix, not an entire filter.

The next line is an LDAP filter tag which specifies the set of LDAP filters to be applied for "Fewer Choices" searching. The line

```
"x500-People"
```

would tell the client program to use the set of LDAP filters from the ldap filter configuration file tagged "x500-People".

The next line specifies an LDAP attribute to retrieve to help the user choose when several entries match the search terms specified. For example:

```
"title"
```

specifies that if more than one entry matches the search criteria, the client program should retrieve the `title` attribute that and present that to the user to allow them to select the appropriate entry. The next line specifies a label for the above attribute, for example,

```
"Title:"
```

Note that the values defined so far in the file are defaults, and are intended to be overridden by the specific search options that follow.

The next line specifies the scope of the LDAP search to be performed. Acceptable values are subtree, onelevel, and base.

The next section is a list of "More Choices" search options, terminated by a line containing only the string END. For example:

```
"Common Name"   cn    11111   ""   ""
"Surname"       sn    11111   ""   ""
"Business Phone" "telephoneNumber" 11101   ""   ""
END
```

Each line represents one method of searching. In this example, there are three ways of searching - by Common Name, by Surname, and by Business Phone number. The first field is the text which should be displayed to user. The second field is the attribute which will be searched. The third field is a bitmap which specifies which of the match types are permitted for this search type. A "1" value in a given bit position indicates that a particular match type is valid, and a "0" indicates that it is not valid. The fourth and fifth fields are, respectively, the select attribute name and on-screen name for the selected attribute. These values are intended to override the defaults defined above. If no specific values are specified, the client software uses the default values above.

The next section is a list of search match options, terminated by a line containing only the string END. Example:


```

"exactly matches"      "(%a=%v)"
"approximately matches" "(%a~=%v)"
"starts with"         "(%a=%v*)"
"ends with"           "(%a=%*v)"
"contains"            "(%a=%*v*)"
END

```

In this example, there are five ways of refining the search. For each method, there is an LDAP filter suffix which is appended to the ldap filter.

Examples EXAMPLE 1 A Sample Configuration Using Search Preference for "people"

The following example illustrates one possible configuration of search preferences for "people".

```

# Version number
Version 1
# Name for this search object
People
# Label to place before text box user types in
"Search For:"
# Filter prefix to append to all "More Choices" searches
"(&(objectClass=person)"
# Tag to use for "Fewer Choices" searches - from ldapfilter.conf file
"x500-People"
# If a search results in > 1 match, retrieve this attribute to help
# user distinguish between the entries...
multilineDescription
# ...and label it with this string:
"Description"
# Search scope to use when searching
subtree
# Follows a list of "More Choices" search options. Format is:
# Label, attribute, select-bitmap, extra attr display name, extra attr ldap name
# If last two are null, "Fewer Choices" name/attributes used
"Common Name"          cn          11111 "" ""
"Surname"              sn          11111 "" ""
"Business Phone"       "telephoneNumber" 11101 "" ""
"E-Mail Address"       "mail"       11111 "" ""
"Uniqname"             "uid"        11111 "" ""
END
# Match types
"exactly matches"      "(%a=%v)"
"approximately matches" "(%a~=%v)"
"starts with"         "(%a=%v*)"
"ends with"           "(%a=%*v)"
"contains"            "(%a=%*v*)"
END

```

In this example, the user may search for People. For "fewer choices" searching, the tag for the [ldapfilter.conf\(4\)](#) file is "x500-People".

Attributes See [attributes\(5\)](#) for a description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	naming/ldap
Interface Stability	Committed

See Also [ldap_searchprefs\(3LDAP\)](#), [attributes\(5\)](#)

- Name** ldaptemplates.conf – configuration file for LDAP display template routines
- Synopsis** /etc/opt/SUNWconn/ldap/current/ldaptemplates.conf
- Description** The ldaptemplates.conf file contains information used by the LDAP display routines.
- Blank lines and lines that start with a hash character ('#') are treated as comments and ignored. Non-comment lines contain one or more tokens. Tokens are separated by white space, and double quotes can be used to include white space inside a token.
- The first non-comment line specifies the version of the template information and must contain the token `Vers ion` followed by an integer version number. For example,
- ```
Vers ion 1
```
- The current version is *1*, so the above example is always the correct first line.
- The remainder of the file consists of one or more display templates. The first two lines of the display template each contain a single token that specifies singular and plural names for the template in a user-friendly format. For example,
- ```
"Person"
"Peop le"
```
- specifies appropriate names for a template designed to display person information.
- The next line specifies the name of the icon or similar element that is associated with this template. For example,
- ```
"person icon"
```
- The next line is a blank-separated list of template options. "" can be used if no options are desired. Available options are: `addable` (it is appropriate to allow entries of this type to be added), `modrdn` (it is appropriate to offer the `mod ify rdn` operation), `altview` (this template is an alternate view of another template). For example,
- ```
"addable" "modrdn"
```
- The next portion of the template is a list of X.500 object classes that is used to determine whether the template should be used to display a given entry. The object class information consists of one or more lines, followed by a terminating line that contains the single token `END`. Each line contains one or more object class names, all of which must be present in a directory entry. Multiple lines can be used to associate more than one set of object classes with a given template. For example,
- ```
emailPerson
orgPerson
END
```
- means that the template is appropriate for display of `emailPerson` entries or `orgPerson` entries.

The next line after the object class list is the name of the attribute to authenticate as to make changes (use "" if it is appropriate to authenticate as the entry itself). For example,

```
"owner"
```

The next line is the default attribute to use when naming a new entry, for example,

```
"cn"
```

The next line is the distinguished name of the default location under which new entries are created. For example,

```
"o=XYZ, c=US"
```

The next section is a list of rules used to assign default values to new entries. The list should be terminated with a line that contains the single token `END`. Each line in this section should either begin with the token `constant` and be followed by the name of the attribute and a constant value to assign, or the line should begin with `addersdn` followed by the name of an attribute whose value will be the DN of the person who has authenticated to add the entry. For example,

```
constant associatedDomain XYZ.us
addersdn seeAlso
END
```

The last portion of the template is a list of items to display. It consists of one or more lines, followed by a terminating line that contains the single token `END`. Each line must begin with the token `same row` or the token `item`

It is assumed that each item appears on a row by itself unless it was preceded by a `same row` line (in which case it should be displayed on the same line as the previous item, if possible). Lines that begin with `same row` should not have any other tokens on them.

Lines that begin with `item` must have at least three more tokens on them: an item type, a label, and an attribute name. Any extra tokens are taken as extra arguments.

The item type token must be one of the following strings:

|                      |                                                             |
|----------------------|-------------------------------------------------------------|
| <code>cis</code>     | case-ignore string attributes                               |
| <code>mls</code>     | multiline string attributes                                 |
| <code>mail</code>    | RFC-822 conformant mail address attributes                  |
| <code>dn</code>      | distinguished name pointer attributes                       |
| <code>bool</code>    | Boolean attributes                                          |
| <code>jpeg</code>    | JPEG photo attributes                                       |
| <code>jpegbtn</code> | a button that will retrieve and show a JPEG photo attribute |
| <code>fax</code>     | FAX T.4 format image attributes                             |

---

```

faxbtn a button that will retrieve and show a FAX photo attribute
audiobtn audio attributes
time UTC time attributes
date UTC time attributes where only the date portion should be shown
url labeled Uniform Resource Locator attributes
searchact define an action that will do a directory search for other entries
linkact define an action which is a link to another display template
protected for an encrypted attribute, with values displayed as asterisks

```

An example of an item line for the drink attribute (displayed with label "Work Phone"):

```
item cis "Work Phone" telephoneNumber
```

### Examples EXAMPLE 1 A Sample Configuration File Containing a Template that Displays People Entries

The following template configuration file contains a templates for display of people entries.

```

#
LDAP display templates
#
Version must be 1 for now
#
Version 1
#
Person template
"Person"
"People"

name of the icon that is associated with this template
"person icon"

blank-separated list of template options (" for none)
"addable"

#
objectclass list
person
END

#
name of attribute to authenticate as (" means auth as this entry)
""

#

```

**EXAMPLE 1** A Sample Configuration File Containing a Template that Displays People Entries  
(Continued)

```

default attribute name to use when forming RDN of a new entry
#
"cn"

#
default location when adding new entries (DN; "" means no default)
"o=XYZ, c=US"

#
rules used to define default values for new entries
END

#
list of items for display
item jpegbtn "View Photo" jpegPhoto "Next Photo"
item audiobtn "Play Sound" audio
item cis "Also Known As" cn
item cis "Title" title
item mls "Work Address" postalAddress
item cis "Work Phone" telephoneNumber
item cis "Fax Number" facsimileTelephoneNumber
item mls "Home Address" homePostalAddress
item cis "Home Phone" homePhone
item cis "User ID" uid
item mail "E-Mail Address" mail
item cis "Description" description
item dn "See Also" seeAlso
END

```

**Attributes** See [attributes\(5\)](#) for a description of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Availability        | naming/ldap     |
| Interface Stability | Committed       |

**See Also** [ldap\\_disptmpl\(3LDAP\)](#), [ldap\\_entry2text\(3LDAP\)](#), [attributes\(5\)](#)

|                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>                | llc2 – LLC2 Configuration file                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Synopsis</b>            | /etc/llc2/default/llc2.*                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Description</b>         | <p>The <i>llc2</i> files contain information needed by LLC2 to establish the appropriate links to the underlying MAC layer drivers as well as the parameters necessary to configure the LLC (Logical Link Control) Class II Station Component structures for that link.</p> <p>The comments are made up of one or more lines starting with the "#" character in column 1.</p> <p>The main section consists of keyword/value pairs of the form <i>keyword=value</i>, used to initialize the particular adapter.</p> <p>A sample of the <i>llc2</i> is presented below:</p> <pre> devicename=/dev/dnet deviceinstance=1 llc2_on=1      # LLC2: On/Off on this device deviceloopback=1 timeinterval=0 # LLC2: Timer Multiplier acktimer=2     # LLC2: Ack Timer rsptimer=2     # LLC2: Response Timer polltimer=4    # LLC2: Poll Timer rejecttimer=6  # LLC2: Reject Timer rembusytimer=8 # LLC2: Remote Busy Timer inacttimer=30  # LLC2: Inactivity Timer maxretry=6     # LLC2: Maximum Retry Value xmitwindowsz=14 # LLC2: Transmit Window Size rcvwindowsz=14 # LLC2: Receive Window Size </pre> |
| MAC specific Parameters    | <p>The <i>llc2.ppa</i> file contains 4 parameters directly related to the underlying MAC-level driver. These are the name of the physical device, the instance of the device, whether LLC2 can be used with this device, and whether the device is capable of looping back data addressed to the node's unique MAC address, broadcast address, or multicast addresses.</p> <p>Setting the <code>llc2_on</code> parameter to 1 means that LLC2 can be used with this device; setting it to 0 means otherwise. Setting the loopback parameter to 1 means that the LLC2 module will loop back data addressed to this node's unique MAC address or to a broadcast/multicast address.</p> <p>The most likely use is for a media that cannot receive its own transmissions (for example, ethernet) or when the MAC-level driver intentionally does not loop back data addressed to the local node under the assumption that the upper layers have already done so.</p>                                                                                                                                    |
| Host-Based LLC2 Parameters | <p>The LLC2 contains ten parameters in the configuration file (<i>/etc/llc2/default/llc2.ppa</i>) that apply to configurations using the Host-Based LLC2 component for connection-oriented operation over an Ethernet, Token Ring, or FDDI media.</p> <p>The ten parameters break down into the following four groups:</p> <ul style="list-style-type: none"> <li>▪ Six parameters deal with timer settings for managing the flow of LLC elements of procedure (PDUs) on a data link connection.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

- One parameter is the multiplier that is used to determine the period of the interval timer for the station. A value of 1 means that each tick count represents 100 milliseconds; 5 means each tick count is 500 milliseconds. Should the parameter be omitted, the default value is 5, except for Token Ring links which use a default of 1.
- One parameter indicates how many times an operation should be retried on a data link connection.
- Two parameters are for controlling the number of unacknowledged I PDUs to send or receive on a data link connection.

Additional information on these parameters can be found in ISO 8802-2:1989, Section 7.8.

The following table of Logical Link Control Parameters provides the LLC configuration parameter names, default values, and ranges.

| Parameter    | Description                                                                                            | Default           | Range  |
|--------------|--------------------------------------------------------------------------------------------------------|-------------------|--------|
| timeinterval | The timer ticks in 100 ms intervals. This parameter is used to scale the following 5 timer parameters. | 5, except TPR - 1 | 0 - 10 |
| acktimer     | The connection acknowledgment timer length in (100 * timeinterval) ms.                                 | 2                 | > 0    |
| rsptimer     | The response acknowledgment timer length in (100 * timeinterval) ms.                                   | 2                 | > 0    |
| polltimer    | The connection poll timer length in (100 * timeinterval) ms.                                           | 4                 | > 0    |
| rejecttimer  | The connection reject timer length in (100 * timeinterval) ms.                                         | 6                 | > 0    |
| rembusytimer | The connection remote busy timer length in (100 * timeinterval) ms.                                    | 8                 | > 0    |
| inacttimer   | The connection inactivity timer length in (100 * timeinterval) ms.                                     | 30                | > 0    |



| Parameter    | Description                                                                                                                                  | Default | Range   |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------|---------|---------|
| maxretry     | The maximum number of retries of an action on a connection.                                                                                  | 6       | 0 - 100 |
| xmitwindowsz | The maximum number of unacknowledged I-format protocol data units that can be transmitted on a connection before awaiting an acknowledgment. | 14      | 0 - 127 |
| rcvwindowsz  | The maximum number of unacknowledged I-format protocol data units that can be received on a connection before an acknowledgment is sent.     | 14      | 0 - 127 |

Default values are set when the following conditions are true:

- The parameter is not set by the user.
- The user requests a default `/etc/llc2/default/llc2.instance` file, where *instance* is the sequence number, starting with 0, of the adapter as detected by `ifconfig(1M)`. For example, if there are 3 adapters on the machine, the default configuration files will be named in order as `/etc/llc2/default/llc2.0`, `/etc/llc2/default/llc2.1`, and `/etc/llc2/default/llc2.2`.
- The user codes a value of 0 for a parameter.

#### Timer Parameter Descriptions

**acktimer** The `acktimer` parameter is used to manage the following sample sequences:

1. Attempting to establish, reset, or disconnect a connection.

```
SABME start acknowledgment timer
or ----->
DISC
```

The acknowledgment timer expires before the receipt of a response.

```
SABME start acknowledgment timer
or ----->
DISC
```

```
 stop acknowledgment timer
<----- UA
```

2. Sending an FRMR in response to a received PDU of dubious distinction:

```
PDU with invalid N(R)
or
I PDU with invalid N(S)
or
```

```

<----- PDU of invalid length
 or
 unexpected UA PDU
 or
 response PDU with
 invalid P/F setting

 start acknowledgment timer
FRMR ----->
Acknowledgment timer expires before the receipt of a PDU.

 start acknowledgment timer
FRMR ----->

 stop acknowledgment timer
 SABME, FRMR
<----- DISC, or DM

```

3. There is also a special case of the acknowledgment timer, referred to in this implementation as the response acknowledgment timer (*rsptimer*). It is used when sending an I PDU.

```

 start response acknowledgement timer
I ----->

Response acknowledgment timer expires before the receipt of an
acknowledgment.

```

```

 start poll timer
RR ----->

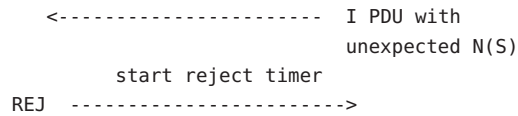
```

**polltimer** The *polltimer* parameter is used to manage situations where a Supervisory command PDU (RR, RNR, or REJ) is sent with the P/F bit set. This type of PDU is typically sent when:

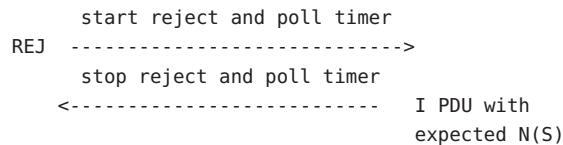
- There has been a period of inactivity on a connection in information transfer mode.
- The remote node must be notified of a local busy condition occurring in information transfer mode.

The expiration of the poll timer causes another Supervisory command PDU (which may be of a different type than the first) to be sent with the P/F bit set, provided the retry count has not exceeded the maximum retry value. This timer, then, provides an extended retry mechanism for a connection in information transfer mode.

`rejecttimer` The `rejecttimer` parameter controls the frequency with which a REJ PDU is sent to a remote node from which an I PDU with an unexpected N(S) was received and which has not corrected the situation by sending an I PDU with the expected N(S).



Reject timer expires before the receipt of an I PDU with an expected N(S).



`rembusytimer` The `rembusytimer` parameter is used to determine how long the local node should wait, after the remote node sends an RNR to indicate it is busy, before sending a Supervisory PDU with the P/F bit set to solicit the current state of the remote node. If the remote node indicates that it has cleared its busy condition before the timer expires, the local node stops the remote busy timer.

`inacttimer` The `inacttimer` parameter controls how much time is allowed to elapse on a connection in information transfer mode between the issuing of command PDUs by the local node. If the inactivity timer expires because a command PDU has not been generated in the configured time interval, a Supervisory PDU with the P/F bit set is sent to the remote node to solicit its current state, provided that the connection is in information transfer mode. Each time a command PDU is sent by the local node, the inactivity timer is restarted.

The following rules of thumb should apply for the timer parameters:

- The `acktimer`, `rsptimer`, and `polltimer` parameters should have small relative values to allow for quick recovery from common transient error conditions on a connection.
- The `rejecttimer` and `rembusytimer` parameters should have intermediate relative values to allow the local and remote nodes time to recover without resorting to possibly unnecessary polling cycles.
- The `inacttimer` parameter should be set to a large relative value to provide a safety net in information transfer mode.

You may need to shift the values for the timer parameters to higher values if bridges are included in the network or a user application requires a substantial amount of time to respond to connection establishment requests or handle information flow.

**Maximum Retry Parameter Description** The `maxretry` parameter determines the number of times a recovery operation is performed before notifying the user that an error has occurred on a connection. Typical examples of its use include the following:

- When the remote node fails to respond to a SABME sent by the local node to establish or reset the connection, the SABME is resent each time the acknowledgment timer expires, up to `maxretry` number of times.
- In information transfer mode, if the response acknowledgment timer expires after an I PDU has been sent, an RR with the P/F bit set is sent (and resent each time the poll timer expires) until the remote node responds or `maxretry` number of RRs have been sent.

In general, the `maxretry` value should not need to be large. Since the acknowledgment and poll timers are typically used in recovery operations that involve the `maxretry` parameter, the product of `maxretry` and either `acktimer`, `rsptimer`, or `polltimer` gives a rough estimate of the length of time allotted for the connection to attempt internal error recovery before notifying the user.

**Window Size Parameter Descriptions** `rcvwindowsz` The `rcvwindowsz` parameter is used to set the receive window size for I PDUs received locally on a connection. This value should agree with the transmit window size set for the connection at the remote node. If the local `rcvwindowsz` is greater than the remote transmit window size, I PDUs sent by the remote node are not acknowledged quickly. If the local `rcvwindowsz` is less than the remote transmit window size, there is a greater risk of the local node generating FRMR PDUs, requiring intervention by the user application when transient errors on the connection require the remote node to retransmit an I PDU. REJ PDUs are recovered internally.

`xmitwindowsz` The `xmitwindowsz` parameter sets the local transmit window size for a connection. It denotes the number of unacknowledged I PDUs that the local node may have outstanding. The configured value should match the receive window size for the connection at the remote node, based on the same reasoning as for the `rcvwindowsz` parameter.

In many cases, the values assigned to `rcvwindowsz` and `xmitwindowsz` for adapters on a server node will depend on the transmit and receive window sizes specified for another LLC implementation on a client node. In cases where this LLC implementation is resident in both nodes, larger values for these parameters are useful in environments where much of the activity on a connection consists of file transfer operations. Smaller values are warranted if analysis of LLC2 connection component statistics reveals that connections are entering local or remote busy state frequently.

**Files** /etc/llc2/default/llc2.\*

**See Also** [llc2\\_autoconfig\(1\)](#), [llc2\\_config\(1\)](#), [ifconfig\(1M\)](#), [llc2\(7D\)](#)

**Name** logadm.conf – configuration file for logadm command

**Synopsis** /etc/logadm.conf

**Description** /etc/logadm.conf is the default configuration file for the log management tool [logadm\(1M\)](#). Comments are allowed using the pound character (#) and extend to the end of line. Each non-comment line has the form:

*logname options*

where *logname* is the name of the entry and *options* are the default command line options for the logadm command. The name of the entry may be the same as the name of the log file, or a log file name may be given in the options section of the entry. Long lines may be folded using a backslash followed by a newline to continue an entry on the next line. Single or double quotes may be used to protect spaces or alternate-style quotes in strings.

The preferred method for changing /etc/logadm.conf is to use the -V, -w, and -r options to the logadm(1M) command, which allow you to lookup an entry, write an entry, or remove an entry from /etc/logadm.conf.

A full description of how and when /etc/logadm.conf is used and sample entries are found in [logadm\(1M\)](#).

By default, [logadm\(1M\)](#) works in GMT. Therefore, all entries in /etc/logadm.conf will have a GMT timestamp. Users can use the -l option to set logadm to local time.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Availability        | system/core-os  |
| Interface Stability | Committed       |

**See Also** [logadm\(1M\)](#), [attributes\(5\)](#)

**Name** logindevperm, fctab – login-based device permissions

**Synopsis** /etc/logindevperm

**Description** The /etc/logindevperm file contains information that is used by [login\(1\)](#) and [ttypmon\(1M\)](#) to change the owner, group, and permissions of devices upon logging into or out of a console device. By default, this file contains lines for the keyboard, mouse, audio, and frame buffer devices.

The owner of the devices listed in /etc/logindevperm is set to the owner of the console by [login\(1\)](#). The group of the devices is set to the owner's group specified in /etc/passwd. The permissions are set as specified in /etc/logindevperm.

If the console is /dev/vt/active, the owner of the devices is the first user logged in on the consoles (/dev/console or /dev/vt/#). Upon this first user's logout the owner and group of these devices is reset by [ttypmon\(1M\)](#) to owner root and root's group as specified in /etc/passwd.

Fields are separated by a TAB or SPACE characters. Blank lines and comments can appear anywhere in the file; comments start with a hashmark, (#), and continue to the end of the line.

The first field specifies the name of a console device (for example, /dev/console). By default, it is /dev/vt/active, which points to the current active console, including /dev/console and all virtual consoles (/dev/vt/#). The second field specifies the permissions to which the devices in the *device\_list* field (third field) are set. These permissions must be expressed in octal format, for example, 0774. A *device\_list* is a colon-separated list of device names. A device name must be a /dev link.

A directory or logical name in the device name can be either one of the following:

- A fully qualified name, for example, fbs.
- A regular expression, for example, [a-z0-9.]+. See [regex\(5\)](#) for more information on regular expressions.
- The wildcard character \* specifying all directory or node names (except . and .., for example, /dev/fbs/\* specifies all frame buffer devices).

Some examples of /etc/logindevperm file entries include:

```
/dev/usb/[0-9a-f]+[.][0-9a-f]+/[0-9]+/[a-z0-9.]+
/dev/usb/[0-9a-f]+[.][0-9a-f]+/[0-9]+/*
/dev/usb/[0-9a-f]+[.][0-9a-f]+/*/*
```

Specify all [ugen\(7D\)](#) endpoints and status nodes.

Drivers can also be specified to limit the permission changes to minor nodes owned by the specified drivers. For example,

```
/dev/console 0600 /dev/usb/[0-9a-f]+[.][0-9a-f]+/[0-9]+/* \
driver=usb_mid,scsa2usb,usbprn # libusb devices
```

Due to the persistence of [devfs\(7FS\)](#) minor node management, the user should be logged in as root if the list of minor nodes will be reduced and the devices should all be plugged in.

Once the devices are owned by the user, their permissions and ownership can be changed using [chmod\(1\)](#) and [chown\(1\)](#), as with any other user-owned file.

Upon logout the owner and group of these devices are reset by [ttymon\(1M\)](#) to owner root and root's group as specified in `/etc/passwd` (typically `other`). The permissions are set as specified in the `/etc/logindevperm` file.

**Files** `/etc/passwd` File that contains user group information.

**See Also** [chmod\(1\)](#), [chown\(1\)](#), [login\(1\)](#), [ttymon\(1M\)](#), [passwd\(4\)](#), [regexp\(5\)](#), [ugen\(7D\)](#)

**Notes** `/etc/logindevperm` provides a superset of the functionality provided by `/etc/fstab` in SunOS 4.x releases.



**Name** loginlog – log of failed login attempts

**Description** After five unsuccessful login attempts, all the attempts are logged in the file `/var/adm/loginlog`. This file contains one record for each failed attempt. Each record contains the login name, tty specification, and time.

This is an ASCII file. Each field within each entry is separated from the next by a colon. Each entry is separated from the next by a new-line.

By default, `loginlog` does not exist, so no logging is done. To enable logging, the log file must be created with read and write permission for owner only. Owner must be `root` and group must be `sys`.

**Files** `/var/adm/loginlog`

**See Also** [login\(1\)](#), [passwd\(1\)](#)

**Name** magic – file command's magic number file

**Synopsis** /etc/magic

**Description** The `file(1)` command identifies the type of a file using, among other tests, a test for whether the file begins with a certain *magic number*. The `/etc/magic` file, or a file specified as an option-argument to the `-m` or `-M` options of `file(1)`, specifies what magic numbers are to be tested for, what message to print if a particular magic number is found, and additional information to extract from the file.

Each line of the file specifies a position-sensitive test to perform. A test compares the data starting at a particular offset in the file with a 1-byte, 2-byte, 4-byte, or 8-byte numeric value or string. If the test succeeds, a message is printed. The line consists of the following fields (separated by tabs): *offset type value message*

*offset*        A number specifying the offset, in bytes, into the file of the data which is to be tested.

*type*         The type of the data to be tested. The possible values are:

|                      |                            |
|----------------------|----------------------------|
| byte, d1, dC         | A one-byte signed value.   |
| short, d2, dS        | A 2-byte signed value.     |
| long, d4, dI, dL, d  | A 4-byte signed value.     |
| llong, d8            | An 8-byte signed value     |
| ubyte, u1, uC        | A one-byte unsigned value. |
| ushort, u2, uS       | A 2-byte unsigned value.   |
| ulong, u4, uI, uL, u | A 4-byte unsigned value.   |
| ullong, u8           | An 8-byte unsigned value.  |
| string, s            | A string of bytes.         |

All type specifiers, except for `string` and `s`, may be followed by a mask specifier of the form `&number`. If a mask specifier is given, the value is AND'ed with the *number* before any comparisons are done. The *number* is specified in C form. For instance, 13 is decimal, 013 is octal, and 0x13 is hexadecimal.

*value*        The value to be compared with the value from the file. If the type is numeric, this value is specified in C form. If it is a string, it is specified as a C string with the usual escapes permitted (for instance, `\n` for NEWLINE).

*Numeric values* may be preceded by a character indicating the operation to be performed, as follows:

=       The value from the file must equal the specified value.

- < The value from the file must be less than the specified value.
- > The value from the file must be greater than the specified value.
- & All the bits in the specified value must be set in the value from the file.
- ^ At least one of the bits in the specified value must not be set in the value from the file.
- x Any value will match.

If the character is omitted, it is assumed to be “=”.

For comparison of numeric values, the sign and size of both the value in the file and the value from the *value* field of the magic entry will match that of the corresponding *type* field. If there is a non-zero mask (&) in the *type* field, the comparison will be unsigned.

For string values, the byte string from the file must match the specified byte string. The byte string from the file which is matched is the same length as the specified byte string. If the value is a string, it can contain the following sequences:

- `\character` The backslash-escape sequences `\\`, `\a`, `\b`, `\f`, `\n`, `\r`, `\t`, `\v`.
- `\octal` Octal sequences that can be used to represent characters with specific coded values. An octal sequence consists of a backslash followed by the longest sequence of one, two, or three octal-digit characters (01234567).

*message* The message to be printed if the comparison succeeds. If the string contains a `printf(3C)` format specification, the value from the file (with any specified masking performed) is printed using the message as the format string.

Some file formats contain additional information which is to be printed along with the file type. A line which begins with the character “>” indicates additional tests and messages to be printed. If the test on the line preceding the first line with a “>” succeeds, the tests specified in all the subsequent lines beginning with “>” are performed, and the messages are printed if the tests succeed. The next line which does not begin with a “>” terminates this.

**Files** /etc/magic

**See Also** [file\(1\)](#), [file\(1B\)](#), [printf\(3C\)](#)

**Notes** In Solaris 9 and prior releases, the file utility may have performed unsigned comparisons for types `byte`, `short`, and `long`. Old user-defined magic files, which were specified with the `-m` option, will need modification of `byte`, `short`, and `long` entries to their corresponding unsigned types (`ubyte`, `ushort`, or `ulong`) for those entries for which all of the following are true:

- The entry uses the “<” or the “>” operator.
- The type field does not contain a non-zero mask.
- The intention of the entry is to test unsigned values.

For example, if the following entry is expected to match any non-zero, one-byte value from the file, including values for which the sign bit is on:

```
#offset type value message
0 byte >0 this matches any non-zero value
```

then that entry should be changed to:

```
0 ubyte >0 this matches any non-zero value
```

In Solaris 7 through Solaris 9, when applying tests for magic file entries whose type field is the numeric type “short” or “long”, the file utility in the x86 environment would switch the byte order of the numeric values read. Starting in Solaris 10, the byte order will not be switched on x86. A test for a numeric value whose byte order is identical in both little- and big-endian architectures may require two magic file entries, to ensure that the test correctly identifies files in both environments. For example, a magic file entry that will match on a big-endian system may look like this:

```
0 long 0xf00000ff extended accounting file
```

Its corresponding magic file entry that will match the same value on a little-endian system would look like this:

```
0 long 0xff0000f0 extended accounting file
```

**Bugs** There should be more than one level of subtests, with the level indicated by the number of ‘>’ at the beginning of the line.

**Name** mddb.cf – metadvice state database replica locations

**Synopsis** /etc/lvm/mddb.cf

**Description** The /etc/lvm/mddb.cf file is created when the `metadb(1M)` command is invoked. You should never directly edit this file.

The file /etc/lvm/mddb.cf is used by the `metainit(1M)` command to find the locations of the metadvice state databases replicas. The `metadb` command creates the file and updates it each time it is run. Similar information is entered in the /kernel/drv/md.conf file.

Each metadvice state database replica has a unique entry in the /etc/lvm/mddb.cf file. Each entry contains the *driver* and *minor* unit numbers associated with the block physical device where a replica is stored. Each entry also contains the block number of the master block, which contains a list of all other blocks in the replica.

Entries in the /etc/lvm/mddb.cf file are of the form: *driver\_name minor\_t daddr\_t checksum* where *driver\_name* and *minor\_t* represent the device number of the physical device storing this replica. *daddr\_t* is the disk block address. *checksum* is used to make certain the entry has not been corrupted. A pound sign (#) introduces a comment.

**Examples** EXAMPLE 1 Sample File

The following example shows a mddb.cf file.

```
#metadvice database location file do not hand edit
#driver minor_t daddr_t device id checksum
sd 152 16 id1,sd@SSEAGATE_JDD288110MC9LH/a -2613
```

In the example above, the value for *daddr\_t* indicates that the offset from the start of a given partition is 16 disk blocks from the start of that partition.

**Files**

- /etc/lvm/mddb.cf
- /kernel/drv/md.conf

**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)`, `metassist(1M)`, `metaset(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.cf(4)`, `md.tab(4)`, `attributes(5)`, `md(7D)`

*Solaris Volume Manager Administration Guide*

**Name** md.tab, md.cf – Solaris Volume Manager utility files

**Synopsis** /etc/lvm/md.tab  
/etc/lvm/md.cf

**Description** The file /etc/lvm/md.tab can be used by `metainit(1M)` and `metadb(1M)` to configure metadevices, hot spare pools, and metadevice state database replicas in a batch-like mode. Solaris Volume Manager does not store configuration information in the /etc/lvm/md.tab file. You can use:

```
metastat -p > /etc/lvm/md.tab
```

to create this file. Edit it by hand using the instructions in the md.tab.4 file. Similarly, if no hot spares are in use, the `cp md.cf md.tab` command generates an acceptable version of the md.tab file, with the editing caveats previously mentioned.

When using the md.tab file, each metadevice, hot spare pool, or state database replica in the file must have a unique entry. Entries can include the following: simple metadevices (stripes, concatenations, and concatenations of stripes); mirrors, soft partitions, and RAID5 metadevices; hot spare pools; and state database replicas. Because md.tab contains only entries that you enter in it, do not rely on the file for the current configuration of metadevices, hot spare pools, and replicas on the system at any given time.

Tabs, spaces, comments (by using a pound sign, #), and continuation of lines (by using a backslash-newline), are allowed.

Typically, you set up metadevices according to information specified on the command line by using the `metainit` command. Likewise, you set up state database replicas with the `metadb` command.

An alternative to the command line is to use the md.tab file. Metadevices and state database replicas can be specified in the md.tab file in any order, and then activated in a batch-like mode with the `metainit` and `metadb` commands.

If you edit the md.tab file, you specify one complete configuration entry per line. Metadevices are defined using the same syntax as required by the `metainit` command. You then run the `metainit` command with either the `-a` option, to activate all metadevices in the 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. 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 will fail when `metainit -an` runs (even if it would succeed with `metainit -a`).

State database replicas are defined in the /etc/lvm/md.tab file as follows: `mddb number options [ slice... ]` Where `mddb number` is the characters `mddb` followed by a number of two or more digits that identifies the state database replica. `slice` is a physical slice. For example:

mddb05 /dev/dsk/c0t1d0s2. The file /etc/lvm/md.cf is a backup of the configuration used for disaster recovery. Whenever the Volume Manager configuration is changed, this file is automatically updated (except when hot sparing occurs). You should not directly edit this file.

### Examples EXAMPLE 1 Concatenation

All drives in the following examples have the same size of 525 Mbytes.

This example shows a metadvice, /dev/md/dsk/d7, consisting of a concatenation of four disks.

```
#
(concatenation of four disks)
#
d7 4 1 c0t1d0s0 1 c0t2d0s0 1 c0t3d0s0 1 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. Note that the first disk sector in all of the above 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. Since skipping only the first sector would create an irregular disk geometry, the entire first cylinder of these disks will be skipped. This allows higher level file system software to optimize block allocations correctly.

### EXAMPLE 2 Stripe

This example shows a metadvice, /dev/md/dsk/d15, consisting of two slices.

```
#
(stripe consisting of two disks)
#
d15 1 2 c0t1d0s2 c0t2d0s2 -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 will be 32 Kbytes. If the interlace size were not specified, the stripe would use the default value of 16 Kbytes.

### EXAMPLE 3 Concatenation of Stripes

This example shows a metadvice, /dev/md/dsk/d75, consisting of a concatenation of two stripes of three disks.

```
#
(concatenation of two stripes, each consisting of three disks)
#
d75 2 3 c0t1d0s2 c0t2d0s2 c0t3d0s2 -i 16k \
 3 c1t1d0s2 c1t2d0s2 c1t3d0s2 -i 32k
```

**EXAMPLE 3** Concatenation of Stripes *(Continued)*

On the first line, the `-i` followed by `16k` specifies that the stripe's interlace size is 16 Kbytes. The second set specifies the stripe interlace size will be 32 Kbytes. If the second set did not specify 32 Kbytes, the set would use default interlace value of 16 Kbytes. The blocks of each set of three disks are interlaced across three disks.

**EXAMPLE 4** Mirroring

This example shows a three-way mirror, `/dev/md/dsk/d50`, consisting of three submirrors. This mirror does not contain any existing data.

```
#
(mirror)
#
d50 -m d51
d51 1 1 c0t1d0s2
d52 1 1 c0t2d0s2
d53 1 1 c0t3d0s2
```

In this example, a one-way mirror is first defined using the `-m` option. The one-way mirror consists of submirror `d51`. The other two submirrors, `d52` and `d53`, are attached later using the `metattach` command. The default read and write options in this example are a round-robin read algorithm and parallel writes to all submirrors. The order in which mirrors appear in the `/etc/lvm/md.tab` file is unimportant.

**EXAMPLE 5** RAID5

This example shows a RAID5 metadvice, `d80`, consisting of three slices:

```
#
(RAID devices)
#
d80 -r c0t1d0s1 c1t0d0s1 c2t0d0s1 -i 20k
```

In this example, a RAID5 metadvice is defined using the `-r` option with an interlace size of 20 Kbytes. The data and parity segments will be striped across the slices, `c0t1d0s1`, `c1t0d0s1`, and `c2t0d0s1`.

**EXAMPLE 6** Soft Partition

This example shows a soft partition, `d85`, that reformats an entire 9 GB disk. Slice 0 occupies all of the disk except for the few Mbytes taken by slice 7, which is space reserved for a state database replica. Slice 7 will be a minimum of 4Mbytes, but could be larger, depending on the disk geometry. `d85` sits on `c3t4d0s0`.

Drives are repartitioned when they are added to a diskset only if Slice 7 is not set up correctly. A small portion of each drive is reserved in Slice 7 for use by Volume Manager. The remainder



**EXAMPLE 6** Soft Partition (Continued)

of the space on each drive is placed into Slice 0. Any existing data on the disks is lost after repartitioning. After adding a drive to a diskset, you can repartition the drive as necessary. However, Slice 7 should not be moved, removed, or overlapped with any other partition.

Manually specifying the offsets and extents of soft partitions is not recommended. This example is included for to provide a better understanding of the file if it is automatically generated and for completeness.

```
#
(Soft Partitions)
d85 -p -e c3t4d0 9g
```

In this example, creating the soft partition and required space for the state database replica occupies all 9 GB of disk c3t4d0.

**EXAMPLE 7** Soft Partition

This example shows the command used to re-create a soft partition with two extents, the first one starting at offset 20483 and extending for 20480 blocks and the second extent starting at 135398 and extending for 20480 blocks:

```
#
(Soft Partitions)
#
d1 -p c0t3d0s0 -o 20483 -b 20480 -o 135398 -b 20480
```

**EXAMPLE 8** Hot Spare

This example shows a three-way mirror, /dev/md/dsk/d10, consisting of three submirrors and three hot spare pools.

```
#
(mirror and hot spare)
#
d10 -m d20
d20 1 1 c1t0d0s2 -h hsp001
d30 1 1 c2t0d0s2 -h hsp002
d40 1 1 c3t0d0s2 -h hsp003
hsp001 c2t2d0s2 c3t2d0s2 c1t2d0s2
hsp002 c3t2d0s2 c1t2d0s2 c2t2d0s2
hsp003 c1t2d0s2 c2t2d0s2 c3t2d0s2
```

In this example, a one-way mirror is first defined using the `-m` option. The submirrors are attached later using the `metattach(1M)` command. The hot spare pools to be used are tied to the submirrors with the `-h` option. In this example, there are three disks used as hot spares, defined in three separate hot spare pools. The hot spare pools are given the names hsp001,

**EXAMPLE 8** Hot Spare (Continued)

hsp002, and hsp003. Setting up three hot spare pools rather than assigning just one hot spare with each component helps to maximize the use of hardware. This configuration enables the user to specify that the most desirable hot spare be selected first, and improves availability by having more hot spares available. At the end of the entry, the hot spares to be used are defined. Note that, when using the `md.tab` file, to associate hot spares with metadevices, the hot spare pool does not have to exist prior to the association. Volume Manager takes care of the order in which metadevices and hot spares are created when using the `md.tab` file.

**EXAMPLE 9** State Database Replicas

This example shows how to set up an initial state database and three replicas on a server that has three disks.

```
#
(state database and replicas)
#
mddb01 -c 3 c0t1d0s0 c0t2d0s0 c0t3d0s0
```

In this example, three state database replicas are stored on each of the three slices. Once the above entry is made in the `/etc/lvm/md.tab` file, the `metadb` command must be run with both the `-a` and `-f` options. For example, typing the following command creates one state database replicas on three slices:

```
metadb -a -f mddb01
```

- Files**
- `/etc/lvm/md.tab`
  - `/etc/lvm/md.cf`

**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)`, `metassist(1M)`, `metaset(1M)`, `metastat(1M)`, `metasync(1M)`, `metattach(1M)`, `md.cf(4)`, `mddb.cf(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.

Stripes and RAID5 metadevices must contains slices or soft partitions only.

Mirroring of RAID5 metadevices is not allowed.

Soft partitions can be built directly on slices or can be the top level (accessible by applications directly), but cannot be in the middle, with other metadevices above and below them.

**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** mech, qop – mechanism and QOP files

**Synopsis** /etc/gss/mech  
/etc/gss/qop

**Description** The /etc/gss/mech and /etc/gss/qop files contain tables showing installed security mechanisms and the Quality of Protection (QOP) associated with them, respectively. As security mechanisms are installed on the system, entries are added to these two files. Contents of these files may be accessed either manually or programmatically. For example, manually with `cat(1)` or `more(1)`, or programmatically with either `rpc_gss_get_mechanisms(3NSL)` or `rpc_gss_get_mech_info(3NSL)`.

The order of entries in the /etc/gss/mech file is significant: the order should be from the most preferred to the least preferred mechanisms.

The /etc/gss/mech file contains five fields:

|                                         |                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mechanism name</i>                   | ASCII string representing the mechanism.                                                                                                                                                                                                                                                                                                                                                           |
| <i>object identifier</i>                | RPC OID for this mechanism.                                                                                                                                                                                                                                                                                                                                                                        |
| <i>shared library</i>                   | Shared library which implements the services provided by this mechanism.                                                                                                                                                                                                                                                                                                                           |
| <i>kernel module</i>                    | Kernel module which implements the services provided by this mechanism.                                                                                                                                                                                                                                                                                                                            |
| <i>library options</i> (optional field) | Optional parameters that are interpreted by the individual mechanism with which they are associated. Specific supported options are described in the documentation for the individual mechanism, if any. Not all mechanisms have support for optional parameters. <i>library options</i> must be enclosed in brackets ([ ]) so they may be differentiated from the optional kernel module entries. |

The /etc/gss/qop file contains three fields:

|                       |                                                                            |
|-----------------------|----------------------------------------------------------------------------|
| <i>QOP string</i>     | Name, in ASCII, of this Quality of Protection.                             |
| <i>QOP value</i>      | Numeric value by which RPC identifies this QOP.                            |
| <i>mechanism name</i> | ASCII string representing the mechanism with which this QOP is associated. |

**Examples** EXAMPLE 1 A Typical Entry in /etc/gss/mech

This is a typical entry in a /etc/gss/mech file:

```
kerberosv5 1.2.840.113554.1.2.2 mech_krb5.so kmech_krb5
```

**EXAMPLE 2** A Typical Entry in /etc/gss/qop

This is a typical entry in a /etc/gss/qop file:

```
GSS_KRB5_CONF_C_QOP_DES 0 kerberosv5
```

**See Also** [rpc\(3NSL\)](#), [rpc\\_gss\\_get\\_mechanisms\(3NSL\)](#), [rpc\\_gss\\_get\\_mech\\_info\(3NSL\)](#), [rpcsec\\_gss\(3NSL\)](#)

*ONC+ Developer's Guide*

**Name** meddb – mediator data file

**Synopsis** /etc/lvm/meddb

**Description** The file /etc/lvm/meddb is a data file used by [rpc.metamedd\(1M\)](#) to store the mediator data used in 2-string HA configurations.

**Files** /etc/lvm/meddb

**See Also** [rpc.metamedd\(1M\)](#)

*Sun Cluster 3.0 Collection*

*Solaris Volume Manager Administration Guide*

**Name** mnttab – mounted file system table

**Description** The file `/etc/mnttab` is really a file system that provides read-only access to the table of mounted file systems for the current host. `/etc/mnttab` is read by programs using the routines described in [getmntent\(3C\)](#). Mounting a file system adds an entry to this table. Unmounting removes an entry from this table. Remounting a file system causes the information in the mounted file system table to be updated to reflect any changes caused by the remount. The list is maintained by the kernel in order of mount time. That is, the first mounted file system is first in the list and the most recently mounted file system is last. When mounted on a mount point the file system appears as a regular file containing the current `mnttab` information.

Each entry is a line of fields separated by TABs in the form:

```
special mount_point fstype options time
```

where:

|                    |                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------|
| <i>special</i>     | The name of the resource that has been mounted.                                                             |
| <i>mount_point</i> | The pathname of the directory on which the filesystem is mounted.                                           |
| <i>fstype</i>      | The file system type of the mounted file system.                                                            |
| <i>options</i>     | The mount options. See respective mount file system man page in the <a href="#">See Also</a> section below. |
| <i>time</i>        | The time at which the file system was mounted.                                                              |

Examples of entries for the *special* field include the pathname of a block-special device, the name of a remote file system in the form of *host:pathname*, or the name of a *swap file*, for example, a file made with [mkfile\(1M\)](#).

**ioctls** The following [ioctl\(2\)](#) calls are supported:

|                   |                                                                                                                                                                                                                                                                                                                                                       |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MNTIOC_NMNTS      | Returns the count of mounted resources in the current snapshot in the <code>uint32_t</code> pointed to by <i>arg</i> .                                                                                                                                                                                                                                |
| MNTIOC_GETDEVLIST | Returns an array of <code>uint32_t</code> 's that is twice as long as the length returned by MNTIOC_NMNTS. Each pair of numbers is the major and minor device number for the file system at the corresponding line in the current <code>/etc/mnttab</code> snapshot. <i>arg</i> points to the memory buffer to receive the device number information. |
| MNTIOC_SETTAG     | Sets a tag word into the options list for a mounted file system. A tag is a notation that will appear in the options string of a mounted file system but it is not recognized or interpreted by the file system code. <i>arg</i> points to a filled in <code>mnttagdesc</code> structure, as shown in the following example:                          |

```

uint_t mtd_major; /* major number for mounted fs */
uint_t mtd_minor; /* minor number for mounted fs */
char *mtd_mntpt; /* mount point of file system */
char *mtd_tag; /* tag to set/clear */

```

If the tag already exists then it is marked as set but not re-added. Tags can be at most `MAX_MNTOPT_TAG` long.

Use of this ioctl is restricted to processes with the `{PRIV_SYS_MOUNT}` privilege.

**MNTIOC\_CLRTAG** Marks a tag in the options list for a mounted file system as not set. *arg* points to the same structure as `MNTIOC_SETTAG`, which identifies the file system and tag to be cleared.

Use of this ioctl is restricted to processes with the `{PRIV_SYS_MOUNT}` privilege.

**Errors**

**EFAULT** The *arg* pointer in an `MNTIOC_*` ioctl call pointed to an inaccessible memory location or a character pointer in a `mnttabdesc` structure pointed to an inaccessible memory location.

**EINVAL** The tag specified in a `MNTIOC_SETTAG` call already exists as a file system option, or the tag specified in a `MNTIOC_CLRTAG` call does not exist.

**ENAMETOOLONG** The tag specified in a `MNTIOC_SETTAG` call is too long or the tag would make the total length of the option string for the mounted file system too long.

**EPERM** The calling process does not have `{PRIV_SYS_MOUNT}` privilege and either a `MNTIOC_SETTAG` or `MNTIOC_CLRTAG` call was made.

**Files**

`/etc/mnttab` Usual mount point for `mnttab` file system

`/usr/include/sys/mntio.h` Header file that contains IOCTL definitions

**See Also** [mkfile\(1M\)](#), [mount\\_hfs\(1M\)](#), [mount\\_nfs\(1M\)](#), [mount\\_pcfs\(1M\)](#), [mount\\_ufs\(1M\)](#), [mount\(1M\)](#), [ioctl\(2\)](#), [read\(2\)](#), [poll\(2\)](#), [stat\(2\)](#), [getmntent\(3C\)](#)

**Warnings** The `mnttab` file system provides the previously undocumented `dev=xxx` option in the option string for each mounted file system. This is provided for legacy applications that might have been using the `dev=information` option.

Using `dev=option` in applications is strongly discouraged. The device number string represents a 32-bit quantity and might not contain correct information in 64-bit environments.

Applications requiring device number information for mounted file systems should use the [getextmntent\(3C\)](#) interface, which functions properly in either 32- or 64-bit environments.



**Notes** The snapshot of the `mnttab` information is taken any time a `read(2)` is performed at offset 0 (the beginning) of the `mnttab` file. The file modification time returned by `stat(2)` for the `mnttab` file is the time of the last change to mounted file system information. A `poll(2)` system call requesting a `POLLRDBAND` event can be used to block and wait for the system's mounted file system information to be different from the most recent snapshot since the `mnttab` file was opened.

**Name** mod\_ipp – Embedded Internet Print Protocol (IPP) listener for the Apache HTTP server

**Synopsis** /usr/apache/libexec/mod\_ipp.so

**Description** The mod\_ipp module implements RFCs 2910 and 2911 to provide an IPP handling service for the Apache HTTP server. When loaded on the Apache server, mod\_ipp processes all HTTP requests with MIME types of application/ipp. The mod\_ipp module also processes additional configuration directives to enable or disable portions of the protocol support.

**Using Configuration Directives** The following is a list of configuration directives that apply to the Apache IPP Listening service:

- `ipp-conformance` (*automatic*|1.0|1.1)
- `ipp-operation` (*operation*) (enable|disable)  
enable|disable

The values `true`, `yes`, `on`, `enable` are considered to be synonymous and will enable support for the named operation. All other values will disable support for the named operation.

**Operations** The following is a list of IPP handling service operations:

|                                     |                                                                                                                                                                                                                                                                |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>print-job</code>              | This operation is a required IPP operation that allows client systems to submit a print job with a single document embedded in the data stream. This operation is primarily used from the IPP support Microsoft has provided for its Windows (9X/ME/NT/2K/XP). |
| <code>print-uri</code>              | This is an optional IPP operation that allows client systems to submit a print job with a reference (URL) for a single document. This operation is currently not supported by the mod_ipp Apache Module.                                                       |
| <code>validate-job</code>           | This is a required IPP operation that allows client systems to simulate the submission of a print job to verify that the server is capable of handling the job. This operation is supported by mod_ipp.                                                        |
| <code>create-job</code>             | This is an optional IPP operation that allows client systems to submit a print job. The operation is used with the <code>send-document</code> and <code>send-uri</code> operations.                                                                            |
| <code>get-jobs</code>               | This is a required IPP operation that allows client systems to retrieve a list of print jobs from the print service.                                                                                                                                           |
| <code>get-printer-attributes</code> | This is a required IPP operation that allows client systems to retrieve attributes from the print service that describes the named printer object.                                                                                                             |

|                                 |                                                                                                                                                                                                                                                                                |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pause-printer</code>      | This is an optional IPP operation that allows client systems to stop job processing on the named print queue.                                                                                                                                                                  |
| <code>resume-printer</code>     | This is an optional IPP operation that allows client systems to resume job processing on the named print queue.                                                                                                                                                                |
| <code>purge-jobs</code>         | This is an optional IPP operation that allows client systems to cancel all print jobs on the named print queue.                                                                                                                                                                |
| <code>send-document</code>      | This is a required IPP operation that allows client systems to add documents to print jobs created with the <code>create-job</code> operation, but not yet submitted.                                                                                                          |
| <code>send-uri</code>           | This is an optional IPP operation that allows a client system to add a document reference (URI) to a print job created with the <code>create-job</code> operation, but not yet submitted. This operation is currently not supported by the <code>mod_ipp</code> Apache Module. |
| <code>cancel-job</code>         | This is a required IPP operation that allows client systems to cancel print jobs.                                                                                                                                                                                              |
| <code>get-job-attributes</code> | This is a required IPP operation that allows client systems to retrieve attributes that describe a print job from the print service.                                                                                                                                           |
| <code>hold-job</code>           | This is an optional IPP operation that allows client systems to hold print jobs.                                                                                                                                                                                               |
| <code>release-job</code>        | This is an optional IPP operation that allows client systems to release print jobs.                                                                                                                                                                                            |
| <code>restart-job</code>        | This is an optional IPP operation that allows client systems to restart print jobs.                                                                                                                                                                                            |
| <code>all</code>                | This is a place holder for enabling or disabling support for all IPP operations implemented by the <code>mod_ipp</code> Apache module.                                                                                                                                         |
| <code>required</code>           | This is a place holder for enabling or disabling support for the required IPP operations implemented by the <code>mod_ipp</code> Apache module.                                                                                                                                |

### Examples **EXAMPLE 1** Using a Configuration File to Start a Standalone Apache Server

The following configuration file can be used to start a standalone Apache server to respond to IPP request sent to port 631.

```
ServerType standalone
ServerRoot "/usr/apache"
PidFile /var/run/httpd-standalone-ipp.pid
ErrorLog /var/lp/logs/ipp-errors
```

**EXAMPLE 1** Using a Configuration File to Start a Standalone Apache Server *(Continued)*

```

Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15

MinSpareServers 1
MaxSpareServers 3
StartServers 1
MaxClients 150
MaxRequestsPerChild 0

LoadModule ipp_module libexec/mod_ipp.so

ClearModuleList
AddModule mod_ipp.c
AddModule mod_so.c

Port 631

User lp
Group lp

ServerAdmin lp@localhost
DefaultType application/ipp

<IFModule mod_app>
 <Location />
 ipp-operation all on
 </Location>
</IFModule mod_app>

```

A more restrictive configuration might include the following parameters:

```

<IFModule mod_app>
 <Location />
 ipp-operation all offn
 ipp-operation required on
 </Location>
</IFModule mod_app>

```

**Attributes** See [attributes\(5\)](#) or descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE           |
|----------------|---------------------------|
| Availability   | print/lp/ipp/ipp-listener |

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Volatile        |

**See Also** [man\(1\)](#), [catman\(1M\)](#), [attributes\(5\)](#)

Herriot, R., Ed., Butler, S., Moore, P., Turner, R., Wenn, J. *RFC 2910, Internet Printing Protocol/1.1: Encoding and Transport*. Network Working Group. September 2000.

Hastings, T., Ed., Herriot, R., deBry, R., Isaacson, S., Powell, P. *RFC 2911, Internet Printing Protocol/1.1: Model and Semantics*. Network Working Group. September 2000.

<http://www.apache.org>

**Notes** Configuration file directives are processed in the order listed in the config file. The default behavior is to enable support for all operations implemented in the `mod_ipp` Apache module.

Since the Apache IPP listening service implements some capabilities that are more of operator features, it may not be desirable to enable all IPP operations without requiring user authentication on the Apache listening service.

The following is an example of a more reasonable configuration for Apache IPP servers without user authentication enabled:

```
ipp-operations all disabled
ipp-operations required enabled
```

The printers and jobs available under this service can be accessed using URIs of the following form:

```
printer:
 http://server[:port]/printers/{queue}
 ipp://server[:port]/printers/{queue}

job:
 http://server[:port]/printers/{queue}/{job-id}
 ipp://server[:port]/printers/{queue}/{job-id}
```

631 is the default IPP port and implied when the URI scheme is `ipp`. However, some client implementations do not recognize the `ipp` URI scheme and require `http://server:631/...` instead. For example, Microsoft's IPP client implementation does not recognize the `ipp` scheme.

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

The `httpd(8)` man page and other Apache man pages are provided with the programming modules. To view the Apache manual pages 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** mpapi.conf – configuration file for libMPAPI

**Synopsis** /etc/mpapi.conf

**Description** The /etc/mpapi.conf file is used to specify the vendor-provided plugin library that is installed on the system. This file is used by the [libMPAPI\(3LIB\)](#) common library to load the individual plugin library when its interface is called. If changes are made to the file while the library is in use, the library should be unloaded and reloaded. Addition and removal of the plugin library should be handled through [MP\\_RegisterPlugin\(3MPAPI\)](#) and [MP\\_DeregisterPlugin\(3MPAPI\)](#).

Each plugin library entry is a single line of the form:

```
"id" "library file name"
```

where

**id** The identification of library. It is the reserved domain name of the vendor followed by . followed by the vendor specific name of the plugin that uniquely identifies the plugin library.

**library file name** The shared object library file in the absolute path format.

**Examples** **EXAMPLE 1** Example of an /etc/mpapi.conf file

```
This file contains names and references to MP API plugin libraries
#
Do NOT manually edit this file
#
Format:
#
<library ID> <library file name>
#
com.sun.mpapi32 /lib/libmpscsi_vhci.so
com.sun.mpapi64 /lib/64/libmpscsi_vhci.so
```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                          |
|---------------------|------------------------------------------|
| Availability        | system/library/storage/snia-mpapi        |
| Interface Stability | Committed                                |
| Standard            | ANSI INCITS 412 Multipath Management API |

**See Also** [libMPAPI\(3LIB\)](#), [MP\\_DeregisterPlugin\(3MPAPI\)](#), [MP\\_RegisterPlugin\(3MPAPI\)](#), [attributes\(5\)](#)

**Name** named.conf – configuration file for named

**Synopsis** named.conf

**Description** named.conf is the configuration file for [named\(1M\)](#). Statements are enclosed in braces and terminated with a semicolon. Clauses in the statements are also terminated with a semicolon. The usual comment styles are supported:

C style        */\* \*/*

C++ style     *//* to end of line

Unix style    *#* to end of line

ACL *acl string { address\_match\_element; ... };*

Key *key domain\_name {  
    algorithm string;  
    secret string;  
};*

Masters *masters string [ port integer ] {  
    ( masters | ipv4\_address [port integer] |  
      ipv6\_address [port integer] ) [ key string ]; ...  
};*

Server *server ( ipv4\_address[/prefixlen] | ipv6\_address[/prefixlen] ) {  
    bogus boolean;  
    edns boolean;  
    edns-udp-size integer;  
    max-udp-size integer;  
    provide-ixfr boolean;  
    request-ixfr boolean;  
    keys server\_key;  
    transfers integer;  
    transfer-format ( many-answers | one-answer );  
    transfer-source ( ipv4\_address | \* )  
        [ port ( integer | \* ) ];  
    transfer-source-v6 ( ipv6\_address | \* )  
        [ port ( integer | \* ) ];  
    support-ixfr boolean; // obsolete  
};*

Trusted-Keys *trusted-keys {  
    domain\_name flags protocol algorithm key; ...  
};*

Controls *controls {  
    inet ( ipv4\_address | ipv6\_address | \* )  
        [ port ( integer | \* ) ]  
    allow { address\_match\_element; ... }  
    [ keys { string; ... } ];*

```
 unix unsupported; // not implemented
 }
Logging logging {
 channel string {
 file log_file;
 syslog optional_facility;
 null;
 stderr;
 severity log_severity;
 print-time boolean;
 print-severity boolean;
 print-category boolean;
 };
 category string { string; ... };
};

LWRES lwres {
 listen-on [port integer] {
 (ipv4_address | ipv6_address) [port integer]; ...
 };
 view
 string optional_class;
 search { string; ... };
 ndots integer;
};

Options options {
 avoid-v4-udp-ports { port; ... };
 avoid-v6-udp-ports { port; ... };
 blackhole { address_match_element; ... };
 coresize size;
 datasize size;
 directory quoted_string;
 dump-file quoted_string;
 files size;
 heartbeat-interval integer;
 host-statistics boolean; // not implemented
 host-statistics-max number; // not implemented
 hostname (quoted_string | none);
 interface-interval integer;
 listen-on [port integer] \
 { address_match_element; ... };
 listen-on-v6 [port integer] \
 { address_match_element; ... };
 match-mapped-addresses boolean;
 memstatistics-file quoted_string;
 pid-file (quoted_string | none);
 port integer;
```



```

querylog boolean;
recursing-file quoted_string;
reserved-sockets integer;
random-device quoted_string;
recursive-clients integer;
serial-query-rate integer;
server-id (quoted_string | none |;
stacksize size;
statistics-file quoted_string;
statistics-interval integer; \
 // not yet implemented
tcp-clients integer;
tcp-listen-queue integer;
tkey-dhkey quoted_string integer;
tkey-gssapi-credential quoted_string;
tkey-domain quoted_string;
transfers-per-ns integer;
transfers-in integer;
transfers-out integer;
use-ixfr boolean;
version (quoted_string | none);
allow-recursion { address_match_element; ... };
allow-recursion-on { address_match_element; ... };
sortlist { address_match_element; ... };
topology { address_match_element; ... }; \
 // not implemented
auth-nxdomain boolean; // default changed
minimal-responses boolean;
recursion boolean;
rrset-order {
 [class string] [type string]
 [name quoted_string] string string; ...
};
provide-ixfr boolean;
request-ixfr boolean;
rfc2308-type1 boolean; // not yet implemented
additional-from-auth boolean;
additional-from-cache boolean;
query-source ((ipv4_address | *) | \
 [address (ipv4_address | *)]) \
 [port (integer | *)];
query-source-v6 ((ipv6_address | *) | \
 [address (ipv6_address | *)]) \
 [port (integer | *)];
use-queryport-pool boolean;
queryport-pool-ports integer;
queryport-pool-updateinterval integer;

```

```
cleaning-interval integer;
min-roots integer; // not implemented
lame-ttl integer;
max-ncache-ttl integer;
max-cache-ttl integer;
transfer-format (many-answers | one-answer);
max-cache-size size;
max-acache-size size;
clients-per-query number;
max-clients-per-query number;
check-names (master | slave | response)\
 (fail | warn | ignore);
check-mx (fail | warn | ignore);
check-integrity boolean;
check-mx-cname (fail | warn | ignore);
check-srv-cname (fail | warn | ignore);
cache-file quoted_string; // test option
suppress-initial-notify boolean; \
 // not yet implemented
preferred-glue string;
dual-stack-servers [port integer] {
 (quoted_string [port integer] |
 ipv4_address [port integer] |
 ipv6_address [port integer]); ...
};
edns-udp-size integer;
max-udp-size integer;
root-delegation-only [exclude
 { quoted_string; ... }];
disable-algorithms string { string; ... };
dnssec-enable boolean;
dnssec-validation boolean;
dnssec-lookaside string trust-anchor string;
dnssec-must-be-secure string boolean;
dnssec-accept-expired boolean;
empty-server string;
empty-contact string;
empty-zones-enable boolean;
disable-empty-zone string;
dialup dialuptype;
ixfr-from-differences ixfrdiff;
allow-query { address_match_element; \
 ... };
allow-query-on { address_match_element; \
 ... };
allow-query-cache { address_match_element; \
 ... };
```

```

allow-query-cache-on { address_match_element; \
 ... };
allow-transfer { address_match_element; \
 ... };
allow-update { address_match_element; \
 ... };
allow-update-forwarding { address_match_element; \
 ... };
update-check-ksk boolean;
masterfile-format (text | raw);
notify notifytype;
notify-source (ipv4_address | *) \
 [port (integer | *)];
notify-source-v6 (ipv6_address | *) \
 [port (integer | *)];
notify-delay seconds;
notify-to-soa boolean;
also-notify [port integer] \
 { (ipv4_address | ipv6_address \
 [port integer]; ... };
allow-notify { address_match_element; ... };
forward (first | only);
forwarders [port integer] {
 (ipv4_address | ipv6_address) [port integer]; ...
};
max-journal-size size_no_default;
max-transfer-time-in integer;
max-transfer-time-out integer;
max-transfer-idle-in integer;
max-transfer-idle-out integer;
max-retry-time integer;
min-retry-time integer;
max-refresh-time integer;
min-refresh-time integer;
multi-master boolean;
sig-validity-interval integer;
sig-re-signing-interval integer;
sig-signing-nodes integer;
sig-signing-signatures integer;
sig-signing-type integer;
transfer-source (ipv4_address | *) \
 [port (integer | *)];
transfer-source-v6 (ipv6_address | *) \
 [port (integer | *)];
alt-transfer-source (ipv4_address | *) \
 [port (integer | *)];
alt-transfer-source-v6 (ipv6_address | *) \

```

```

 [port (integer | *)];
 use-alt-transfer-source boolean;
 zone-statistics boolean;
 key-directory quoted_string;
 try-tcp-refresh boolean;
 zero-no-soa-ttl boolean;
 zero-no-soa-ttl-cache boolean;
 nsec3-test-zone boolean; // testing only
 allow-v6-synthesis { address_match_element; ... }; \
 // obsolete
 deallocate-on-exit boolean; // obsolete
 fake-iquery boolean; // obsolete
 fetch-glue boolean; // obsolete
 has-old-clients boolean; // obsolete
 maintain-ixfr-base boolean; // obsolete
 max-ixfr-log-size size; // obsolete
 multiple-cnames boolean; // obsolete
 named-xfer quoted_string; // obsolete
 serial-queries integer; // obsolete
 treat-cr-as-space boolean; // obsolete
 use-id-pool boolean; // obsolete
};

View view string optional_class {
 match-clients { address_match_element; ... };
 match-destinations { address_match_element; ... };
 match-recursive-only boolean;
 key string {
 algorithm string;
 secret string;
 };
 zone string optional_class {
 ...
 };
 server (ipv4_address[/prefixlen] | ipv6_address[/prefixlen]) {
 ...
 };
 trusted-keys {
 string integer integer integer quoted_string; ...
 };
 allow-recursion { address_match_element; ... };
 allow-recursion-on { address_match_element; ... };
 sortlist { address_match_element; ... };
 topology { address_match_element; ... }; // not implemented
 auth-nxdomain boolean; // default changed
 minimal-responses boolean;
 recursion boolean;
 rrset-order {

```

```

 [class string] [type string]
 [name quoted_string] string string; ...
};
provide-ixfr boolean;
request-ixfr boolean;
rfc2308-type1 boolean; // not yet implemented
additional-from-auth boolean;
additional-from-cache boolean;
query-source ((ipv4_address | *) | [address \
 (ipv4_address | *)]) [port (integer | *)];
query-source-v6 ((ipv6_address | *) | [address \
 (ipv6_address | *)]) [port (integer | *)];
use-queryport-pool boolean;
queryport-pool-ports integer;
queryport-pool-updateinterval integer;
cleaning-interval integer;
min-roots integer; // not implemented
lame-ttl integer;
max-ncache-ttl integer;
max-cache-ttl integer;
transfer-format (many-answers | one-answer);
max-cache-size size;
max-acache-size size;
clients-per-query number;
max-clients-per-query number;
check-names (master | slave | response) \
 (fail | warn | ignore);
check-mx (fail | warn | ignore);
check-integrity boolean;
check-mx-cname (fail | warn | ignore);
check-srv-cname (fail | warn | ignore);
cache-file quoted_string; // test option
suppress-initial-notify boolean; // not yet implemented
preferred-glue string;
dual-stack-servers [port integer] {
 (quoted_string [port integer] |
 ipv4_address [port integer] |
 ipv6_address [port integer]); ...
};
edns-udp-size integer;
max-udp-size integer;
root-delegation-only [exclude { quoted_string; ... }];
disable-algorithms string { string; ... };
dnssec-enable boolean;
dnssec-validation boolean;
dnssec-lookaside string trust-anchor string;
dnssec-must-be-secure string boolean;

```

```
dnssec-accept-expired boolean;
empty-server string;
empty-contact string;
empty-zones-enable boolean;
disable-empty-zone string;
dialup dialuptype;
ixfr-from-differences ixfrdiff;
allow-query { address_match_element; ... };
allow-query-on { address_match_element; ... };
allow-query-cache {
address_match_element; ... };
allow-query-cache-on { address_match_element; ... };
allow-transfer { address_match_element; ... };
allow-update { address_match_element; ... };
allow-update-forwarding { address_match_element; ... };
update-check-ksk boolean;
masterfile-format (text | raw);
notify notifytype;
notify-source (ipv4_address | *) \
 [port (integer | *)];
notify-source-v6 (ipv6_address | *) \
 [port (integer | *)];
notify-delay seconds;
notify-to-soa boolean;
also-notify [port integer] { (ipv4_address | \
 ipv6_address) [port integer]; ... };
allow-notify { address_match_element; ... };
forward (first | only);
forwarders [port integer] \
 { (ipv4_address | ipv6_address) \
 [port integer]; ...
};
max-journal-size size_no_default;
max-transfer-time-in integer;
max-transfer-time-out integer;
max-transfer-idle-in integer;
max-transfer-idle-out integer;
max-retry-time integer;
min-retry-time integer;
max-refresh-time integer;
min-refresh-time integer;
multi-master boolean;
sig-validity-interval integer;
transfer-source (ipv4_address | *) \
 [port (integer | *)];
transfer-source-v6 (ipv6_address | *) \
 [port (integer | *)];
```

```

alt-transfer-source (ipv4_address | *)\
 [port (integer | *)];
alt-transfer-source-v6 (ipv6_address | *)\
 [port (integer | *)];
use-alt-transfer-source boolean;
zone-statistics boolean;
try-tcp-refresh boolean;
key-directory quoted_string;
zero-no-soa-ttl boolean;
zero-no-soa-ttl-cache boolean;
allow-v6-synthesis { address_match_element; ... };
 // obsolete
fetch-glue boolean; // obsolete
maintain-ixfr-base boolean; // obsolete
max-ixfr-log-size size; // obsolete
};

Zone zonestring optional_class {
 type (master | slave | stub | hint |
 forward | delegation-only);
 file quoted_string;
 masters [port integer] \{
 (masters |
 ipv4_address [port integer] |
 ipv6_address [port integer]) [key string]; ...
 };
 database string;
 delegation-only boolean;
 check-names (fail | warn | ignore);
 check-mx (fail | warn | ignore);
 check-integrity boolean;
 check-mx-cname (fail | warn | ignore);
 check-srv-cname (fail | warn | ignore);
 dialup dialuptype;
 ixfr-from-differences boolean;
 journal quoted_string;
 zero-no-soa-ttl boolean;
 allow-query { address_match_element; ... };
 allow-query-on { address_match_element; ... };
 allow-transfer { address_match_element; ... };
 allow-update { address_match_element; ... };
 allow-update-forwarding { address_match_element; ... };
 update-policy {
 (grant | deny) string
 (name | subdomain | wildcard | self | selfsub |
 selfwild | krb5-self | ms-self | krb5-subdomain |
 ms-subdomain | tcp-self | 6to4-self) string
 rrtypelist; ...
 }
};

```

```
};
update-check-ksk boolean;
masterfile-format (text | raw);
notify notifytype;
notify-source (ipv4_address | *) [port (integer | *)];
notify-source-v6 (ipv6_address | *) [port (integer | *)];
notify-delay seconds;
notify-to-soa boolean;
also-notify [port integer] { (ipv4_address | ipv6_address)
 [port integer]; ... };
allow-notify { address_match_element; ... };
forward (first | only);
forwarders [port integer] {
 (ipv4_address | ipv6_address) [port integer]; ...
};
max-journal-size size_no_default;
max-transfer-time-in integer;
max-transfer-time-out integer;
max-transfer-idle-in integer;
max-transfer-idle-out integer;
max-retry-time integer;
min-retry-time integer;
max-refresh-time integer;
min-refresh-time integer;
multi-master boolean;
sig-validity-interval integer;
transfer-source (ipv4_address | *)
 [port (integer | *)];
transfer-source-v6 (ipv6_address | *)
 [port (integer | *)];
alt-transfer-source (ipv4_address | *)
 [port (integer | *)];
alt-transfer-source-v6 (ipv6_address | *)
 [port (integer | *)];
use-alt-transfer-source boolean;
zone-statistics boolean;
try-tcp-refresh boolean;
key-directory quoted_string;
nsec3-test-zone boolean; // testing only
ixfr-base quoted_string; // obsolete
ixfr-tmp-file quoted_string; // obsolete
maintain-ixfr-base boolean; // obsolete
max-ixfr-log-size size; // obsolete
pubkey integer integer integer quoted_string; // obsolete
};
```



**See Also** [named\(1M\)](#), [named-checkconf\(1M\)](#), [rndc\(1M\)](#)

*BIND 9 Administrator Reference Manual*

**Name** ncad\_addr – name of the Solaris Network Cache and Accelerator (NCA) socket utility library

**Synopsis** /usr/lib/ncad\_addr.so

**Description** ncad\_addr.so is the Solaris Network Cache and Accelerator (NCA) socket utility library. Use this library with a web server to avoid support for the PF\_NCA family type socket. The web server can take advantage of NCA functionality.

Interpose the ncad\_addr interfaces before the interfaces in libsocket by setting the environment variable LD\_PRELOAD to ncad\_addr.so so that it is preloaded before libsocket.so.1. The ncad\_addr.so interfaces will be interposed only if NCA is enabled. See [ncakmod\(1\)](#).

**Examples** EXAMPLE 1 Interposing ncad\_addr

Using Bourne shell syntax as an example, set LD\_PRELOAD as shown below to interpose the ncad\_addr socket utility library:

```
LD_PRELOAD=/usr/lib/ncad_addr.so /usr/bin/httpd
```

**Files** /usr/lib/ncad\_addr.so ncad\_addr socket utility library shared object

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                                |
|---------------------|------------------------------------------------|
| Availability        | system/network/http-cache-accelerator (32-bit) |
|                     | SUNWncarx (64-bit)                             |
| Interface Stability | Uncommitted                                    |

**See Also** [ncab2clf\(1\)](#), [ncakmod\(1\)](#), [socket\(3SOCKET\)](#), [nca.if\(4\)](#), [ncakmod.conf\(4\)](#), [attributes\(5\)](#), [nca\(7d\)](#)

**Notes** Only applications that use the NCA feature, for example, web servers, should interpose this library.

**Name** nca.if – the NCA configuration file that specifies physical interfaces

**Synopsis** /etc/nca/nca.if

**Description** Specify the physical interfaces for which the Solaris Network Cache and Accelerator (“NCA”) feature will be configured in the `nca.if` configuration file. List the physical interfaces in the file, one per line. To configure NCA to listen on all physical interfaces present on the system backed by a `hostname.{interface_name}`, then list only an asterisk (“\*”) in `nca.if`.

When the `nca` [nca](#) `mod(1)` initialization script is invoked during system boot, it will attempt to configure each physical interface specified in the `nca.if` file by using `ncaconfd(1M)`. Note that there must be an accompanying `hostname.{interface_name}` file and an entry in `/etc/hosts` for the contents of `hostname.{interface_name}`.

You must reboot in order to implement changes to the `nca.if` file.

### Examples

x86 **EXAMPLE 1** `nca.if` on x86

The following is an example of an `nca.if` file that would be used on an x86 system:

```
iprb1
iprb6
iprb8
```

SPARC **EXAMPLE 2** `nca.if` on SPARC

The following is an example of an `nca.if` file that would be used on a SPARC system:

```
hme2
hme3
hme4
```

All Platforms **EXAMPLE 3** Configuring NCA to Listen on All Physical Interfaces

The following example shows the contents of an `nca.if` file that would be used to configure either platform to listen on all physical interfaces present on the system:

```
*
```

|              |                                  |                                                                                                                                                        |
|--------------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Files</b> | <code>/etc/nca/nca.if</code>     | Lists the physical interfaces on which NCA will run.                                                                                                   |
|              | <code>/etc/hostname.{0-9}</code> | Lists all physical interfaces configured on the server.                                                                                                |
|              | <code>/etc/hosts</code>          | Lists all host names associated with the server. Entries in this file must match with entries in <code>/etc/hostname.{0-9}</code> for NCA to function. |

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                       |
|---------------------|---------------------------------------|
| Availability        | system/network/http-cache-accelerator |
| Interface Stability | Committed                             |

**See Also** [ncab2clf\(1\)](#), [ncakmod\(1\)](#), [ifconfig\(1M\)](#), [ncakmod.conf\(4\)](#), [ncalogd.conf\(4\)](#), [attributes\(5\)](#), [nca\(7d\)](#)

*Oracle Solaris Administration: IP Services*

**Name** ncakmod.conf – ncakmod configuration file

**Synopsis** /etc/nca/ncakmod.conf

**Description** The ncakmod.conf file is used to configure the Solaris Network Cache and Accelerator (NCA) kernel module. The file contains two fields, key and value.

The `status` key is used to indicate if the user wants to have NCA turned on as a feature. If the value of `status` key is `enabled`, then the NCA kernel module will be pushed on to the specified interfaces. If the value of the `status` key is `disabled`, then the NCA kernel module will not be pushed on to any interfaces. The default is `disabled`.

The `httpd_door_path` key specifies the path name of the Solaris Door RPC mechanism that will be used to communicate with the `http` daemon. The default value is `/var/run/nca_httpd_1.door`.

Use the `nca_active` key to indicate whether to allow NCA to actively open outgoing TCP connections. The default value for `nca_active` is `disabled`. If set to `enabled`, `ncaconfd` sets up NCA for each interface and then operates as a daemon, allowing NCA to make outgoing TCP connections. This functionality is possible only by using the `doors` interface to NCA. A web server that uses the `sockets` interface with `PF_NCA` or `ncad_addr`. so cannot connect by means of `nca_active`.

NCA supports the logging of in-kernel cache hits. See [ncalogd.conf\(4\)](#). NCA stores logs in a binary format. Use the [ncab2clf\(1\)](#) utility to convert the log from a binary format to the Common Log File format.

In order to implement changes to the ncakmod.conf file, you will need to reboot.

**Examples** EXAMPLE 1 A Sample ncakmod.conf File

The following is a sample ncakmod.conf file:

```
#
NCA Kernel Module Configuration File
#
status=disabled
httpd_door_path=/var/run/nca_httpd_1.door
nca_active=disabled
```

**Files** /etc/nca/ncakmod.conf The NCA kernel module configuration file.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                       |
|---------------------|---------------------------------------|
| Availability        | system/network/http-cache-accelerator |
| Interface Stability | Committed                             |

**See Also** [ncab2clf\(1\)](#), [ncakmod\(1\)](#), [door\\_create\(3C\)](#), [nca.if\(4\)](#), [ncad\\_addr\(4\)](#), [ncalogd.conf\(4\)](#), [attributes\(5\)](#), [nca\(7d\)](#)

*Oracle Solaris Administration: IP Services*

**Name** ncaologd.conf – NCA logging configuration file

**Synopsis** /etc/nca/ncaologd.conf

**Description** The `ncaologd.conf` is used to configure Solaris Network Cache and Accelerator (“NCA”) logging. The file contains two fields, `key` and `value`.

The `status` key is used to indicate if the user wants to have NCA logging turned on. If the value of `status` key is `enabled`, then NCA logging will be turned on. If the value of the `status` key is `disabled`, then NCA logging will not be invoked. The default value is `disabled`.

The `logd_path_name` key specifies the absolute pathname of the log file. The log file must be a raw device without a filesystem or a file on a local file system. The default value is `/var/nca/log`. `logd_path_name` can also contain a whitespace-delimited list of values for multiple log files to a maximum of 16. If you specify multiple log files, you must enclose the list in quotation marks (“”). With multiple files, NCA logging moves to the next file on the list once the file size specified by `logd_file_size` has been reached. When the last file is full, NCA logging rotates back to the first file in the list. A pointer to the current log file is stored in `/var/nca/current`.

The `logd_file_size` key specifies the value of the file size, in bytes, allowed for each log file specified in by the `logd_path_name` key. The default value is 1000000 bytes.

In order to implement changes to the `ncaologd.conf` file, you will need to stop and start NCA logging or reboot.

NCA stores logs in a binary format. Use the `ncab2clf(1)` utility to convert the log from a binary format to the Common Log File format.

**Examples** EXAMPLE 1 A Sample `ncaologd.conf` File

The following is a sample `ncaologd.conf` file that specifies three log files:

```
#
NCA Log Daemon Configuration File
#

status=enabled
logd_path_name="/var/nca/log1 /var/nca/log2 /var/nca/log3"
logd_file_size=1000000
```

Note that there is no NCA logging daemon. Logging is performed as one of the functions of the NCA software.

**Files** /etc/nca/ncaologd.conf Lists configuration parameters for NCA logging.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                       |
|---------------------|---------------------------------------|
| Availability        | system/network/http-cache-accelerator |
| Interface Stability | Committed                             |

**See Also** [ncab2clf\(1\)](#), [ncakmod\(1\)](#), [dd\(1M\)](#), [door\\_create\(3C\)](#), [nca.if\(4\)](#), [ncakmod.conf\(4\)](#), [attributes\(5\)](#), [nca\(7d\)](#)

*Oracle Solaris Administration: IP Services*



**Name** ncaport.conf – ncaport configuration file

**Synopsis** /etc/nca/ncaport.conf

**Description** The ncaport.conf file is used to configure the IP addresses and ports that the Solaris Network Cache and Acceleration (NCA) kernel module services. The file contains two fields, key and value, in the format of ncaport=*ipaddress/port*. IPv4 addresses must be in the dot notation *d.d.d.d*. IPv6 addresses must be in one of the three conventional forms (see [inet\\_pton\(3SOCKET\)](#)). If an asterisk (\*) is used for an IP address, it is interpreted as INADDR\_ANY, which matches any IP address.

A web server uses the environment variable LD\_PRELOAD and the ncaport.conf configuration file to convert an AF\_INET socket to an AF\_NCA socket. LD\_PRELOAD enables the NCA socket utility library to be loaded before libsocket.so.1. See the [ncad\\_addr\(4\)](#) for details. When a web server issues the [bind\(3SOCKET\)](#) system call, it is intercepted by the interposition library ncad\_addr.so. If the bind address is in the ncaport.conf file, the AF\_INET socket is converted to a AF\_NCA socket.

**Examples** EXAMPLE 1 Sample ncaport.conf File

The following is a sample ncaport.conf file:

```
#
NCA Kernel Module Port Configuration File
#
ncaport=1080:0:0:0:8:800:200C:417A/100
ncaport=192.168.84.71/80
ncaport=*/9000
```

**See Also** [bind\(3SOCKET\)](#), [inet\\_pton\(3SOCKET\)](#), [ncad\\_addr\(4\)](#), [attributes\(5\)](#), [nca\(7d\)](#)

**Notes** For those web servers that use AF\_NCA sockets, the NCA port configuration described here has no effect.

NCA does not currently support IPv6. Any IPv6 addresses in the file ncaport.conf are ignored.

**Name** ndmp – configuration properties for Solaris Network Data Management Protocol (NDMP) server

**Description** The behavior of the Solaris NDMP server is specified by property values that are stored in the Service Management Facility, [smf\(5\)](#).

An authorized user can use the [ndmpadm\(1M\)](#) command to set global values for these properties in SMF.

The `ndmpd` service requires a restart by [svcadm\(1M\)](#) whenever properties are set. This restart is required whether the properties are set by [ndmpadm\(1M\)](#) or [svccfg\(1M\)](#).

*tcp-port* is the TCP port number that the NDMP service accepts for incoming connections. The default is 10000.

You can set the following properties by using the `ndmpadm set` command:

|                                   |                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>backup-quarantine</code>    | Backup the files marked as quarantined by AV. Acceptable values are yes or no. The default is no.                                                                                                                                                                                                                                                                                 |
| <code>dar-support</code>          | Set the Direct Access Recovery mode. Acceptable values are yes or no. The default is no.                                                                                                                                                                                                                                                                                          |
| <code>debug-level</code>          | Set the debug level. The debug-level can be set to either 0 (off) or 1 (on). The default is 0.                                                                                                                                                                                                                                                                                    |
| <code>debug-path</code>           | The path to which to save the debug log. The default is <code>/var/ndmp</code> .                                                                                                                                                                                                                                                                                                  |
| <code>drive-type</code>           | Choose either SystemV or BSD tape drives for the data management application. Valid values are <code>sysv</code> and <code>bsd</code> . The default is <code>sysv</code> .                                                                                                                                                                                                        |
| <code>dump-pathnode</code>        | Enable or disable backing up the directories containing modified files or directories in <a href="#">dump(1)</a> backup format. Acceptable values are yes or no. The default is no.                                                                                                                                                                                               |
| <code>ignore-ctime</code>         | Determines whether the change timestamp ( <code>ctime</code> ) of files and directories is used to determine whether a file should be backed up in level backup. If this parameter is set to yes, only the modification time ( <code>mtime</code> ) of the file or directory determines whether it should be backed up. Acceptable values are yes or no. The default value is no. |
| <code>overwrite-quarantine</code> | Restore quarantined files on top of current files if they already exist. Acceptable values are yes or no. The default value is no.                                                                                                                                                                                                                                                |
| <code>restore-quarantine</code>   | Restore the files that had been marked as quarantined by AV and are backed up. Acceptable values are yes or no. The default value is no.                                                                                                                                                                                                                                          |

|                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>tar-pathnode</code>       | Enable or disable backing up the directories containing modified files or directories in <code>tar(1)</code> backup format. Acceptable values are <code>yes</code> or <code>no</code> . The default value is <code>no</code> .                                                                                                                                                                                                                                                   |
| <code>token-maxseq</code>       | Set the maximum sequence number for subsequent token-based incremental backup in NDMP-V4. The default value is 9. There are two limits for this value: <code>soft-limit</code> , which is 59, and <code>hard-limit</code> , equal to 64. If the token sequence number, passed by the DMA, is between the soft and hard limits, a warning message is issued to the DMA. The token sequence number can never exceed the <code>hard-limit</code> value.                             |
| <code>type-override</code>      | Override the value of <code>TYPE</code> . Valid values are <code>zfs</code> and <code>off</code> . By default, <code>type-override</code> has a value of <code>off</code> and does not override <code>TYPE</code> . Invalid values are treated as <code>off</code> .<br><br>If <code>type-override</code> is set to <code>zfs</code> , the user must make sure that the path format used in the data management application is consistent with the <code>zfs</code> backup type. |
| <code>version</code>            | Set the maximum active NDMP protocol version. Valid values are currently 2, 3, and 4. The default is 4.                                                                                                                                                                                                                                                                                                                                                                          |
| <code>zfs-force-override</code> | Override the value of <code>ZFS_FORCE</code> . <code>yes</code> forces a value of <code>y</code> . <code>no</code> forces a value of <code>n</code> . By default, <code>zfs-force-override</code> has a value of <code>off</code> and does not override <code>ZFS_FORCE</code> .                                                                                                                                                                                                 |

The following property can only be set when using the `ndmpadm enable` or `ndmpadm disable` command:

`auth-type` Sets the password encryption type for the authentication of local users. Valid values are `cram-md5` or `cleartext`.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE      |
|---------------------|----------------------|
| Availability        | service/storage/ndmp |
| Interface Stability | Committed            |

**See Also** [dump\(1\)](#), [tar\(1\)](#), [ndmpadm\(1M\)](#), [ndmpd\(1M\)](#), [svcadm\(1M\)](#), [svccfg\(1M\)](#), [attributes\(5\)](#), [smf\(5\)](#)

**Name** ndpd.conf – configuration file for IPv6 router autoconfiguration

**Synopsis** /etc/inet/ndpd.conf

**Description** The `ndpd.conf` file contains configuration information for `in.ndpd(1M)`. On a host, this file does not need to exist or can be empty. The file has one configuration entry per line; note that lines can be extended with a backslash (\) followed by a NEWLINE. There are four forms of configuration entries which are identified by the first field on the line: `ifdefault`, `prefixdefault`, `if`, or `prefix`. The `ifdefault` and `if` entries set interface configuration variables. The former establishes the routing behavior for all interfaces, the latter sets per-interface parameters. Any `ifdefault` entries must precede any `if` entries in the file.

The `prefixdefault` and `prefix` entries control prefix configuration variables. `prefixdefault` establishes the default behavior for all prefix advertisements on all interfaces. The `prefix` keyword advertises per-prefix information. Any `prefixdefault` entries must precede any `prefix` entries in the file.

Each `ifdefault` entry is composed of a single line of the form:

```
ifdefault [if-variable-name value]*
```

Each `if` entry is composed of a single line of the form:

```
if interface [if-variable-name value]*
```

Each `prefixdefault` entry is composed of a single line of the form:

```
prefixdefault [prefix-variable-name value]*
```

Each `prefix` entry is composed of a single line of the form:

```
prefix prefix/prefix_length interface [prefix-variable-name value]*
```

Fields are separated by either SPACE or TAB characters. A '#' (number sign) indicates the beginning of a comment. Characters up to the end of the line are not interpreted by routines that search this file.

*interface*                   The name of a network interface, for example, `eri0`.

*prefix*                     An IPv6 address in standard hexadecimal notation, for example, `fec0:0:0:1::0`.

*prefix\_length*             A number between 0 and 128.

*if-variable-name*           An interface variable. Below is the list of interface variables applicable to routers only along with their default values and units as discussed in *RFC 2461* and *RFC 2462*. The `Tmp*` variables apply to hosts and routers. The `Tmp*` variables configure temporary address functionality as defined in *RFC 3041*.

| Variable Name          | Default   | Unit         |
|------------------------|-----------|--------------|
| AdvSendAdvertisements  | false     | Boolean      |
| DupAddrDetectTransmits | 1         | Counter      |
| MaxRtrAdvInterval      | 600       | Seconds      |
| MinRtrAdvInterval      | 200       | Seconds      |
| AdvManagedFlag         | false     | Boolean      |
| AdvOtherConfigFlag     | false     | Boolean      |
| AdvLinkMTU             | 0         | Bytes        |
| AdvReachableTime       | 0         | Milliseconds |
| AdvRetransTimer        | 0         | Milliseconds |
| AdvCurHopLimit         | see below | Counter      |
| AdvDefaultLifetime     | 1800      | Seconds      |

These variables are described as follows:

|                        |                                                                                                                                                                                                                                                 |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AdvSendAdvertisements  | Indicates whether the node should send out advertisements and respond to router solicitations. You need to explicitly configure this value to turn on router advertisement functions.                                                           |
| DupAddrDetectTransmits | Defines the number of consecutive Neighbor Solicitation messages that the Neighbor Discovery protocol should send during Duplicate Address Detection of the local node's address.                                                               |
| MaxRtrAdvInterval      | Specifies the maximum time to wait between sending unsolicited multicast advertisements.                                                                                                                                                        |
| MinRtrAdvInterval      | Specifies the minimum amount of time to wait between sending unsolicited multicast advertisements.                                                                                                                                              |
| AdvManagedFlag         | Indicates the value to be placed in the "Manage address configuration" flag in the Router Advertisement. This flag causes hosts to run DHCPv6 to acquire addresses and other configuration information. This flag causes hosts to run DHCPv6 to |

|                    |                                                                                                                                                                        |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                    | acquire configuration information, but only if AdvManagedFlag is not set.                                                                                              |
| AdvOtherConfigFlag | Indicates the value to be placed in the “Other stateful configuration” flag in the Router Advertisement.                                                               |
| AdvLinkMTU         | Specifies an MTU value to be sent by the router. The default of zero indicates that the router does not specify MTU options.                                           |
| AdvReachableTime   | Specifies the value in the Reachable Time field in the advertisement messages sent by the router.                                                                      |
| AdvRetransTimer    | Specifies the value in the Retrans Timer field in the advertisement messages sent by the router.                                                                       |
| AdvCurHopLimit     | Specifies the value to be placed in the current hop limit field in the advertisement messages sent by the router. The default is the current diameter of the Internet. |
| AdvDefaultLifetime | Specifies the default lifetime of the router advertisements.                                                                                                           |

Listed below is the interface variable that applies to both hosts and routers.

| Variable Name        | Default                                                                                                                      | Unit    |
|----------------------|------------------------------------------------------------------------------------------------------------------------------|---------|
| StatefulAddrConf     | true                                                                                                                         | Boolean |
| StatelessAddrConf    | true                                                                                                                         | Boolean |
| TmpAddrsEnabled      | false                                                                                                                        | Boolean |
| TmpValidLifetime     | 604800<br>(1 week)                                                                                                           | Seconds |
| TmpPreferredLifetime | 86400<br>(1 day)                                                                                                             | Seconds |
| TmpRegenAdvance      | 5                                                                                                                            | Seconds |
| TmpMaxDesyncFactor   | 600                                                                                                                          | Seconds |
| StatefulAddrConf     | Controls whether the system configures its IPv6 addresses by means of the Stateful Address Autoconfiguration mechanism, also |         |

---

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | known as DHCPv6, as described in RFC 3315. If enabled (the default), hosts automatically run DHCPv6 based on the “managed” and “other” flags sent by routers. If disabled, <code>in.ndpd</code> will not invoke DHCPv6 automatically. DHCPv6 can still be invoked manually by using <code>ifconfig(1M)</code> , in which case <code>in.ndpd</code> automatically sets the prefix length as needed.                                                                                                                                                                             |
| StatelessAddrConf    | Controls whether the system configures its IPv6 addresses by means of the Stateless Address Autoconfiguration mechanism described in <i>RFC 2462</i> . If enabled hosts (the default) autoconfigure addresses based on prefixes advertised by routers, routers will only autoconfigure addresses based on the prefixes they advertise themselves. In other words, even when enabled, routers do not autoconfigure addresses based on prefixes that other routers advertise. If you specify <code>false</code> for this variable, then the address must be configured manually. |
| TmpAddrsEnabled      | Indicates whether a temporary address should be created for all interfaces or for a particular interface of a node.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| TmpValidLifetime     | Sets the valid lifetime for a temporary address.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| TmpPreferredLifetime | Sets the preferred lifetime of a temporary address.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| TmpRegenAdvance      | Specifies the lead time in advance of address deprecation for generation of a new temporary address.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| TmpMaxDesyncFactor   | Sets the upper bound on the <code>DesyncFactor</code> , which is a random value that is used to shorten the preferred lifetime so that clients do not                                                                                                                                                                                                                                                                                                                                                                                                                          |

regenerate an address at the same time.

*prefix-variable-name*

A prefix variable as discussed in *RFC 2461* and *RFC 2462*. The following lists the each interface variable and its default value and unit:

| Variable Name          | Default | Unit      |
|------------------------|---------|-----------|
| AdvValidLifetime       | 2592000 | Seconds   |
| AdvOnLinkFlag          | true    | Boolean   |
| AdvPreferredLifetime   | 604800  | Seconds   |
| AdvAutonomousFlag      | true    | Boolean   |
| AdvValidExpiration     | not set | Date/Time |
| AdvPreferredExpiration | not set | Date/Time |

These variables are described as follows:

|                        |                                                                                                        |
|------------------------|--------------------------------------------------------------------------------------------------------|
| AdvValidLifetime       | Specifies the valid lifetime of the prefix that is being configured.                                   |
| AdvOnLinkFlag          | Specifies the value to be placed in the on-link flag (“L-bit”) field in the Prefix Information option. |
| AdvPreferredLifetime   | Specifies the value to be placed in the Preferred Lifetime in the Prefix Information option.           |
| AdvAutonomousFlag      | Specifies the value to be placed in the Autonomous Flag field in the Prefix Information option.        |
| AdvValidExpiration     | Specifies the valid expiration date of the prefix.                                                     |
| AdvPreferredExpiration | Specifies the preferred expiration date of the prefix.                                                 |

The AdvValidExpiration and AdvPreferredExpiration variables are used to specify that the lifetime should be decremented in real time as specified in *RFC 2461*. If an Expiration variable is set, it takes precedence over the corresponding AdvValidLifetime or AdvPreferredLifetime variable setting.



*value* The value is a function of the unit. Boolean values are `true`, `false`, `on`, `off`, `1`, or `0`.

Values in seconds can have characters appended for day (d), hour (h), minute (m) and second (s). The default is seconds. For example, `1h` means 1 hour. This is equivalent to the value `3600`.

Values in milliseconds can have characters appended for day (d), hour (h), minute (m) second (s), and millisecond (ms). The default is milliseconds. For example, `1h` is equivalent to the value `3600000`.

Date/time values are strings that use the recommended ISO date format described as “%Y-%m-%d %R”, which represents a 4 digit year, a dash character, a numeric month, a dash character, and a numeric day of the month, followed by one or more whitespace characters and finally a 24 hour clock with hours, a colon, and minutes. For example, `1999-01-31 20:00` means 8pm January 31 in 1999. Since the date/time values contain a space, use single or double quotes to declare the value. For example:

```
prefixdefault AdvPreferredExpiration '1999-01-31 20:00'
```

**Examples** EXAMPLE 1 Sending Router Advertisements for all Interfaces

The following example can be used to send router advertisements out to all interfaces:

```
Send router advertisements out all interfaces
ifdefault AdvSendAdvertisements on
prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on

Advertise a (bogus) global prefix and a site
local prefix on three interfaces using the default lifetimes
prefix 2:0:0:9255::0/64 eri0
prefix fec0:0:0:9255::0/64 eri0

prefix 2:0:0:9256::0/64 eri1
prefix fec0:0:0:9256::0/64 eri1

prefix 2:0:0:9259::0/64 eri2
prefix fec0:0:0:9259::0/64 eri2
```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Availability   | system/core-os  |

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Committed       |

**See Also** [dhcpageant\(1M\)](#), [ifconfig\(1M\)](#), [in.ndpd\(1M\)](#), [routeadm\(1M\)](#), [attributes\(5\)](#), [icmp6\(7P\)](#), [ip6\(7P\)](#)

Narten, T., Nordmark, E., and Simpson, W. *RFC 2461, Neighbor Discovery for IP Version 6 (IPv6)*. The Internet Society. December 1998.

Thomson, S., and 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.

Droms, R. *RFC 3315, Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*. Cisco Systems. July 2003.

*Oracle Solaris Administration: IP Services*

**Name** netconfig – network configuration database

**Synopsis** /etc/netconfig

**Description** The network configuration database, `/etc/netconfig`, is a system file used to store information about networks that are connected to the system. The `netconfig` database and the routines that access it (see [getnetconfig\(3NSL\)](#)) are part of the Network Selection component. The Network Selection component also includes [getnetpath\(3NSL\)](#) routines to provide application-specific network search paths. These routines access the `netconfig` database based on the environment variable `NETPATH`. See [environ\(5\)](#).

`netconfig` contains an entry for each network available on the system. Entries are separated by newlines. Fields are separated by whitespace and occur in the order in which they are described below. Whitespace can be embedded as “`\blank`” or “`\tab`”. Backslashes may be embedded as “`\\`”. Lines in `/etc/netconfig` that begin with a # (hash) in column 1 are treated as comments.

Each of the valid lines in the `netconfig` database correspond to an available transport. Each entry is of the form:

```
network ID semantics flag protocol-family \
protocol-name network-device translation-libraries
```

*network ID* A string used to uniquely identify a network. *network ID* consists of non-null characters, and has a length of at least 1. No maximum length is specified. This namespace is locally significant and the local system administrator is the naming authority. All *network IDs* on a system must be unique.

*semantics* The *semantics* field is a string identifying the “semantics” of the network, that is, the set of services it supports, by identifying the service interface it provides. The *semantics* field is mandatory. The following semantics are recognized.

|                           |                                                                              |
|---------------------------|------------------------------------------------------------------------------|
| <code>tpi_clts</code>     | Transport Provider Interface, connectionless                                 |
| <code>tpi_cots</code>     | Transport Provider Interface, connection oriented                            |
| <code>tpi_cots_ord</code> | Transport Provider Interface, connection oriented, supports orderly release. |

*flag* The *flag* field records certain two-valued (“true” and “false”) attributes of networks. *flag* is a string composed of a combination of characters, each of which indicates the value of the corresponding attribute. If the character is present, the attribute is “true.” If the character is absent, the attribute is “false.” “-” indicates that none of the attributes are present. Only one character is currently recognized:

- v Visible (“default”) network. Used when the environment variable `NETPATH` is unset.

*protocol family*

The *protocol family* and *protocol name* fields are provided for protocol-specific applications. The *protocol family* field contains a string that identifies a protocol family. The *protocol family* identifier follows the same rules as those for *network IDs*; the string consists of non-null characters, it has a length of at least 1, and there is no maximum length specified. A "-" in the *protocol family* field indicates that no protocol family identifier applies (the network is experimental). The following are examples:

|           |                                                                     |
|-----------|---------------------------------------------------------------------|
| loopback  | Loopback (local to host).                                           |
| inet      | Internetwork: UDP, TCP, and the like.                               |
| inet6     | Internetwork over IPv6: UDP, TCP, and the like.                     |
| imlink    | ARPANET imp addresses                                               |
| pup       | PUP protocols: for example, BSP                                     |
| chaos     | MIT CHAOS protocols                                                 |
| ns        | XEROX NS protocols                                                  |
| nbs       | NBS protocols                                                       |
| ecma      | European Computer Manufacturers Association                         |
| datakit   | DATAKIT protocols                                                   |
| ccitt     | CCITT protocols, X.25, and the like.                                |
| sna       | IBM SNA                                                             |
| decnet    | DECNET                                                              |
| dli       | Direct data link interface                                          |
| lat       | LAT                                                                 |
| hlink     | NSC Hyperchannel                                                    |
| appletalk | Apple Talk                                                          |
| nit       | Network Interface Tap                                               |
| ieee802   | IEEE 802.2; also ISO 8802                                           |
| osi       | Umbrella for all families used by OSI (for example, protosw lookup) |
| x25       | CCITT X.25 in particular                                            |
| osinet    | AFI = 47, IDI = 4                                                   |
| gossip    | U.S. Government OSI                                                 |

|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>protocol name</i>         | <p>The <i>protocol name</i> field contains a string that identifies a protocol. The <i>protocol name</i> identifier follows the same rules as those for <i>network IDs</i>; that is, the string consists of non-NULL characters, it has a length of at least 1, and there is no maximum length specified. A “-” indicates that none of the names listed apply. The following protocol names are recognized.</p> <p>tcp      Transmission Control Protocol</p> <p>udp      User Datagram Protocol</p> <p>icmp     Internet Control Message Protocol</p>                                                                                                                                                                                                                              |
| <i>network device</i>        | <p>The <i>network device</i> is the full pathname of the device used to connect to the transport provider. Typically, this device will be in the /dev directory. The <i>network device</i> must be specified.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <i>translation libraries</i> | <p>The <i>name-to-address translation libraries</i> support a “directory service” (a name-to-address mapping service) for the network. A “-” in this field indicates the absence of any <i>translation libraries</i>. This has a special meaning for networks of the protocol family <code>inet</code> : its name-to-address mapping is provided by the name service switch based on the entries for <code>hosts</code> and <code>services</code> in <code>nsswitch.conf(4)</code>. For networks of other families, a “-” indicates non-functional name-to-address mapping. Otherwise, this field consists of a comma-separated list of pathnames to dynamically linked libraries. The pathname of the library can be either absolute or relative. See <code>dlopen(3C)</code>.</p> |

Each field corresponds to an element in the `struct netconfig` structure. `struct netconfig` and the identifiers described on this manual page are defined in `<netconfig.h>`. This structure includes the following members:

|                                         |                                                     |
|-----------------------------------------|-----------------------------------------------------|
| <code>char *nc_netid</code>             | Network ID, including NULL terminator.              |
| <code>unsigned long nc_semantics</code> | Semantics.                                          |
| <code>unsigned long nc_flag</code>      | Flags.                                              |
| <code>char *nc_protofmly</code>         | Protocol family.                                    |
| <code>char *nc_proto</code>             | Protocol name.                                      |
| <code>char *nc_device</code>            | Full pathname of the network device.                |
| <code>unsigned long nc_nlookups</code>  | Number of directory lookup libraries.               |
| <code>char **nc_lookups</code>          | Names of the name-to-address translation libraries. |
| <code>unsigned long nc_unused[9]</code> | Reserved for future expansion.                      |

The `nc_semantics` field takes the following values, corresponding to the semantics identified above:

```
NC_TPI_CLTS
NC_TPI_COTS
NC_TPI_COTS_ORD
```

The `nc_flag` field is a bitfield. The following bit, corresponding to the attribute identified above, is currently recognized. NC\_NOFLAG indicates the absence of any attributes.

```
NC_VISIBLE
```

### Examples **EXAMPLE 1** A Sample netconfig File

Below is a sample netconfig file:

```
#
The "Network Configuration" File.
#
Each entry is of the form:
#
<networkid> <semantics> <flags> <protofamily> <protoname> <device>
<nametoaddrlibs>
#
The "-" in <nametoaddrlibs> for inet family transports indicates
redirection to the name service switch policies for "hosts" and
"services". The "-" may be replaced by nametoaddr libraries that
comply with the SVr4 specs, in which case the name service switch
will not be used for netdir_getbyname, netdir_getbyaddr,
gethostbyname, gethostbyaddr, getservbyname, and getservbyport.
There are no nametoaddr_libs for the inet family in Solaris anymore.
#
udp6 tpi_clts v inet6 udp /dev/udp6 -
tcp6 tpi_cots_ord v inet6 tcp /dev/tcp6 -
udp tpi_clts v inet udp /dev/udp -
tcp tpi_cots_ord v inet tcp /dev/tcp -
rawip tpi_raw - inet - /dev/rawip -
ticlts tpi_clts v loopback - /dev/ticlts straddr.so
ticotsord tpi_cots_ord v loopback - /dev/ticotsord straddr.so
ticots tpi_cots v loopback - /dev/ticots straddr.so
```

**Files** <netconfig.h>

**See Also** [dlopen\(3C\)](#), [getnetconfig\(3NSL\)](#), [getnetpath\(3NSL\)](#), [nsswitch.conf\(4\)](#)

*Oracle Solaris Administration: IP Services*

**Name** netgroup – list of network groups

**Synopsis** /etc/netgroup

**Description** A netgroup defines a network-wide group of hosts and users. Use a netgroup to restrict access to shared NFS filesystems and to restrict remote login and shell access.

Network groups are stored in a network information services, such as LDAP or NIS, not in a local file.

This manual page describes the format for a file that is used to supply input to a program such as [ldapaddent\(1M\)](#) for LDAP or [makedbm\(1M\)](#) for NIS. These programs build maps used by their corresponding network information services.

Each line of the file defines the name and membership of a network group. The line should have the format:

```
groupname member...
```

The items on a line can be separated by a combination of one or more spaces or tabs.

The *groupname* is the name of the group being defined. This is followed by a list of members of the group. Each *member* is either another group name, all of whose members are to be included in the group being defined, or a triple of the form:

```
(hostname,username,domainname)
```

In each triple, any of the three fields *hostname*, *username*, and *domainname*, can be empty. An empty field signifies a wildcard that matches any value in that field. Thus:

```
everything (, ,this.domain)
```

defines a group named "everything" for the domain "this.domain" to which every host and user belongs.

The *domainname* field refers to the domain in which the triple is valid, not the domain containing the host or user. In fact, applications using netgroup generally do not check the *domainname*. Therefore, using

```
(, ,domain)
```

is equivalent to

```
(, ,)
```

You can also use netgroups to control NFS mount access (see [share\\_nfs\(1M\)](#)) and to control remote login and shell access (see [hosts.equiv\(4\)](#)). You can also use them to control local login access (see [passwd\(4\)](#), [shadow\(4\)](#), and [compat in nsswitch.conf\(4\)](#)).

When used for these purposes, a host is considered a member of a `netgroup` if the `netgroup` contains any triple in which the `hostname` field matches the name of the host requesting access and the `domainname` field matches the domain of the host controlling access.

Similarly, a user is considered a member of a `netgroup` if the `netgroup` contains any triple in which the `username` field matches the name of the user requesting access and the `domainname` field matches the domain of the host controlling access.

Note that when netgroups are used to control NFS mount access, access is granted depending only on whether the requesting host is a member of the `netgroup`. Remote login and shell access can be controlled both on the basis of host and user membership in separate netgroups.

**Files** `/etc/netgroup` Used by a network information service's utility to construct a map or table that contains `netgroup` information. For example, `ldapaddent(1M)` uses `/etc/netgroup` to construct an LDAP container.

Note that the `netgroup` information must always be stored in a network information service, such as LDAP or NIS. The local file is only used to construct a map or table for the network information service. It is never consulted directly.

**See Also** `ldapaddent(1M)`, `makedbm(1M)`, `share_nfs(1M)`, `inetgr(3C)`, `hosts(4)`, `hosts.equiv(4)`, `nsswitch.conf(4)`, `passwd(4)`, `shadow(4)`

**Notes** `netgroup` requires a network information service such as LDAP or NIS.

Applications may make general membership tests using the `inetgr()` function. See `inetgr(3C)`.

Because the "-" character will not match any specific username or hostname, it is commonly used as a placeholder that will match only wildcarded membership queries. So, for example:

```
onlyhosts (host1,-,our.domain) (host2,-,our.domain)
onlyusers (-,john,our.domain) (-,linda,our.domain)
```

effectively define netgroups containing only hosts and only users, respectively. Any other string that is guaranteed not to be a legal username or hostname will also suffice for this purpose.

Use of placeholders will improve search performance.

When a machine with multiple interfaces and multiple names is defined as a member of a `netgroup`, one must list all of the names. See `hosts(4)`. A manageable way to do this is to define a `netgroup` containing all of the machine names. For example, for a host "gateway" that has names "gateway-subnet1" and "gateway-subnet2" one may define the `netgroup`:

```
gateway (gateway-subnet1,our.domain) (gateway-subnet2,our.domain)
```

and use this `netgroup` "gateway" whenever the host is to be included in another `netgroup`.



**Name** netid – netname database

**Synopsis** /etc/netid

**Description** The `netid` file is a local source of information on mappings between netnames (see [secure\\_rpc\(3NSL\)](#)) and user ids or hostnames in the local domain. The `netid` file can be used in conjunction with, or instead of, the network source, NIS. The `publickey` entry in the `nsswitch.conf` (see [nsswitch.conf\(4\)](#)) file determines which of these sources will be queried by the system to translate netnames to local user ids or hostnames.

Each entry in the `netid` file is a single line of the form:

```
netname uid:gid, gid, gid...
```

or

```
netname 0:hostname
```

The first entry associates a local user id with a netname. The second entry associates a hostname with a netname.

The `netid` file field descriptions are as follows:

*netname* The operating system independent network name for the user or host. *netname* has one of two formats. The format used to specify a host is of the form:

```
unix.hostname@domain
```

where *hostname* is the name of the host and *domain* is the network domain name.

The format used to specify a user id is of the form:

```
unix.uid@domain
```

where *uid* is the numerical id of the user and *domain* is the network domain name.

*uid* The numerical id of the user (see [passwd\(4\)](#)). When specifying a host name, *uid* is always zero.

*group* The numerical id of the group the user belongs to (see [group\(4\)](#)). Several groups, separated by commas, may be listed for a single *uid*.

*hostname* The local hostname (see [hosts\(4\)](#)).

Blank lines are ignored. Any part of a line to the right of a '#' symbol is treated as a comment.

**Examples** EXAMPLE 1 A sample `netid` file.

Here is a sample `netid` file:

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

```
unix.789@West.Sun.COM 789:30,65
unix.123@Bldg_xy.Sun.COM 123:20,1521
unix.candlestick@campus1.bayarea.EDU 0:candlestick
```

**Files** /etc/group groups file  
/etc/hosts hosts database  
/etc/netid netname database  
/etc/passwd password file  
/etc/publickey public key database

**See Also** [netname2user\(3NSL\)](#), [secure\\_rpc\(3NSL\)](#), [group\(4\)](#), [hosts\(4\)](#), [nsswitch.conf\(4\)](#), [passwd\(4\)](#), [publickey\(4\)](#)

**Name** netmasks – network mask database

**Synopsis** /etc/inet/netmasks

/etc/netmasks

**Description** The netmasks file contains network masks used to implement IP subnetting. It supports both standard subnetting as specified in *RFC-950* and variable length subnetting as specified in *RFC-1519*. When using standard subnetting there should be a single line for each network that is subnetted in this file with the network number, any number of SPACE or TAB characters, and the network mask to use on that network. Network numbers and masks may be specified in the conventional IP '.' (dot) notation (like IP host addresses, but with zeroes for the host part). For example,

```
128.32.0.0 255.255.255.0
```

can be used to specify that the Class B network 128.32.0.0 should have eight bits of subnet field and eight bits of host field, in addition to the standard sixteen bits in the network field.

When using variable length subnetting, the format is identical. However, there should be a line for each subnet with the first field being the subnet and the second field being the netmask that applies to that subnet. The users of the database, such as `ifconfig(1M)`, perform a lookup to find the longest possible matching mask. It is possible to combine the *RFC-950* and *RFC-1519* form of subnet masks in the netmasks file. For example,

```
128.32.0.0 255.255.255.0
128.32.27.0 255.255.255.240
128.32.27.16 255.255.255.240
128.32.27.32 255.255.255.240
128.32.27.48 255.255.255.240
128.32.27.64 255.255.255.240
128.32.27.80 255.255.255.240
128.32.27.96 255.255.255.240
128.32.27.112 255.255.255.240
128.32.27.128 255.255.255.240
128.32.27.144 255.255.255.240
128.32.27.160 255.255.255.240
128.32.27.176 255.255.255.240
128.32.27.192 255.255.255.240
128.32.27.208 255.255.255.240
128.32.27.224 255.255.255.240
128.32.27.240 255.255.255.240
128.32.64.0 255.255.255.192
```

can be used to specify different netmasks in different parts of the 128.32.0.0 Class B network number. Addresses 128.32.27.0 through 128.32.27.255 have a subnet mask with 28 bits in the combined network and subnet fields (often referred to as the subnet field) and 4 bits in the host field. Furthermore, addresses 128.32.64.0 through 128.32.64.63 have a 26 bits in the subnet field. Finally, all other addresses in the range 128.32.0.0 through 128.32.255.255 have a 24 bit subnet field.

Invalid entries are ignored.

**See Also** [ifconfig\(1M\)](#), [inet\(7P\)](#)

Postel, Jon, and Mogul, Jeff, *Internet Standard Subnetting Procedure*, RFC 950, Network Information Center, SRI International, Menlo Park, Calif., August 1985.

V. Fuller, T. Li, J. Yu, K. Varadhan, *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*, RFC 1519, Network Information Center, SRI International, Menlo Park, Calif., September 1993.

T. Pummill, B. Manning, *Variable Length Subnet Table For IPv4*, RFC 1878, Network Information Center, SRI International, Menlo Park, Calif., December 1995.

**Notes** `/etc/inet/netmasks` is the official SVr4 name of the netmasks file. The symbolic link `/etc/netmasks` exists for BSD compatibility.

**Name** netrc – file for ftp remote login data

**Description** The `.netrc` file contains data for logging in to a remote host over the network for file transfers by `ftp(1)`. This file resides in the user's home directory on the machine initiating the file transfer. Its permissions should be set to disallow read access by group and others. See `chmod(1)`.

Tokens can be separated by SPACE, TAB, or NEWLINE characters. The following tokens are supported:

|                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>account <i>string</i></code>  | Supply an additional account password. If this token is present, the auto-login process supplies the specified string if the remote server requires an additional account password. If the remote server does not require an additional account password, the auto-login process initiates an ACCT command.                                                                                                                                                  |
| <code>default</code>                | Same as <code>machine <i>name</i></code> , except that <code>default</code> matches any name. There can be only one <code>default</code> token, and it must be after all <code>machine</code> tokens. The <code>default</code> token is normally used as follows:<br><br><code>default login anonymous password <i>user@site</i></code><br><br>Such an entry gives the user automatic anonymous ftp login to machines not specified in <code>.netrc</code> . |
| <code>login <i>name</i></code>      | Identify a user on the remote machine. If this token is present, the auto-login process initiates a login using the specified name.                                                                                                                                                                                                                                                                                                                          |
| <code>machine <i>name</i></code>    | Identify a remote machine name. The auto-login process searches the <code>.netrc</code> file for a <code>machine</code> token that matches the remote machine specified on the <code>ftp</code> command line or as an open command argument. Once a match is made, the subsequent <code>.netrc</code> tokens are processed, stopping when the EOF is reached or another <code>machine</code> token is encountered.                                           |
| <code>macdef <i>name</i></code>     | Define a macro. This token functions the same as <code>ftp macdef</code> . A macro is defined with the specified name; its contents begin with the next <code>.netrc</code> line and continue until a null line (consecutive NEWLINE characters) is encountered. If a macro named <code>init</code> is defined, it is automatically executed as the last step in the auto-login process.                                                                     |
| <code>password <i>string</i></code> | Supply a password. If this token is present, the auto-login process supplies the specified string if the remote server requires a password as part of the login process. If this token is present in the <code>.netrc</code> file, <code>ftp</code> aborts the auto-login process if the <code>.netrc</code> is readable by anyone besides the user.                                                                                                         |
| <code>skipsyst</code>               | Skip the SYST command that is sent by default to all remote servers upon connection. The system command is what enables the automatic use of binary mode rather than the protocol default ascii mode.                                                                                                                                                                                                                                                        |

As some older servers cannot handle the `ftp` command, this directive is provided to allow inter-operability with these servers.

**Examples** EXAMPLE 1 A Sample `.netrc` File

A `.netrc` file containing the following line:

```
machine ray login demo password mypassword
```

allows an autologin to the machine `ray` using the login name `demo` with password `mypassword`.

**Files** `~/.netrc` Contains data for logging into a remote host.

**See Also** [chmod\(1\)](#), [ftp\(1\)](#)

**Name** networks – network name database

**Synopsis** /etc/inet/networks  
/etc/networks

**Description** The networks file is a local source of information regarding the networks which comprise the Internet. The networks file can be used in conjunction with, or instead of, other networks sources, including the NIS maps `networks.byname` and `networks.byaddr` and the NIS+ table `networks`. Programs use the [getnetbyname\(3SOCKET\)](#) routines to access this information.

The network file has a single line for each network, with the following information:

*official-network-name network-number aliases*

Items are separated by any number of SPACE or TAB characters. A '#' indicates the beginning of a comment. Characters up to the end of the line are not interpreted by routines which search the file. This file is normally created from the official network database maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown networks.

Network numbers may be specified in the conventional dot ('.') notation using the `inet_network` routine from the Internet address manipulation library, [inet\(7P\)](#). Network names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

**See Also** [getnetbyaddr\(3SOCKET\)](#), [getnetbyname\(3SOCKET\)](#), [inet\(3SOCKET\)](#), [nsswitch.conf\(4\)](#), [inet\(7P\)](#)

**Notes** The official SVR4 name of the networks file is `/etc/inet/networks`. The symbolic link `/etc/networks` exists for BSD compatibility.

The network number in networks database is the host address shifted to the right by the number of 0 bits in the address mask. For example, for the address `24.132.47.86` that has a mask of `fffffe00`, its network number is `803351`. This is obtained when the address is shifted right by 9 bits. The address maps to `12.66.23`. The trailing 0 bits should not be specified. The network number here is different from that described in [netmasks\(4\)](#). For this example, the entry in `netmasks` would be `24.132.46.0ffffe00`.

**Name** `nfs` – file containing parameter values for NFS-related daemons

**Synopsis** `/etc/default/nfs`

**Description** The settings formerly managed by the `nfs` file have been moved to SMF properties and are now managed by [sharectl\(1M\)](#). The functionality described here is provided for backward compatibility only.

The `nfs` file resides in directory `/etc/default` and provides startup parameters for the [nfsd\(1M\)](#) and [lockd\(1M\)](#) daemons.

The `nfs` file format is ASCII; comment lines begin with the crosshatch (`#`) character. Parameters consist of a keyword followed by an equals (`=`) sign followed by the parameter value, of the form:

*keyword=value*

The following parameters are currently supported in the `nfs` file:

`NFS_CLIENT_VERSMIN=num`

`NFS_CLIENT_VERSMAX=num`

The NFS client only uses NFS versions in the range specified by these variables. Valid values or versions are: 2, 3, and 4. By default these variables are unspecified (commented out) and the client's default minimum is Version 2. The default maximum is Version 4. You can override this range on a per-mount basis by using the `-o vers=` option to [mount\\_nfs\(1M\)](#).

`NFS_SERVER_VERSMIN=num`

`NFS_SERVER_VERSMAX=num`

The NFS server only uses NFS versions in the range specified by these variables. Valid values or versions are: 2, 3, and 4. As with the client, the default is to leave these variables commented out and the default minimum version is 2, while the default maximum version is 4.

`NFS_SERVER_DELEGATION=on | off`

By default, this variable is commented out and the NFS server provides delegations to clients. The user can turn off delegations for all exported filesystems by setting this variable to `off` (case-sensitive). This variable only applies to NFS Version 4.

`NFSMAPID_DOMAIN=domain-string`

By default, the `nfsmapid` uses the DNS domain of the system. This setting overrides the default. This domain is used for identifying user and group attribute strings in the NFS Version 4 protocol. Clients and servers must match with this domain for



---

|                                      |                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                      | operation to proceed normally. This variable only applies to NFS Version 4. See “Setting NFSMAPID_DOMAIN,” below for further details.                                                                                                                                                                                                                                                                                   |
| NFSD_MAX_CONNECTIONS= <i>num</i>     | Sets the maximum number of concurrent, connection-oriented connections. The default is unlimited and is obtained by not setting (that is, commenting out) NFSD_MAX_CONNECTIONS. Equivalent to the -c option in <code>nfsd</code> .                                                                                                                                                                                      |
| NFSD_LISTEN_BACKLOG= <i>num</i>      | Set connection queue length for the NFS over a connection-oriented transport. The default value is 32, meaning 32 entries in the queue. Equivalent to the -l option in <code>nfsd</code> .                                                                                                                                                                                                                              |
| NFSD_PROTOCOL=ALL                    | Start <code>nfsd</code> over the specified protocol only. Equivalent to the -p option in <code>nfsd</code> . ALL is equivalent to -a on the <code>nfsd</code> command line. Mutually exclusive of NFSD_DEVICE. One or the other of NFSD_DEVICE and NFSD_PROTOCOL must be commented out. For the UDP protocol, only version 2 and version 3 service is established. NFS Version 4 is not supported for the UDP protocol. |
| NFSD_DEVICE= <i>devname</i>          | Start NFS daemon for the transport specified by the given device only. Equivalent to the -t option in <code>nfsd</code> . Mutually exclusive of NFSD_PROTOCOL. One or the other of NFSD_DEVICE and NFSD_PROTOCOL must be commented out.                                                                                                                                                                                 |
| NFSD_SERVERS= <i>num</i>             | Maximum number of concurrent NFS requests. Equivalent to last numeric argument on the <code>nfsd</code> command line. The default is 1024.                                                                                                                                                                                                                                                                              |
| LOCKD_LISTEN_BACKLOG= <i>num</i>     | Set connection queue length for <code>lockd</code> over a connection-oriented transport. The default and minimum value is 32.                                                                                                                                                                                                                                                                                           |
| LOCKD_SERVERS= <i>num</i>            | Maximum number of concurrent <code>lockd</code> requests. The default is 1024.                                                                                                                                                                                                                                                                                                                                          |
| LOCKD_RETRANSMIT_TIMEOUT= <i>num</i> | Retransmit timeout, in seconds, before <code>lockd</code> retries. The default is 5.                                                                                                                                                                                                                                                                                                                                    |
| GRACE_PERIOD= <i>num</i>             | Grace period, in seconds, that all clients (both NLM and NFSv4) have to reclaim locks after a server reboot. This parameter also controls the NFSv4 lease                                                                                                                                                                                                                                                               |

interval and overrides the deprecated setting `LOCKD_GRACE_PERIOD`. The default is 90.

`LOCKD_GRACE_PERIOD=num`

Deprecated. Same as `GRACE_PERIOD=num` above. The default is 90.

Setting  
`NFSMAPID_DOMAIN`

As described above, the setting for `NFSMAPID_DOMAIN` overrides the domain used by `nfsmapid(1M)` for building and comparing outbound and inbound attribute strings, respectively. This setting overrides any other mechanism for setting the NFSv4 domain. In the absence of a `NFSMAPID_DOMAIN` setting, the `nfsmapid(1M)` daemon determines the NFSv4 domain as follows:

- If a properly configured `/etc/resolv.conf` (see [resolv.conf\(4\)](#)) exists, `nfsmapid` queries specified nameservers for the domain.
- If a properly configured `/etc/resolv.conf` (see [resolv.conf\(4\)](#)) exists, but the queried nameserver does not have a proper record of the domain name, `nfsmapid` attempts to obtain the domain name through the BIND interface (see [resolver\(3RESOLV\)](#)).
- If no `/etc/resolv.conf` exists, `nfsmapid` falls back on using the configured domain name (see [domainname\(1M\)](#)), which is returned with the leading domain suffix removed. For example, for `widgets.sales.acme.com`, `sales.acme.com` is returned.
- If `/etc/resolv.conf` does not exist, no domain name has been configured (or no `/etc/defaultdomain` exists), `nfsmapid` falls back on obtaining the domain name from the host name, if the host name contains a fully qualified domain name (FQDN).

If a domainname is still not obtained following all of the preceding steps, `nfsmapid` has no domain configured. This results in the following behavior:

- Outbound `owner` and `owner_group` attribute strings are encoded as literal id's. For example, the UID 12345 is encoded as 12345.
- `nfsmapid` ignores the “domain” portion of the inbound attribute string and performs name service lookups only for the user or group. If the user/group exists in the local system name service databases, then the proper uid/gid is mapped even when no domain has been configured.

This behavior implies that the same administrative user/group domain exists between NFSv4 client and server (that is, the same uid/gid's for users/groups on both client and server). In the case of overlapping id spaces, the inbound attribute string could potentially be mapped to the wrong id. However, this is not functionally different from mapping the inbound string to nobody, yet provides greater flexibility.

**See Also** [lockd\(1M\)](#), [mount\\_nfs\(1M\)](#), [nfsd\(1M\)](#), [nfsmapid\(1M\)](#), [sharectl\(1M\)](#)

*Oracle Solaris Administration: Network Services*

**Name** nfslog.conf – NFS server logging configuration file

**Synopsis** /etc/nfs/nfslog.conf

**Description** The `nfslog.conf` file specifies the location of the NFS server logs, as well as the location of the private work files used by the NFS server and `nfslogd(1M)` daemon during logging. Each entry in the file consists of a mandatory tag identifier and one or more parameter identifiers. The parameter identifier specifies the value or location of the specific parameter. For instance, the parameter identifier `"log=/var/nfs/logs/serverLog"` specifies the location of the NFS server activity log. The mandatory tag identifier serves as an index into the `/etc/nfs/nfslog.conf` file to identify the various parameters to be used. At export time, the `share_nfs(1M)` command specifies the NFS server logging parameters to use by associating a tag from the `/etc/nfs/nfslog.conf` file to the exported file system. It is legal for more than one file system to be exported using the same logging tag identifier.

NFS server logging is not supported on Solaris machines that are using NFS Version 4.

A "global" tag identifier is included in `/etc/nfs/nfslog.conf`. It specifies the default set of values to be used during logging. If no tag identifier is specified at export time, then the values in the "global" entry are used. The "global" values can be modified by updating this entry in `/etc/nfs/nfslog.conf`.

Each entry in the file must contain a mandatory tag identifier and at least one parameter/value pair. If a parameter is not specified in a given entry, the global value of the parameter will be used. The exact entry syntax follows:

```
<tag> [defaultdir=<path>] [log=<path><file>] \
[fhtable=<path><file>] [buffer=<path><file>] [logformat=basic|extended]
```

`defaultdir=<path>` Specifies the directory where the logging files and working files will be placed. This path is prepended to all relative paths specified in other parameters.

`log=<path><file>` Specifies the location of the user-readable log file. The log will be located in the `defaultdir`, unless `<path>` is an absolute path.

`fhtable=<path><file>` Specifies the location of the private file handle to path mapping database files. These database files are for the private use of the NFS server kernel module and the `nfslogd` daemon. These files will be located in the `defaultdir`, unless `<path>` is an absolute path. These database files are permanently stored in the file system. Consult `nfslogd(1M)` for information on pruning the database files.

`buffer=<path><file>` Specifies the location of the private work buffer file used by the NFS server kernel module to record raw RPC

information. This file is later processed by the `nfslog` daemon, which in turn generates the user-readable log file. This work buffer file will be located in the `defaultdir`, unless `<path>` is an absolute path.

`logformat=basic|extended` Sets the format of the user-readable log file. If not specified, the basic format is used. The basic format is compatible with log files generated by the Washington University FTPd. The extended format provides a more detailed log, which includes directory modification operations not included in the basic format, such as `mkdir`, `rmdir` and `remove`. Note that the extended format is not compatible with Washington University's FTPd log format.

### Examples EXAMPLE 1 Using the `global` Tag

The "global" tag may be modified so that all exported file systems that enabled logging use a common set of parameters that conform to the specific needs of the user. These values are used until a specific tag identifier overrides them.

```
global defaultdir=/var/nfs log=logs/nfslog \
 fhstable=tables/fhtable buffer=buffers/nfslog_workbuffer \
 logformat=basic
```

### EXAMPLE 2 Overriding the Global `defaultdir` and `logformat`

Because log files can become very large, it may be desirable to store the logs and working files in separate file systems. This can be easily accomplished by simply specifying a different `defaultdir` for every file system exported by means of a unique tag:

```
engineering defaultdir=/engineering/logging \
 logformat=extended
accounting defaultdir=/accounting/logging
marketing defaultdir=/marketing/logging
```

File systems shared with the `engineering` identifier will have their logs and workfiles located in `/engineering/logging`. For instance, the log file will be located at `/engineering/logging/logs/nfslog`. Note that the `engineering` log file will be stored in the extended format, while the rest of the log files will remain in the basic format.

Any of the parameters can be updated in a tag identifier, which overrides the global settings.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE         |
|----------------|-------------------------|
| Availability   | service/file-system/nfs |

**See Also** [nfslogd\(1M\)](#), [share\\_nfs\(1M\)](#), [attributes\(5\)](#)

**Notes** Logs, work files, and file handle to path mapping database can become very large. Be aware of appropriate placement within the file system name space. See [nfslogd\(1M\)](#) for information on pruning the database files and cycling logs.

**Name** nfssec.conf – list NFS security modes

**Synopsis** /etc/nfssec.conf

**Description** The `nfssec.conf` file lists the NFS security modes supported on a system. These modes are defined in [nfssec\(5\)](#).

The `nfssec.conf` file should not be edited by a user. The [kclient\(1M\)](#) program edits `nfssec.conf`.

**See Also** [kclient\(1M\)](#), [nfssec\(5\)](#)

- Name** NISLDAPmapping – mapping file used by the NIS server components
- Synopsis** /var/yp/NISLDAPmapping
- Description** The NISLDAPmapping file specifies the mapping between NIS map entries and equivalent Directory Information Tree (DIT) entries.
- The presence of /var/yp/NISLDAPmapping on a NIS master server causes that server to obtain NIS data from LDAP. See [ypserv\(4\)](#). If /var/yp/NISLDAPmapping is present but the connection configuration file that is defined in /etc/default/ypserv cannot be found, a warning is logged. See [ypserv\(1M\)](#).
- NIS slave servers always obtain their data from a NIS master server, whether or not that server is getting data from LDAP, and ignore the /var/yp/NISLDAPmapping file.
- A simple NISLDAPmapping file is created using [inityp2l\(1M\)](#). You can customize your NISLDAPmapping file as you require.
- Each attribute defined below can be specified in /var/yp/NISLDAPmappingLDAP or as an LDAP attribute. If both are specified, then the attribute in /var/yp/NISLDAPmapping (including empty values) takes precedence.
- A continuation is indicated by a '\' (backslash) in the last position, immediately before the newline of a line. Characters are escaped, that is, exempted from special interpretation, when preceded by a backslash character.
- The '#' (hash) character starts a comment. White space is either ASCII space or a horizontal tab. In general, lines consist of optional white space, an attribute name, at least one white space character, and an attribute value.
- Extended Description**
- File Syntax** Repeated fields, with separator characters, are described by the following syntax:
- One or more entries     entry:entry:entry  
                          entry[":...]
- Zero or more entries  
                          [entry[":...]
- Attributes** Attributes generally apply to one more more NIS maps. Map names can be specified either on their own, that is in passwd.byname, in which case they apply to all domains, or for individual NIS domains, for example, in passwd.byname, example.sun.uk. Where a map is mentioned in more than one attribute, both versions are applied. If any parts of the attributes are in conflict, the domain specific version takes precedence over the non-domain specific version.
- Each domain specific attributes must appear in NISLDAPmapping before any related non-domain specific attribute. If non-domain specific attributes appear first, behavior may be unpredictable. Errors are logged when non-domain specific attributes are found first.

You can associate a group of map names with a `databaseId`. In effect, a macro is expanded to the group of names. Use this mechanism where the same group of names is used in many attributes or where domain specific map names are used. Then, you can make any changes to the domain name in one place.

Unless otherwise noted, all elements of the syntaxes below may be surrounded by white space. Separator characters and white space must be escaped if they are part of syntactic elements.

The following attributes are recognized.

`nisLDAPdomainContext`

The context to use for a NIS domain.

The syntax for `nisLDAPdomainContext` is:

```
NISDomainName ":" context
```

The following is an example of the `nisLDAPdomainContext` attribute:

```
domain.one : dc=site, dc=company, dc=com
```

The mapping file should define the context for each domain before any other attribute makes use of the `NISDomainName` specified for that domain.

`nisLDAPyppasswddDomains`

Lists the domains for which password changes should be made. NIS password change requests do not specify the domains in which any given password should be changed. In traditional NIS this information is effectively hard coded in the NIS makefile.

The syntax for the `nisLDAPyppasswddDomains` attribute is:

```
domainname
```

If there are multiple domains, use multiple `nisLDAPyppasswddDomain` entries withone `domainname` per entry.

`nisLDAPdatabaseIdMapping`

Sets up an alias for a group of NIS map names. There is no default value.

The syntax for the `nisLDAPdatabaseIdMapping` attribute is:

```
databaseId ":" ["["indexlist"]"] mapname[" "...]
```

where



```

databaseId = Label identifying a (subset of a) NIS
 object for mapping purposes.
indexlist = fieldspec["..."]
fieldspec = fieldname "=" fieldvalue
fieldname = The name of a entry field as defined in
 nisLDAPnameFields.
fieldvalue = fieldvaluestring | \" fieldvaluestring \"

```

`indexlist` is used for those cases where it is necessary to select a subset of entries from a NIS map. The subset are those NIS entries that match the `indexlist`. If there are multiple specifications indexed for a particular NIS map, they are tried in the order retrieved until one matches. Note that retrieval order usually is unspecified for multi-valued LDAP attributes. Hence, if using indexed specifications when `nisLDAPdatabaseIdMapping` is retrieved from LDAP, make sure that the subset match is unambiguous.

If the `fieldvaluestring` contains white space or commas, it must either be surrounded by double quotes, or the special characters must be escaped. Wildcards are allowed in the `fieldvaluestring`.

To associate the `passwd.byname` and `passwd.byuid` maps with the `passwd.databaseId`:

```
passwd:passwd.byname passwd.byuid
```

The `passwd` and `passwd.adjunct` `databaseIds` receive special handling. In addition to its normal usage, `passwd` defines which maps `yppasswd` is to update when a `passwd` is changed. In addition to its normal usage `passwd.adjunct` defines which maps `yppasswd` is to update when an adjunct `passwd` is changed.

You may not alias a single map name to a different name, as the results are unpredictable.

```
nisLDAPentryTtl
```

Establish TTLs for NIS entries derived from LDAP.

The syntax for the `nisLDAPentryTtl` attribute is:

```

mapName["..."]:"
 initialTTLlo ":" initialTTLhi ":" runningTTL

```

where

|                           |                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>initialTTLlo</code> | The lower limit for the initial TTL (in seconds) for data read from LDAP when the <code>ypserv</code> starts. If the <code>initialTTLhi</code> also is specified, the actual <code>initialTTL</code> will be randomly selected from the interval <code>initialTTLlo</code> to <code>initialTTLhi</code> , inclusive. Leaving the field empty yields the default value of 1800 seconds. |
| <code>initialTTLhi</code> | The upper limit for the initial TTL. If left empty, defaults to 5400.                                                                                                                                                                                                                                                                                                                  |
| <code>runningTTL</code>   | The TTL (in seconds) for data retrieved from LDAP while the <code>ypserv</code> is running. Leave the field empty to obtain the default value of 3600 seconds.                                                                                                                                                                                                                         |

If there is no specification of TTLs for a particular map, the default values are used.

If the `initialTTLlo` and `initialTTLhi` have the same value, the effect will be that all data known to the `ypserv` at startup times out at the same time. Depending on NIS data lookup patterns, this could cause spikes in `ypserv`-to-LDAP traffic. In order to avoid that, you can specify different `initialTTLlo` and `initialTTLhi` values, and obtain a spread in initial TTLs.

The following is an example of the `nisLDAPentryTTL` attribute used to specify that entries in the NIS host maps read from LDAP should be valid for four hours. When `ypserv` restarts, the disk database entries are valid for between two and three hours.

```
hosts.byname hosts.byaddr:7200:10800:14400
```

`nisLDAPobjectDN`

Specifies the connection between a group of NIS maps and the LDAP directory. This attribute also defines the 'order' of the NIS maps. When NIS maps are bulk copied to or from the DIT, they are

processed in the same order as related nisLDAPobjectDN attributes appear in /var/yp/NISLDAPmapping.

The syntax for the nisLDAPobjectDN attribute is:

```
mapName[" ...] ":" objectDN *(";" objectDN)
```

where

```
objectDN = readObjectSpec [":" [writeObjectSpec]]
readObjectSpec = [baseAndScope [filterAttrValList]]
writeObjectSpec = [baseAndScope [attrValList]]
baseAndScope = [baseDN] ["?" [scope]]
filterAttrValList = ["?" [filter | attrValList]]
scope = "base" | "one" | "sub"
attrValList = attribute "=" value
 *("," attribute "=" value)
```

The baseDN defaults to the value of the nisLDAPdomainContext attribute for the accessed domain. If the baseDN ends in a comma, the nisLDAPdomainContext value is appended.

scope defaults to one. scope has no meaning and is ignored in a writeObjectSpec.

The filter is an LDAP search filter and has no default value.

The attrValList is a list of attribute and value pairs. There is no default value.

As a convenience, if an attrValList is specified in a readObjectSpec, it is converted to a search filter by ANDing together the attributes and the values. For example, the attribute and value list:

```
objectClass=posixAccount,objectClass=shadowAccount
```

is converted to the filter:

```
(&(objectClass=posixAccount)\
 (objectClass=shadowAccount))
```

Map entries are mapped by means of the relevant mapping rules in the nisLDAPnameFields and nisLDAPattributeFromField.

If a `writeObjectSpec` is omitted, the effect is one of the following:

- If there is no trailing colon after the `readObjectSpec`, then there is no write at all.
- If there is a colon after the `readObjectSpec`, then `writeObjectSpec` equals `readObjectSpec`.

The following is an example of a `nisLDAPobjectDN` attribute declaration that gets the `hosts.byaddr` map entries from the `ou=Hosts` container under the default search base and writes to the same place.

```
hosts.byaddr:ou=Hosts,?one?objectClass=ipHost:
```

The following is an example of a `nisLDAPobjectDN` attribute declaration that obtains `passwd` map entries from the `ou=People` containers under the default search base, and also from `dc=another,dc=domain`.

```
passwd:ou=People,?one?\
 objectClass=shadowAccount,\
 objectClass=posixAccount;|\
ou=People,dc=another,dc=domain,?one?\
 objectClass=shadowAccount,\
 objectClass=posixAccount
```

#### `nisLDAPnameFields`

Specifies the content of entries in a NIS map and how they should be broken into named fields.

`nisLDAPnameFields` is required, because NIS maps do not store information in named fields.

The syntax for the `nisLDAPnameFields` attribute is as follows:

```
"nisLDAPnameFields" mapName ":" "(" matchspec "," fieldNames ")"
fieldName = nameOrArrayName["..."]
nameOrArrayName = Name of field or 'array' of repeated fields.
matchspec = "\" formatString \"
```

`formatString` may contain a list of `%s` and `%a` elements each of which represents a single named field or a list of repeated fields. A `%a` field is interpreted as an IPv4 address or an IPv6 address in preferred format. If an IPv6 address in non preferred format is found, then it is converted and a warning is logged.

Where there are a list of repeated fields, the entire list is stored as one entry. The fields are broken up into individual entries, based on the internal separator, at a latter stage. Other characters represent separators which must be present. Any separator, including whitespace, specified by the `formatString`, may be surrounded by a number of whitespace and tab characters. The whitespace and tab characters are ignored.

Regardless of the content of this entry some `fieldNames` are reserved:

|                             |                                                                                                                                |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>rf_key</code>         | The DBM key value                                                                                                              |
| <code>rf_ipkey</code>       | The DBM key value handled as an IP address. See the discussion of <code>%a</code> fields.                                      |
| <code>rf_comment</code>     | Everything following the first occurrence of a symbol. <code>rf_comment</code> is defined by <code>nisLDAPcommentChar</code> . |
| <code>rf_domain</code>      | The name of the domain in which the current NIS operation is being carried out.                                                |
| <code>rf_searchipkey</code> | The <code>rf_searchkey</code> value handled as an IP address. See the discussion of <code>%a</code> fields above.              |
| <code>rf_searchkey</code>   | See the description under <code>nisLDAPattributeFromField</code> below.                                                        |

For example, the `rpc.bynumber` map has the format:

```
name number alias[" "...]
```

The NIS to LDAP system is instructed to break it into a name, a number, and an array of alias field by the following entry in the mapping file:

```
nisLDAPnameFields rpc.bynumber : \
 "%s %s %s", name,number,aliases)
```

**nisLDAPsplitFields**

Defines how a field, or list of fields, named by `nisLDAPnameFields` is split into subfields. The original field is compared with each line of this attribute until one matches. When a match is found named subfields are generated. In latter operations subfield names can be used in the same way as other field names.

The syntax for the `nisLDAPsplitFields` attribute is as follows:

```
"nisLDAPsplitFields" fieldName ":" splitSpec["...]
splitSpec = "(" matchsSpec " " subFieldNames ")"
fieldName = Name of a field from nisLDAPnameFields
subFieldNames = subFieldName["...]
matchSpec = \" formatString \"
```

The netgroup memberTriples can have format (host, user, domain) or groupname. The format is specified by the attribute:

```
nisLDAPsplitField memberTriple: \
 ("(%s,%s,%s)", host, user, domain) , \
 ("%s", group)
```

Later operations can then use field names `host`, `user`, `domain`, `group` or `memberTriple`. Because lines are processed in order, if `host`, `user` and `domain` are found, `group` will not be generated.

Several maps and databaseIds may contain fields that are to be split in the same way. As a consequence, the names of fields to be split must be unique across all maps and databaseIds.

Only one level of splitting is supported. That is, a subfield cannot be split into further subfields.

**nisLDAPrepeatedFieldSeparators**

Where there is a list of repeated, splittable fields, `nisLDAPrepeatedFieldSeparators` specifies which characters separate instances of the splittable field.

The syntax for the `nisLDAPrepeatedFieldSeparators` attribute is as follows:

```
"nisLDAPrepeatedFieldSeparators" fieldName \"sepChar[...]\"
sepChar = A separator character.
```

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nisLDAPcommentChar        | <p>The default value is space or tab. If repeated splittable fields are adjacent, that is, there is no separating character, then the following should be specified:</p> <pre data-bbox="791 267 1300 293">nisLDAPrepeatedFieldSeparators netIdEntry: ""</pre> <p>Specifies which character represents the start of the special comment field in a given NIS map. If this attribute is not present then the default comment character # is used.</p> <p>To specify that a map uses an asterisk to mark the start of comments.</p> <pre data-bbox="791 548 1153 574">nisLDAPcommentChar mapname : '*'</pre> <p>If a map cannot contain comments, then the following attribute should be specified.</p> <pre data-bbox="791 673 1143 699">nisLDAPcommentChar mapname : ''</pre> |
| nisLDAPmapFlags           | <p>Indicates if YP_INTERDOMAIN or YP_SECURE entries should be created in a map. Using nisLDAPmapFlags is equivalent to running <code>makedbm(1M)</code> with the <code>-b</code> or the <code>-s</code> option. When a map is created from the contents of the DIT, the mapping file attribute is the only source for the YP_INTERDOMAIN or YP_SECURE entries.</p> <p>The syntax for the nisLDAPmapFlags attribute is as follows:</p> <pre data-bbox="791 1046 1222 1072">"nisLDAPmapFlags" mapname ":" ["b"]["s"]</pre> <p>By default neither entry is created.</p>                                                                                                                                                                                                          |
| nisLDAPfieldFromAttribute | <p>Specifies how a NIS entries field values are derived from LDAP attribute values.</p> <p>The syntax for the nisLDAPfieldFromAttribute attribute is as follows:</p> <pre data-bbox="791 1308 1300 1334">mapName ":" fieldattrspec *("," fieldattrspec)</pre> <p>The format of fieldattrspec is shown below.</p> <p>To map by direct copy and assignment the value of the ipHostNumber attribute to the addr named field, for example:</p> <pre data-bbox="791 1524 986 1550">addr=ipHostNumber</pre>                                                                                                                                                                                                                                                                         |

Formats for the named field and attribute conversion syntax are discussed below, including examples of complex attribute to field conversions.

#### `nisLDAPAttributeFromField`

Specifies how an LDAP attribute value is derived from a NIS entry field value.

The syntax for the `nisLDAPAttributeFromField` attribute is as follows:

```
mapName ":" fieldattrspec *(", " fieldattrspec)
```

The format of `fieldattrspec` is shown below.

As a special case, if the `dn` attribute value derived from a `fieldattrspec` ends in a comma (“,”), the domains context from `nisLDAPdomainContext` is appended.

Use the following example to map the value of the `addr` field to the `ipHostNumber` attribute by direct copy and assignment:

```
ipHostNumber=addr
```

All relevant attributes, including the `dn`, must be specified.

For every map it must be possible to rapidly find a DIT entry based on its key. There are some maps for which a NIS to LDAP mapping for the key is not desirable, so a key mapping cannot be specified. In these cases a mapping that uses the reserved `rf_searchkey` must be specified. Mappings that use this field name are ignored when information is mapped into the DIT.

#### Field and Attribute Conversion Syntax

The general format of a `fieldattrspec` is:

```
fieldattrspec = lhs "=" rhs
lhs = lval | namespeclist
rhs = rval | [namespec]
namespeclist = namespec | "(" namespec *(", " namespec) ")"
```

The `lval` and `rval` syntax are defined below.. The format of a `namespec` is:

```
namespec
 ["ldap:" attrspec [searchTriple] | ["yp:" fieldname
 [mapspec]
```



---

|                           |                                                        |
|---------------------------|--------------------------------------------------------|
| <code>fieldname</code>    | <code>field   "(" field ")"</code>                     |
| <code>attrspec</code>     | <code>attribute   "(" attribute ")"</code>             |
| <code>searchTriple</code> | <code>":" [baseDN] ["?" [scope] ["?" [filter]]]</code> |
| <code>baseDN</code>       | Base DN for search                                     |
| <code>filter</code>       | LDAP search filter                                     |
| <code>mapspec</code>      | Map name                                               |

The repository specification in a namespec defaults is as follows:

- For assignments to a field:

on the LHS    `yp`  
on the RHS    `ldap`

NIS field values on the RHS are those that exist before the NIS entry is modified.

- For assignments to an attribute:

on the LHS    `ldap`  
on the RHS    `yp`

Attribute values on the RHS are those that exist before the LDAP entry is modified.

When the field or attribute name is enclosed in parenthesis, it denotes a list of field or attribute values. For attributes, the meaning is the list of all attributes of that name, and the interpretation depends on the context. The list specification is ignored when a `searchTriple` or `mapspec` is supplied.

For fields, the `fieldname` syntax is used to map multiple attribute instances to multiple NIS entries.

The `searchTriple` can be used to specify an attribute from a location other than the read or write target. The default values are as follows:

|                     |                                                                                                                                                                                                                  |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>baseDN</code> | If <code>baseDN</code> is omitted, the default is the current <code>objectDN</code> . If the <code>baseDN</code> ends in a comma, the context of the domain is appended from <code>nislDAPdomainContext</code> . |
| <code>scope</code>  | <code>one</code>                                                                                                                                                                                                 |
| <code>filter</code> | Empty                                                                                                                                                                                                            |

Similarly, the `mapspec` can be used to specify a field value from a NIS map other than the one implicitly indicated by the `mapName`. If `searchTriple` or `mapspec` is explicitly specified in a `namespec`, the retrieval or assignment, whether from or to LDAP or NIS, is performed without checking if read and write are enabled for the LDAP container or NIS map.

The omission of the `namespec` in an `rhs` is only allowed if the `lhs` is one or more attributes. The effect is to delete the specified attribute(s). In all other situations, an omitted `namespec` means that the rule is ignored.

The `filter` can be a value. For example, to find the `ipHostNumber` that uses the `cn`, you specify the following in the `filter` field:

```
ldap:ipHostNumber:?one?("cn=%s", (cname, "%s.*"))
```

In order to remove ambiguity, the unmodified value of a single field or attribute must be specified as the following when used in the `filter` field.

```
("%s", namespec)
```

If the `filter` is not specified, the scope will be base, and the `baseDN` is assumed to be the DN of the entry that contains the attribute to be retrieved or modified. To use previously existing field or attribute values in the mapping rules requires a lookup to find those values. Obviously, this adds to the time required to perform the modification. Also, there is a window between the time when a value is retrieved and then slightly later stored back. If the values have changed in the mean time, the change may be overwritten.

When `fieldattrspecs` are grouped into rule sets, in the value of a `nisLDAPfieldFromAttribute` or `nisLDAPattributeFromField` attribute, the evaluation of the `fieldattrspecs` proceed in the listed order. However, evaluation may be done in parallel for multiple `fieldattrspecs`. If there is an error when evaluating a certain `fieldattrspec`, including retrieval or assignment of entry or field values, the extent to which the other `fieldattrspec` rules are evaluated is unspecified.

**Wildcards** Where wildcard support is available, it is of the following limited form:

- \* Matches any number of characters
- [x] Matches the character x
- [x-y] Matches any character in the range x to y, inclusive

Combinations such as [a-cA-C0123] are also allowed, which would match any one of a, b, c, A, B, C, 0, 1, 2, or 3.

```
Substring Extraction substringextract = "(" namespec "," matchspec ")"
name = field or attribute name
matchspec =
```

The `matchspec` is a string like the `sscanf(3C)` format string, except that there may be at most one format specifier, a single `%s`. The output value of the `substringextract` is the substring that matches the location of the `%s`.

If there is no `%s` in the `formatstring`, it must instead be a single character, which is assumed to be a field separator for the `namespec`. The output values are the field values. Wild cards are supported. If there is no match, the output value is the empty string, `""`.

For example, if the `fieldcname` has the value `user.some.domain.name.`, the value of the expression:

```
(cname, "%s.*")
```

is `user`, which can be used to extract the user name from a NIS principal name.

Similarly, use this expression to extract the third of the colon-separated fields of the shadow field:

```
(shadow, "*:*:%s:*")
```

This form can be used to extract all of the shadow fields. However, a simpler way to specify that special case is:

```
(shadow, ":")
```

|        |                           |                                                                                         |
|--------|---------------------------|-----------------------------------------------------------------------------------------|
| Values | <code>lval</code>         | = "(" <code>formatspec</code> "," <code>namespec</code> * "(" <code>namespec</code> ")" |
|        | <code>rval</code>         | = "(" <code>formatspec</code> [" <code>namelist</code> [" <code>elide</code> ] ] ")"    |
|        | <code>namelist</code>     | = <code>name_or_sse</code> * ( " <code>name_or_sse</code> )                             |
|        | <code>name_or_sse</code>  | = <code>namespec</code>   <code>removespec</code>   <code>substringextract</code>       |
|        | <code>removespec</code>   | = <code>list_or_name</code> "-" <code>namespec</code>                                   |
|        | <code>list_or_name</code> | = "(" <code>namespec</code> ")"   <code>namespec</code>                                 |
|        | <code>formatspec</code>   | =                                                                                       |
|        | <code>formatstring</code> | = A string combining text and % field specifications                                    |
|        | <code>elide</code>        | =                                                                                       |
|        | <code>singlechar</code>   | = Any character                                                                         |

The syntax above is used to produce `rval` values that incorporate field or attribute values, in a manner like `sprintf(3C)`, or to perform assignments to `lval` like `sscanf(3C)`. One important restriction is that the format specifications, % plus a single character, use the designations from `ber_printf(3LDAP)`. Thus, while `%s` is used to extract a string value, `%i` causes BER conversion from an integer. Formats other than `%s`, for instance, `%i`, are only meaningfully defined in simple format strings without any other text.

The following `ber_printf()` format characters are recognized:

```
b i n o s
```

If there are too few format specifiers, the format string may be repeated as needed.

When used as an `lval`, there is a combination of pattern matching and assignment, possibly to multiple fields or attributes.

In an assignment to an attribute, if the value of the `addr` field is `1.2.3.4`, the `rval`:

```
("ipNetworkNumber=%s", addr)
```

produces the value `ipNetworkNumber=1.2.3.4`, while:

```
("(%s,%s,%s)", host, user, domain)
```

results in:

```
(assuming host="xyzyz", user="-", domain="x.y.z")
"xyzyz,-,x.y.z"
```

The elide character feature is used with attribute lists. So:

```
("%s", (mgrprfc822mailmember), ",")
```

concatenates all `mgrprfc822mailmember` values into one comma-separated string, and then elides the final trailing comma. Thus, for

```
mgrprfc822mailmember=usera
mgrprfc822mailmember=userb
mgrprfc822mailmember=userc
```

the value would be:

```
usera,userb,userc
```

As a special case, to combine an LHS extraction with an RHS implicit list creates multiple entries and values. So

```
("(%s,%s,%s)", host, user, domain)=(nisNetgroupTriple)
```

creates one NIS entry for each `nisNetgroupTriple` value.

The `'removespec'` form is used to exclude previously assigned fields values from a list. So, if an LDAP entry contains:

```
name: foo
cn: foo
cn: foo1
cn: foo2
```

and the mapping file specifies :

```
myName = name, \
myAliases = ("%s ", (cn) - yp:myName, " ")
```

then the following assignments are carried out:

1. Assign value `foo` to `myName`
2. Assign value `foo foo1 foo2` to `myAliases`
3. Remove value of `myName` from value `myAliases`

This results in the field values `myName` is set to `foo`, and `myAliases` is set to `foo1 foo2`.

Assignments The assignment syntax is as follows:

```
fieldattrspec = lhs "=" rhs
lhs = lval | namespeclist
rhs = rval | namespec
namespeclist = namespec | "(" namespec *(", " namespec) ")"
```

The general form of a simple assignment, which is a one-to-one mapping of field to attribute, is:

```
("%s", fieldname) = ("%s", attrname)
```

As a convenient shorthand, this can also be written as:

```
fieldname=attrname
```

A list specification, which is a name enclosed in parenthesis, can be used to make many-to-many assignments. The expression:

```
(fieldname)=(attrname)
```

where there are multiple instances of `attrname`, creates one NIS entry for each such instance, differentiated by their `fieldname` values. The following combinations of lists are allowed, but they are not particularly useful:

|                                     |                                               |
|-------------------------------------|-----------------------------------------------|
| <code>(attrname)=(fieldname)</code> | Equivalent to <code>attrname=fieldname</code> |
| <code>attrname=(fieldname)</code>   | Equivalent to <code>attrname=fieldname</code> |
| <code>(fieldname)=attrname</code>   | Equivalent to <code>fieldname=attrname</code> |
| <code>fieldname=(attrname)</code>   | Equivalent to <code>fieldname=attrname</code> |

If a multi-valued RHS is assigned to a single-valued LHS, the LHS value will be the first of the RHS values. If the RHS is an attribute list, the first attribute is the first one returned by the LDAP server when queried. Otherwise, the definition of “first” is implementation dependent.

Finally, the LHS can be an explicit list of fields or attributes, such as:

```
(name1, name2, name3)
```

If the RHS is single-valued, this assigns the RHS value to all entities in the list. If the RHS is multi-valued, the first value is assigned to the first entity of the list, the second value to the second entity, and so on. Excess values or entities are silently ignored.

**Examples** EXAMPLE 1 Assigning an Attribute Value to a Field

The following example illustrates how to assign the value of the `ipHostNumber` attribute to the `addr` field

```
addr=ipHostNumber
```

## EXAMPLE 2 Creating Multiple NIS Entries from Multi-Valued LDAP Attributes

An LDAP entry with:

```
cn=name1
cn=name2
cn=name3
```

and the following assignments:

```
cname=cn
(name)=(cn)
```

creates three NIS entries. Other attributes and fields are omitted for clarity.

```
cname=name1, name=name1
cname=name1, name=name2
cname=name1, name=name3
```

## EXAMPLE 3 Assigning String Constants

The following expression sets the `passwd` field to `x`:

```
passwd=("x")
```

## EXAMPLE 4 Splitting Field Values to Multi-Valued Attributes

The `expansion` field contains a comma-separated list of alias member names. In the following example, the expression assigns each member name to an instance of `mgrpRFC822mailmember`:

```
(mgrpRFC822mailmember)=(expansion, ",")
```

**Files** `/var/yp/NISLDAPmapping` Mapping file used by the NIS server components

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE     |
|---------------------|---------------------|
| Availability        | service/network/nis |
| Interface Stability | Obsolete            |

**See Also** [inityp21\(1M\)](#), [makedbm\(1M\)](#), [ypserv\(1M\)](#), [ber\\_printf\(3LDAP\)](#), [sprintf\(3C\)](#), [sscanf\(3C\)](#), [ypserv\(4\)](#), [attributes\(5\)](#)

*Oracle Solaris Administration: Naming and Directory Services*

**Name** nodename – local source for system name

**Synopsis** /etc/nodename

**Description** When a machine is standalone or its IP address is configured locally, the `svc:/system/identity:node` SMF service has a property, namely `nodename`, for setting the system name.

To set the name of the node use the following command:

```
svccfg -s node setprop config/nodename = "mynodename"
svcadm refresh node
```

To see the currently set `nodename` for the host use the following command:

```
$ svcprop -p config/nodename svc:/system/identity:node
```

The value of this property used to be located in the `/etc/nodename` file. If `/etc/nodename` is present and the SMF service property is not set, the SMF service reads `/etc/nodename` and populates the property.

Once the SMF service property has been set, the `/etc/nodename` file is removed and no longer consulted. By convention, the system name is the same as the hostname associated with the IP address of the primary network interface, for example, `hostname.hme0`.

If the machine's network configuration is delivered by the RPC `bootparams` protocol, neither the SMF service property nor the `/etc/nodename` file are used, as the system name is delivered by the remote service.

If the machine's network configuration is delivered by the DHCP protocol, the SMF service property is used only if the DHCP server does not provide a value for the `Hostname` option (DHCP standard option code 12).

A system name configured in the SMF service should be unique within the system's name service domain in order to ensure that any network services provided by the system operate correctly.

Given a system name value, regardless of source, `uname -S` sets the system name of the running system.

**Examples** **EXAMPLE 1** Syntax for the `nodename` File

The syntax for `nodename` consists of a single line containing the system's name. For example, for a system named `myhost`:

```
myhost
```



**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Availability   | system/core-os  |

**See Also** [uname\(1\)](#), [named\(1M\)](#), [ypbind\(1M\)](#), [attributes\(5\)](#)

**Notes** The nodename file is modified by Solaris installation and de-installation scripts.

**Name** nologin – message displayed to users attempting to log on in the process of a system shutdown

**Synopsis** /etc/nologin

**Description** The /etc/nologin file contains the message displayed to users attempting to log on to a machine in the process of being shutdown. After displaying the contents of the nologin file, the login procedure terminates, preventing the user from logging onto the machine.

This procedure is preferable to terminating a user's session by shutdown shortly after the user has logged on.

Logins by super-user are not affected by this procedure.

The message contained in the nologin file is editable by super-user. A typical nologin file contains a message similar to:

NO LOGINS: System going down in 10 minutes.

**See Also** [login\(1\)](#), [rlogin\(1\)](#), [telnet\(1\)](#), [shutdown\(1M\)](#)

**Name** note – specify legal annotations

**Synopsis** /usr/lib/note

**Description** Each file in this directory contains the NOTE (also \_NOTE) annotations legal for a single tool. The name of the file, by convention, should be the tool vendor's stock name, followed by a hyphen, followed by the tool name. For example, for Sun's lock\_lint tool the filename should be SUNW-lock\_lint.

The file should contain the names of the annotations understood by the tool, one per line. For example, if a tool understands the following annotations:

```
NOTE(NOT_REACHED)
NOTE(MUTEX_PROTECTS_DATA(list_lock, list_head))
```

then its file in /usr/lib/note should contain the entries:

```
NOT_REACHED
MUTEX_PROTECTS_DATA
```

Blank lines, and lines beginning with a pound (#), are ignored.

While /usr/lib/note is the default directory tools search for such files, they can be made to search other directories instead simply by setting environment variable NOTEPATH to contain the paths, separated by colons, of directories to be searched, e.g., /usr/mytool/note:/usr/lib/note.

**Usage** These files are used by such tools whenever they encounter NOTEs they do not understand. If a file in /usr/lib/note contains the annotation, then it is valid. If no such file contains the annotation, then the tool should issue a warning complaining that it might be invalid.

**Environment Variables** NOTEPATH specify paths to be searched for annotation files. Paths are separated by colons (".").

**See Also** [NOTE\(3EXT\)](#)

**Name** notrouter – flag to turn off IPv4 routing

**Synopsis** /etc/notrouter

**Description** The /etc/notrouter file is no longer used as of the current release of the Solaris operating system. IPv4 forwarding is disabled by default and can be enabled using [routeadm\(1M\)](#).

**See Also** [routeadm\(1M\)](#)

**Name** nscd.conf – name service cache daemon configuration

**Synopsis** svc:/system/name-service/cache  
/etc/nscd.conf

**Description** The SMF service svc:/system/name-service-cache contains the configuration information for *nscd(1M)*. The nscd.conf file is considered obsolete.

The following nscd properties are supported:

|                                         |                                                                                                                                                                                                                                                                                                                                                         |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| config                                  | Property Group                                                                                                                                                                                                                                                                                                                                          |
| config/enable_per_user_lookup           | Enable per-user lookups                                                                                                                                                                                                                                                                                                                                 |
| config/per_user_nscd_time_to_live       | Per-user Nscd time to live                                                                                                                                                                                                                                                                                                                              |
| config/logfile                          | Log file location                                                                                                                                                                                                                                                                                                                                       |
| config/debug_level                      | Debugging/Log level flags                                                                                                                                                                                                                                                                                                                               |
| config/debug_components                 | subsystem debugging                                                                                                                                                                                                                                                                                                                                     |
| config/global_check_smf_state_interval  | SMF status check interval                                                                                                                                                                                                                                                                                                                               |
| enable_per_user_lookup <i>value</i>     | Enables or disables the ability of nscd to create a per-user nscd. A per-user nscd performs per-user lookups and manages the per-user cache. The per-user lookups might not be possible if the corresponding name service switch backends do not support it or are not configured to do so. The <i>value</i> of this attribute can be true or false.    |
| per_user_nscd_time_to_live <i>value</i> | Sets the time-to-live value for a per-user nscd based on the last time the per-user nscd was active. <i>value</i> is an integer that specifies a number of seconds.                                                                                                                                                                                     |
| logfile                                 | Specifies name of the file to which debug info should be written. Use /dev/tty for standard output.                                                                                                                                                                                                                                                     |
| debug_level <i>value</i>                | Sets the debug level desired. Note the configuration values and output of the debugging command is not likely to remain the same from release-to-release. Use the svccfg describe command to see the available levels. Debugging output formats should not be depended upon in scripts. Use the svccfg describe command to see release specific values. |
| debug_components <i>value</i>           | Sets the nscd components to generate debugs messages. The configuration values and output of the debugging command is not likely to remain the same from release-to-release. Use the svccfg describe command to see the available levels. Debugging output formats should not be depended                                                               |

upon in scripts. Use the `svccfg describe` command to see release specific values.

Additionally the following set of property groups share the same set of properties and manage the individual caches for each of the name service switch databases. The default property group contains any common configuration values for all caches. Additionally the following property groups can contain cache specific configuration:

host, password, group, network, protocol, rpc, ether, netmask,  
bootparam, publickey, netgroup, service, printer, project,  
auth\_attr, prof\_attr, tnrtcp, tnrdhdb

The following properties are common to all the above property groups:

|                                                 |                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>check_files</code> <i>value</i>           | Enables or disables checking the file belonging to the specified cachename for changes. If enabled (which is the default), changes in the corresponding file cause the cache to be invalidated within 10 seconds. Can be disabled if files are never modified for a slight performance boost, particularly over NFS. <i>value</i> can be either <code>true</code> or <code>false</code> .       |
| <code>enable_cache</code> <i>value</i>          | Enables or disables the specified cache. <i>value</i> can be either <code>true</code> or <code>false</code> .                                                                                                                                                                                                                                                                                   |
| <code>keep_hot_count</code> <i>value</i>        | This attribute allows the administrator to set the number of entries <code>nscd(1M)</code> is to keep current in the specified cache. <i>value</i> is an integer number which should approximate the number of entries frequently used during the day.                                                                                                                                          |
| <code>negative_time_to_live</code> <i>value</i> | Sets the time-to-live for negative entries (unsuccessful queries) in the specified cache. <i>value</i> is in integer seconds. Can result in significant performance improvements if there are several files owned by uids (user IDs) not in system databases. Should be kept small to reduce cache coherency problems.                                                                          |
| <code>positive_time_to_live</code> <i>value</i> | Sets the time-to-live for positive entries (successful queries) in the specified cache. <i>value</i> is in integer seconds. Larger values increase cache hit rates and reduce mean response times, but increase problems with cache coherence. Sites that push (update) NIS maps nightly can set <i>value</i> to be the equivalent of 12 hours or more with very good performance implications. |

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

---

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Availability        | system/core-os  |
| Interface Stability | Committed       |

**See Also** [nscd\(1M\)](#), [auth\\_attr\(4\)](#), [bootparams\(4\)](#), [ethers\(4\)](#), [exec\\_attr\(4\)](#), [group\(4\)](#), [hosts\(4\)](#), [netmasks\(4\)](#), [networks\(4\)](#), [passwd\(4\)](#), [prof\\_attr\(4\)](#), [project\(4\)](#), [protocols\(4\)](#), [rpc\(4\)](#), [services\(4\)](#), [user\\_attr\(4\)](#), [attributes\(5\)](#)

**Name** nss – configuration file for initgroups

**Synopsis** /etc/default/nss

**Description** The /etc/default/nss configuration file provides methods for [initgroups\(3C\)](#) lookup method. The file also provides a method to disable address sorting by name lookup functions. The file controls the behavior of the name service switch routines outside of the source database mappings provided by the /etc/nsswitch.conf file.

/etc/default/nss supports the following options:

**NETID\_AUTHORITATIVE** Changes the behavior of the name service lookups to use the `netid` table in response to the `initgroups()` call. By default, `initgroups()` uses the `group` table. When `NETID_AUTHORITATIVE` is set to `TRUE`, `initgroups()` uses `netid` as the source for supplementary groups rather than the `group` table.

The name service administrator must ensure that the `netid` table contains valid supplementary group information for users. Not all name services can automatically keep the members listed in the `group` table in sync with the `netid` table.

**SORT\_ADDR** If this option is set to `FALSE`, the sorting of addresses is disabled on addresses that are returned by name lookup functions such as `initgroups()`, [gethostbyname\(3NSL\)](#), [netdir\\_getbyname\(3NSL\)](#), [getaddrinfo\(3SOCKET\)](#), and [getipnodebyname\(3SOCKET\)](#). Setting this option to `FALSE` is useful when the order of addresses returned by the nameserver needs to be maintained. To use the DNS round robin feature, for example, address sorting by name lookup functions should be disabled.

By default, address sorting is enabled.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Availability        | system/core-os  |
| Interface Stability | Committed       |

**See Also** [getaddrinfo\(3SOCKET\)](#), [gethostbyname\(3NSL\)](#), [getipnodebyname\(3SOCKET\)](#), [initgroups\(3C\)](#), [netdir\\_getbyname\(3NSL\)](#), [nsswitch.conf\(4\)](#), [attributes\(5\)](#)



**Name** nsswitch.conf – configuration file for the name service switch

**Synopsis** svc:/system/name-service/switch  
/etc/nsswitch.conf

**Description** The operating system uses a number of databases of information about hosts, ipnodes, users ([passwd\(4\)](#), [shadow\(4\)](#), and [user\\_attr\(4\)](#)), and groups. Data for these can come from a variety of sources: hostnames and host addresses, for example, can be found in `/etc/hosts`, NIS, LDAP, DNS, or Multicast DNS. Zero or more sources can be used for each database; the sources and their lookup order are specified in `svc:/system/name-service/switch` service. For the purposes of backwards compatibility, the `/etc/nsswitch.conf` file is regenerated from the SMF properties configured in the `svc:/system/name-service/switch` service. The `/etc/nsswitch.conf` file is considered obsolete.

The following databases use the `switch` file:

| Database  | Used By                                                                                                                                                                                |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| alias     | <a href="#">sendmail(1M)</a>                                                                                                                                                           |
| auth_attr | <a href="#">getauthnam(3C)</a>                                                                                                                                                         |
| automount | <a href="#">automount(1M)</a>                                                                                                                                                          |
| bootparam | <a href="#">rpc.bootparamd(1M)</a>                                                                                                                                                     |
| ether     | <a href="#">ethers(3SOCKET)</a>                                                                                                                                                        |
| group     | <a href="#">getgrnam(3C)</a>                                                                                                                                                           |
| host      | <a href="#">gethostbyname(3NSL)</a> , <a href="#">getaddrinfo(3SOCKET)</a> . See <a href="#">Interaction with netconfig</a> .                                                          |
| netgroup  | <a href="#">innetgr(3C)</a>                                                                                                                                                            |
| netmask   | <a href="#">ifconfig(1M)</a>                                                                                                                                                           |
| network   | <a href="#">getnetbyname(3SOCKET)</a>                                                                                                                                                  |
| passwd    | <a href="#">getpwnam(3C)</a> , <a href="#">getspnam(3C)</a> , <a href="#">getusernam(3C)</a>                                                                                           |
| prof_attr | <a href="#">getprofnam(3C)</a> , <a href="#">getexecprof(3C)</a>                                                                                                                       |
| project   | <a href="#">getproject(3PROJECT)</a> , <a href="#">getdefaultproj(3PROJECT)</a> , <a href="#">inproj(3PROJECT)</a> , <a href="#">newtask(1)</a> , <a href="#">setproject(3PROJECT)</a> |
| protocol  | <a href="#">getprotobyname(3SOCKET)</a>                                                                                                                                                |
| publickey | <a href="#">getpublickey(3NSL)</a> , <a href="#">secure_rpc(3NSL)</a>                                                                                                                  |
| rpc       | <a href="#">getrpcbyname(3NSL)</a>                                                                                                                                                     |
| service   | <a href="#">getservbyname(3SOCKET)</a> .                                                                                                                                               |

| Database  | Used By                         |
|-----------|---------------------------------|
|           | See Interaction with netconfig. |
| user_attr | <a href="#">getuserattr(3C)</a> |

The following sources can be used:

| Source | Uses                                                                                                                                                                                                                            |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| files  | /etc/hosts, /etc/passwd, /etc/inet/ipnodes, /etc/shadow, /etc/security/auth_attr, /etc/user_attr                                                                                                                                |
| nis    | NIS(YP)                                                                                                                                                                                                                         |
| ldap   | LDAP                                                                                                                                                                                                                            |
| ad     | Active Directory                                                                                                                                                                                                                |
| dns    | Valid only for hosts and ipnodes. Uses the Internet Domain Name Service.                                                                                                                                                        |
| mdns   | Valid only for hosts and ipnodes. Uses the Multicast Domain Name Service.                                                                                                                                                       |
| compat | The compat source is obsolete. It Implements + and - interaction from SunOS 4. The primary purpose of the compat pseudo database has been replaced by the <a href="#">pam_list(5)</a> module. See <a href="#">pam_list(5)</a> . |

/etc/inet/ipnodes is a symbolic link to /etc/hosts.

The config property group of the svc:/system/name-service/switch service contains the configuration for the nsswitch.conf file.

The config/default property sets the default property for all nsswitch databases, while the other properties can be used to override the default property if desired.

For instance:

```
config/default set to "files", and
config/host set to "files dns" defaults all
databases to local files database access, except for the
host database which will search files first and dns second,
if dns access is configured.
```

The following single-valued properties are supported:

|                 |                          |
|-----------------|--------------------------|
| config/default  | Default db configuration |
| config/host     | Override for host db     |
| config/password | Override for passwd db   |

```

config/group Override for group db
config/network Override for network db
config/protocol Override for protocol db
config/rpc Override for rpc db
config/ether Override for ether db
config/netmask Override for netmask db
config/bootparam Override for bootparam db
config/publickey Override for publickey db
config/netgroup Override for netgroup db
config/automount Override for automount db
config/alias Override for alias db
config/service Override for service db
config/project Override for project db
config/auth_attr Override for auth_attr db
config/prof_attr Override for prof_attr db
config/tnrhtp Override for tnrhtp db
config/tnrhdb Override for tnrhdb db

```

The compat switch source is considered obsolete. It has been replaced with the `pam_list(5)` module. The compat module can still be enabled with the following properties:

```

config/enable_passwd_compat Enable passwd compat
config/enable_group_compat Enable group compat

```

Typically the property values are simple, such as "files" "files nis". However, when multiple sources are specified, it is sometimes necessary to define precisely the circumstances under which each source is tried. A source can return one of the following codes:

| Status   | Meaning                                                      |
|----------|--------------------------------------------------------------|
| SUCCESS  | Requested database entry was found.                          |
| UNAVAIL  | Source is not configured on this system or internal failure. |
| NOTFOUND | Source responded "no such entry"                             |
| TRYAGAIN | Source is busy or not responding, might respond to retries.  |

For each status code, two actions are possible:

| Action   | Meaning                          |
|----------|----------------------------------|
| continue | Try the next source in the list. |
| return   | Return now.                      |

Additionally, for TRYAGAIN only, the following actions are possible:

| Action   | Meaning                                                                                                                                                                                                                                                                                                     |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| forever  | Retry the current source forever.                                                                                                                                                                                                                                                                           |
| <i>n</i> | Retry the current source <i>n</i> more times, where <i>n</i> is an integer between 0 and MAX_INT (that is, 2.14 billion). After <i>n</i> retries has been exhausted, the TRYAGAIN action transitions to continue, until a future request receives a response, at which time TRYAGAIN= <i>n</i> is restored. |

The complete syntax of an entry is:

```
<entry> ::= <database> ":" [<source> [<criteria>]]*
<criteria> ::= "[" <criteria>+ "]"
<criteria> ::= <status> "=" <action>
<status> ::= "success" | "notfound" | "unavail" | "tryagain"
```

For every status except TRYAGAIN, the action syntax is:

```
<action> ::= "return" | "continue"
```

For the TRYAGAIN status, the action syntax is:

```
<action> ::= "return" | "continue" | "forever" | <n>
<n> ::= 0...MAX_INT
```

Each property is a single valued string. The <source> names are case-sensitive, but <action> and <status> names are case-insensitive.

If an database entry or default is absent, the system defaults to "files" in all situations. Additionally if the name service cache service (svc:/system/name-service-cache) is not enabled, all attempts access to remote file services can be ignored.

The default criteria for DNS and the NIS server in "DNS-forwarding mode" is [SUCCESS=return NOTFOUND=continue UNAVAIL=continue TRYAGAIN=3].

The default criteria for all other sources is [SUCCESS=return NOTFOUND=continue UNAVAIL=continue TRYAGAIN=forever].

The default, or explicitly specified, criteria are meaningless following the last source in an entry; and they are ignored, since the action is always to return to the caller irrespective of the status code the source returns.

Interaction with netconfig In order to ensure that they all return consistent results, `gethostbyname(3NSL)`, `getaddrinfo(3SOCKET)`, `getservbyname(3SOCKET)`, and `netdir_getbyname(3NSL)` functions are all implemented in terms of the same internal library function. This function obtains the system-wide source lookup policy for hosts, ipnodes, and services based on the `inet` family entries in `netconfig(4)` and uses the switch entries only if the `netconfig` entries have a - (hyphen) in the last column for `nametoaddr` libraries. See the [Notes](#) section in `gethostbyname(3NSL)` and `getservbyname(3SOCKET)` for details.

Interaction with server in DNS-forwarding Mode The NIS (YP) server can be run in DNS-forwarding mode, where it forwards lookup requests to DNS for host - names and - addresses that do not exist in its database. In this case, specifying `nis` as a source for `hosts` is sufficient to get DNS lookups; `dns` need not be specified explicitly as a source.

Interaction with Password Aging When password aging is turned on, only a limited set of possible name services are permitted for the `passwd` database:

```
passwd: files
passwd: files nis
passwd: files ldap
```

You can add the `ad` keyword to any of the `passwd` configurations listed above. However, you cannot use the `passwd` command to change the password of an Active Directory (AD) user. If the `ad` keyword is found in the `passwd` entry during a password update operation, it is ignored. To update the password of an AD user, use the `kpasswd(1)` command.

Any other settings causes the `passwd(1)` command to fail when it attempts to change the password after expiration and prevents the user from logging in. These are the *only* permitted settings when password aging has been turned on. Otherwise, you can work around incorrect `passwd: lines` by using the `-r repository` argument to the `passwd(1)` command and using `passwd -r repository` to override the `nsswitch.conf` settings and specify in which name service you want to modify your password.

Interaction with +/- syntax The +/- syntax is considered obsolete. It has been replaced both by the `nsswitch` configuration and the `pam_list(5)` module. This syntax will be removed in a future release.

Releases prior to SunOS 5.0 did not have the name service switch but did allow the user some policy control. In `/etc/passwd` one could have entries of the form `+user` (include the specified user from NIS `passwd.byname`), `-user` (exclude the specified user) and `+` (include everything, except excluded users, from NIS `passwd.byname`). The desired behavior was often *everything in the file followed by everything in NIS*, expressed by a solitary `+` at the end of `/etc/passwd`. The switch provides an alternative for this case (`passwd: files nis`) that does not require `+` entries in `/etc/passwd` and `/etc/shadow` (the latter is a new addition to SunOS 5.0, see `shadow(4)`).

If this is not sufficient, the NIS/YP compatibility source provides full +/- semantics. It reads `/etc/passwd` for `getpwnam(3C)` functions and `/etc/shadow` for `getspnam(3C)` functions and, if it finds +/- entries, invokes an appropriate source. By default, the source is `nis`, but this can be overridden by specifying `ldap` as the source for the pseudo-database `passwd_compat`.

In `compat` mode, for every `/etc/passwd` entry, there must be a corresponding entry in the `/etc/shadow` file.

The NIS/YP compatibility source also provides full +/- semantics for group; the relevant pseudo-database is `group_compat`.

**Hard-wired Policies** The compiled-in default entries for all databases is `"files"`.

**Useful Configuration Notes** The files source for the ipnodes and hosts databases is identical, as `/etc/inet/ipnodes` is a symbolic link to `/etc/hosts`. The host property is used for host lookups.

When using Active Directory, `dns` is required to perform hosts resolution.

In order to get information from the Internet Domain Name Service for hosts that are not listed in the enterprise level name service LDAP, use the following configuration and set up the `/etc/resolv.conf` file (see [resolv.conf\(4\)](#) for more details):

```
hosts: files dns
```

**Enumeration - getXXXent()** Many of the databases have enumeration functions: `passwd` has `getpwent()`, `hosts` has `gethostent()`, and so on. These were reasonable when the only source was `files` but often make little sense for hierarchically structured sources that contain large numbers of entries, much less for multiple sources. The interfaces are still provided and the implementations strive to provide reasonable results, but the data returned can be incomplete (enumeration for `hosts` is simply not supported by the `dns` source), inconsistent (if multiple sources are used), formatted in an unexpected fashion (for a host with a canonical name and three aliases, a source might return four hostents, and they might not be consecutive), or very expensive (enumerating a `passwd` database of 5,000 users is probably a bad idea). Furthermore, multiple threads in the same process using the same reentrant enumeration function (`getXXXent_r()` are supported beginning with SunOS 5.3) share the same enumeration position; if they interleave calls, they enumerate disjoint subsets of the same database.

In general, the use of the enumeration functions is deprecated. In the case of `passwd`, `shadow`, and `group`, it might sometimes be appropriate to use `fgetgrent()`, `fgetpwent()`, and `fgetspent()` (see [getgrnam\(3C\)](#), [getpwnam\(3C\)](#), and [getspnam\(3C\)](#), respectively), which use only the `files` source.

**Files** A source named `SSS` is implemented by a shared object named `nss_SSS.so.1` that resides in `/usr/lib`.

`/etc/nsswitch.conf` Configuration file. (Obsolete.)

`/usr/lib/nss_compat.so.1` Implements `compat` source.

`/usr/lib/nss_dns.so.1` Implements `dns` source.

`/usr/lib/nss_files.so.1` Implements `files` source.

`/usr/lib/nss_mdns.so.1` Implements `mdns` source.

`/usr/lib/nss_nis.so.1` Implements `nis` source.

|                                     |                                                                                                               |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------|
| <code>/usr/lib/nss_ldap.so.1</code> | Implements ldap source.                                                                                       |
| <code>/usr/lib/nss_ad.so.1</code>   | Implements ad source.                                                                                         |
| <code>/usr/lib/nss_user.so.1</code> | Implements user source.                                                                                       |
| <code>/etc/netconfig</code>         | Configuration file for <code>netdir(3NSL)</code> functions that redirects hosts/devices policy to the switch. |
| <code>/etc/nsswitch.files</code>    | Sample configuration file that uses files only.                                                               |
| <code>/etc/nsswitch.nis</code>      | Sample configuration file that uses files and nis.                                                            |
| <code>/etc/nsswitch.ldap</code>     | Sample configuration file that uses files and ldap.                                                           |
| <code>/etc/nsswitch.ad</code>       | Sample configuration file that uses files and ad.                                                             |
| <code>/etc/nsswitch.dns</code>      | Sample configuration file that uses files, dns and mdns (dns and mdns only for hosts).                        |

**See Also** `kpasswd(1)`, `newtask(1)`, `passwd(1)`, `automount(1M)`, `ifconfig(1M)`, `mdnsd(1M)`, `rpc.bootparamd(1M)`, `sendmail(1M)`, `getgrnam(3C)`, `getnetgrent(3C)`, `getpwnam(3C)`, `getspnam(3C)`, `gethostbyname(3NSL)`, `getpublickey(3NSL)`, `getrpcbyname(3NSL)`, `netdir(3NSL)`, `secure_rpc(3NSL)`, `getproject(3PROJECT)`, `getdefaultproj(3PROJECT)`, `inproj(3PROJECT)`, `setproject(3PROJECT)`, `getauthnam(3C)`, `getexecprof(3C)`, `getprofnam(3C)`, `getuserattr(3C)`, `getusernam(3C)`, `ethers(3SOCKET)`, `getaddrinfo(3SOCKET)`, `getnetbyname(3SOCKET)`, `getprotobyname(3SOCKET)`, `getservbyname(3SOCKET)`, `auth_attr(4)`, `hosts(4)`, `netconfig(4)`, `project(4)`, `resolv.conf(4)`, `user_attr(4)`, `ypfiles(4)`, `ad(5)`, `ldap(5)`, `pam_list(5)`

**Notes** Within each process that uses `nsswitch.conf`, the entire file is read only once; if the file is later changed, the process continues using the old configuration.

The use of both `nis` and `ldap` as sources for the same database is strongly discouraged since both the name services are expected to store similar information and the lookups on the database can yield different results depending on which name service is operational at the time of the request.

Do not use the `ldap` and `ad` keywords together when the Solaris LDAP client uses schema mapping to talk to Active Directory.

Misspelled names of sources and databases are treated as legitimate names of (most likely nonexistent) sources and databases.

`nsswitch.conf` does not control the name service configuration for everything in Solaris.

The following functions do *not* use the switch: `fgetgrent(3C)`, `fgetproject(3PROJECT)`, `fgetpwent(3C)`, `fgetspent(3C)`, `getpw(3C)`, `putpwent(3C)`, `shadow(4)`.

**Name** packingrules – packing rules file for filesync

**Synopsis** \$HOME/.packingrules

**Description** \$HOME/.packingrules is a packing rules file for filesync. \$HOME/.packingrules contains a list of directories and files that are to be packed and synchronized. It also contains a list of directories and files that are to be specifically excluded from packing and synchronization. See [filesync\(1\)](#).

The \$HOME/.packingrules file is automatically created if users invoke filesync with filename arguments. By using filesync options, users can augment the packing rules in \$HOME/.packingrules.

Many users choose to manually create the packing rules file and edit it by hand. Users can edit \$HOME/.packingrules (using any editor) to permanently change the \$HOME/.packingrules file, or to gain access to more powerful options that are not available from the command line (such as IGNORE commands). It is much easier to enter complex wildcard expressions by editing the \$HOME/.packingrules file.

Blank lines and lines that begin with a pound sign ('#') are ignored.

Any line can be continued by placing a backslash ('\') immediately before the NEWLINE.

All other lines in the \$HOME/.packingrules file have one of the following formats:

- PACKINGRULES *major.minor*. This line is not actually required, but it should be the first line of every packing rules file. This line identifies the packing rules file for the [file\(1\)](#) command and specifies a format version number. The current version number is 1.1. See [file\(1\)](#).
- BASE *directory-1 [directory-2]* This line identifies a directory (or pair of directories) under which files should be packed and synchronized. At least one directory name must be specified. For rules that are to be used by filesync a second directory name (where the copies are to be kept) must also be specified. The arguments must be fully qualified path names, and may include environment variables.
- LIST *name ...* This line enumerates a list of files and sub-directories (beneath the current BASE) that are to be kept synchronized. This specification is recursive, in that specifying the name of a directory automatically includes all files and subdirectories it contains. Regular expressions (as described in [glob](#) and [gmatch](#)) are permitted. See [glob\(1\)](#) and [gmatch\(3GEN\)](#).



`IGNORE name ...` This line enumerates a list of files that are not to be kept synchronized. Regular expressions (using `glob` and `gmatch`) are permitted.

There are important differences between the arguments to `LIST` and `IGNORE` statements. The arguments to a `LIST` statement can contain slashes and are interpreted as file names relative to the `BASE` directories. The arguments to an `IGNORE` statement are simpler names or expressions that cannot contain slashes. An `IGNORE` statement will not override a `LIST` statement. `IGNORE` statements only exclude files that are found beneath `LIST`ed directories.

If the first name argument to a `LIST` statement begins with an exclamation point (`!`), the remainder of the statement will be executed as a command. The command will be run in the current `BASE` directory. The output of the command will be treated as a list of newline separated file names to be packed/synchronized. The resulting file names will be interpreted relative to the enclosing `BASE` directory.

If the first name argument to an `IGNORE` statement begins with an exclamation point (`!`), the remainder of the statement will be executed as a command. The command will be run in the current `BASE` directory. The command will be expected to figure out which names should not be synchronized. The output of the command will be treated as a list of newline separated file names that should be excluded from the packing and synchronization list.

Commands will be broken into distinct arguments and run directly with `sh -c`. Blanks can be embedded in an argument by escaping them with a backslash (`\`) or enclosing the argument in double quotes (`" "`). Double quotes can be passed in arguments by escaping the double quotes with a backslash (`\`).

`LIST` lines only apply to the `BASE` statement that precedes them. `IGNORE` lines can appear before any `BASE` statement (in which case they apply to all `BASE`s) or after a `BASE` statement (in which case they only apply to the `BASE` that precedes them). Any number of these statements can occur in any combination. The order is not important.

**Examples** `EXAMPLE 1` A sample `$HOME.packingrules` file.

The use of these statements is illustrated in the following `$HOME.packingrules` file.

```
#
junk files, not worth copying
#
IGNORE core *.o *.bak *%
#
most of the stuff I want to keep in sync is in my $HOME
#
BASE /net/bigserver/export/home/myname $HOME
everything in my work sub-directory should be maintained
LIST work
a few of my favorite mail boxes should be replicated
```

EXAMPLE 1 A sample \$HOME.packingrules file. (Continued)

```
LIST m/incoming
LIST m/action
LIST m/pending
#
I like to carry around a couple of project directories
but skip all the postscript output
#
BASE /net/bigserver/export/projects $HOME/projects
LIST poindexter epiphany
IGNORE *.ps
#
the foonly package should always be kept on every machine
#
BASE /net/bigserver/opt/foonly /opt/foonly
LIST !cat .packinglist
#
and the latest executables for the standard build environment
#
BASE /net/bigserver/export/buildenv $HOME/buildenv
LIST !find . -type f -a -perm -111 -a -print
```

**See Also** [file\(1\)](#), [filesync\(1\)](#)

**Name** pam.conf – configuration file for pluggable authentication modules

**Synopsis** /etc/pam.conf

**Description** pam.conf is the configuration file for the Pluggable Authentication Module architecture, or PAM. A PAM module provides functionality for one or more of four possible services: authentication, account management, session management, and password management.

|                               |                                                                                                                                                                                        |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| authentication service module | Provides functionality to authenticate a user and set up user credentials.                                                                                                             |
| account management module     | Provides functionality to determine if the current user's account is valid. This includes checking for password and account expiration, as well as verifying access hour restrictions. |
| session management module     | Provides functionality to set up and terminate login sessions.                                                                                                                         |
| password management module    | Provides functionality to change a user's authentication token or password.                                                                                                            |

Each of the four service modules can be implemented as a shared library object which can be referenced in the pam.conf configuration file.

**Simplified pam.conf Configuration File** The pam.conf file contains a listing of services. Each service is paired with a corresponding service module. When a service is requested, its associated module is invoked. Each entry may be a maximum of 256 characters, including the end of line, and has the following format:

*service\_name module\_type control\_flag module\_path options*

The following is an example of a pam.conf configuration file with support for authentication, account management, session management and password management modules (See the pam.conf file that is shipped with your system for the contents of this file):

```
login auth requisite pam_authok_get.so.1
login auth required pam_dhkeys.so.1
login auth required pam_unix_auth.so.1
login auth required pam_dial_auth.so.1

other account requisite pam_roles.so.1
other account required pam_unix_account.so.1

other session required pam_unix_session.so.1

other password required pam_dhkeys.so.1
other password requisite pam_authok_get.so.1
other password requisite pam_authok_check.so.1
other password required pam_authok_store.so.1
```

*service\_name* denotes the service (for example, `login`, `dtlogin`, or `rlogin`).

The keyword, “other,” indicates the module that all other applications which have not been specified should use. The “other” keyword can also be used if all services of the same *module\_type* have the same requirements.

In the example, since all of the services use the same session module, they could have been replaced by a single other line.

*module\_type* denotes the service module type: authentication (`auth`), account management (`account`), session management (`session`), or password management (`password`).

The *control\_flag* field determines the behavior of stacking.

The *module\_path* field specifies the relative pathname to a shared library object, or an included PAM configuration file, which implements the service functionality. If the pathname is not absolute, shared library objects are assumed to be relative to `/usr/lib/security/$ISA/`, and included PAM configuration files are assumed to be relative to `/usr/lib/security/`.

The ISA token is replaced by an implementation defined directory name which defines the path relative to the calling program's instruction set architecture.

The *options* field is used by the PAM framework layer to pass module specific options to the modules. It is up to the module to parse and interpret the options.

This field can be used by the modules to turn on debugging or to pass any module specific parameters such as a TIMEOUT value. The options supported by the modules are documented in their respective manual pages.

#### Integrating Multiple Authentication Services With Stacking

When a *service\_name* of the same *module\_type* is defined more than once, the service is said to be stacked. Each module referenced in the *module\_path* for that service is then processed in the order that it occurs in the configuration file. The *control\_flag* field specifies the continuation and failure semantics of the modules, and can contain one of the following values:

- |                      |                                                                                                                                                                                                                                                                                                    |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>binding</code> | If the service module returns success and no preceding required modules returned failures, immediately return success without calling any subsequent modules. If a failure is returned, treat the failure as a required module failure, and continue to process the PAM stack.                     |
| <code>include</code> | Process the lines from the PAM configuration file that is specified in the <i>module_path</i> at this point in the PAM stack. The “other” keyword is used if the specified <i>service_name</i> is not found. 32 levels of included PAM configuration files are supported. Any options are ignored. |

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>optional</code>   | If the service module returns success, record the success, and continue to process the PAM stack. If a failure is returned, and it is the first <code>optional</code> module failure, save the failure code as an <code>optional</code> failure. Continue to process the PAM stack.                                                                                                                                                |
| <code>required</code>   | If the service module returns success, record the success, and continue to process the PAM stack. If a failure is returned, and it is the first <code>required</code> failure, save the failure code as a <code>required</code> failure. Continue to process the PAM stack.                                                                                                                                                        |
| <code>requisite</code>  | If the service module returns success, record the success, and continue to process the PAM stack. If a failure is returned, immediately return the first non-optional failure value recorded without calling any subsequent modules. That is, return this failure unless a previous <code>required</code> service module failed. If a previous <code>required</code> service module failed, then return the first of those values. |
| <code>sufficient</code> | If the service module return success and no preceding <code>required</code> modules returned failures, immediately return success without calling any subsequent modules. If a failure is returned, treat the failure as an <code>optional</code> module failure, and continue to process the PAM stack.                                                                                                                           |

If the PAM stack runs to completion, that is, neither a `requisite` module failed, nor a `binding` or `sufficient` module success stops it, success is returned if no `required` modules failed and at least one `required`, `requisite`, `optional` module succeeded. If no module succeeded and a `required` or `binding` module failed, the first of those errors is returned. If no `required` or `binding` module failed and an `optional` module failed, the first of the option module errors is returned. If no module in the stack succeeded or failed, that is, all modules returned an `ignore` status, a default error based on module type, for example, “User account expired,” is returned.

All errors in `pam.conf` entries are logged to `syslog` as `LOG_AUTH | LOG_ERR` errors. The use of a service with an error noted in the `pam.conf` entry for that service will fail. The system administrator will need to correct the noted errors before that service may be used. If no services are available or the `pam.conf` file is missing, the system administrator may enter system maintenance mode to correct or restore the file.

The following is a sample configuration file that stacks the `su`, `login`, and `rlogin` services.

```
su auth required pam_inhouse.so.1
su auth requisite pam_authtok_get.so.1
su auth required pam_dhkeys.so.1
su auth required pam_unix_auth.so.1

login auth requisite pam_authtok_get.so.1
login auth required pam_dhkeys.so.1
login auth required pam_unix_auth.so.1
login auth required pam_dial_auth.so.1
```

```
login auth optional pam_inhouse.so.1

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
```

In the case of `su`, the user is authenticated by the `inhouse` and `authtok_get`, `dhkeys`, and `unix_auth` authentication modules. Because the `inhouse` and the other authentication modules are `required` and `requisite`, respectively, an error is returned back to the application if any module fails. In addition, if the `requisite` authentication (`pam_authtok_get` authentication) fails, the other authentication modules are never invoked, and the error is returned immediately back to the application.

In the case of `login`, the `required` keyword for `control_flag` requires that the user be allowed to login only if the user is authenticated by all the service modules. If `pam_unix_auth` authentication fails, control continues to proceed down the stack, and the `inhouse` authentication module is invoked. `inhouse` authentication is optional by virtue of the `optional` keyword in the `control_flag` field. The user can still log in even if `inhouse` authentication fails, assuming the modules stacked above succeeded.

In the case of `rlogin`, the `sufficient` keyword for `control_flag` specifies that if the `rhosts` authentication check succeeds, then PAM should return success to `rlogin` and `rlogin` should not prompt the user for a password. The other authentication modules, which are in the stack, will only be invoked if the `rhosts` check fails. This gives the system administrator the flexibility to determine if `rhosts` alone is sufficient enough to authenticate a remote user.

Some modules return `PAM_IGNORE` in certain situations. In these cases the PAM framework ignores the entire entry in `pam.conf` regardless of whether or not it is `binding`, `requisite`, `required`, `optional`, or `sufficient`.

**Utilities and Files** The specific service names and module types for each service should be documented in the man page for that service. For instance, the `sshd(1M)` man page lists all of the PAM service names and module types for the `sshd` command.

The PAM configuration file does not dictate either the name or the location of the service specific modules. The convention, however, is the following:

|                                         |                                                                                                                                                                         |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pam_module_name.so.x</code>       | File that implements various function of specific authentication services. As the relative pathname specified, <code>/usr/lib/security/\$ISA</code> is prepended to it. |
| <code>/etc/pam.conf</code>              | Configuration file                                                                                                                                                      |
| <code>/usr/lib/\$ISA/libpam.so.1</code> | File that implements the PAM framework library                                                                                                                          |

**Examples** EXAMPLE 1 Using the include control flag

The following example collects the common Unix modules into a single file to be included as needed in the example of a `pam.conf` file. The common Unix module file is named `unix_common` and consists of:

```
OTHER auth requisite pam_authtok_get.so.1
OTHER auth required pam_dhkeys.so.1
OTHER auth required pam_unix_auth.so.1
OTHER auth required pam_unix_cred.so.1
OTHER account requisite pam_roles.so.1
OTHER account required pam_unix_account.so.1
OTHER session required pam_unix_session.so.1
OTHER password required pam_dhkeys.so.1
OTHER password requisite pam_authtok_get.so.1
OTHER password requisite pam_authtok_check.so.1
OTHER password required pam_authtok_store.so.1
```

The `pam.conf` file and consists of:

```
Authentication management
#
login service (explicit because of pam_dial_auth)
#
login auth include unix_common
login auth required pam_dial_auth.so.1
#
rlogin service (explicit because of pam_rhost_auth)
#
rlogin auth sufficient pam_rhosts_auth.so.1
rlogin auth include unix_common
#
Default definitions for Authentication management
Used when service name is not explicitly mentioned
#
OTHER auth include unix_common
#
Default definition for Account management
Used when service name is not explicitly mentioned
#
OTHER account include unix_common
#
Default definition for Session management
Used when service name is not explicitly mentioned
#
OTHER session include unix_common
#
Default definition for Password management
Used when service name is not explicitly mentioned
```

EXAMPLE 1 Using the include control flag (Continued)

```
#
OTHER password include unix_common
```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | See Below.      |

The format is Committed. The contents has no stability attributes.

**See Also** [login\(1\)](#), [passwd\(1\)](#), [in.rlogind\(1M\)](#), [in.rshd\(1M\)](#), [in.telnetd\(1M\)](#), [in.uucpd\(1M\)](#), [init\(1M\)](#), [rpc.rexd\(1M\)](#), [ttymon\(1M\)](#), [su\(1M\)](#), [pam\(3PAM\)](#), [syslog\(3C\)](#), [libpam\(3LIB\)](#), [attributes\(5\)](#), [environ\(5\)](#), [pam\\_authtok\\_check\(5\)](#), [pam\\_authtok\\_get\(5\)](#), [pam\\_authtok\\_store\(5\)](#), [pam\\_dhkeys\(5\)](#), [pam\\_krb5\(5\)](#), [pam\\_passwd\\_auth\(5\)](#), [pam\\_unix\\_account\(5\)](#), [pam\\_unix\\_auth\(5\)](#), [pam\\_unix\\_session\(5\)](#)

**Notes** The `pam_unix` 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\(5\)](#).

With the removal of the `pam_unix` module, the SunOS delivered PAM service modules no longer need or support the “`use_first_pass`” or “`try_first_pass`” options. This functionality is provided by stacking [pam\\_authtok\\_get\(5\)](#) above a module that requires a password.



**Name** passwd – password file

**Synopsis** /etc/passwd

**Description** The file `/etc/passwd` is a local source of information about users' accounts. The password file can be used in conjunction with other naming sources, such as the NIS maps `passwd.byname` or password data stored on an LDAP server. Programs use the `getpwnam(3C)` routines to access this information.

Each `passwd` entry is a single line of the form:

```
username:password:uid:
gid:gcos-field:home-dir:
login-shell
```

where

*username* is the user's login name.

The `login` (`login`) and `role` (`role`) fields accept 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 displayed if these restrictions are not met.

The `login` and `role` fields must contain at least one character and must not contain a colon (`:`) or a newline (`\n`).

*password* is an empty field. The encrypted password for the user is in the corresponding entry in the `/etc/shadow` file. `pwconv(1M)` relies on a special value of `'x'` in the password field of `/etc/passwd`. If this value of `'x'` exists in the password field of `/etc/passwd`, this indicates that the password for the user is already in `/etc/shadow` and should not be modified.

*uid* is the user's unique numerical ID for the system.

*gid* is the unique numerical ID of the group that the user belongs to.

*gcos-field* is the user's real name, along with information to pass along in a mail-message heading. (It is called the `gcos-field` for historical reasons.) An `"&"` (ampersand) in this field stands for the login name (in cases where the login name appears in a user's real name).

*home-dir* is the pathname to the directory in which the user is initially positioned upon logging in.

*login-shell* is the user's initial shell program. If this field is empty, the default shell is `/usr/bin/sh`.

The maximum value of the *uid* and *gid* fields is 2147483647. To maximize interoperability and compatibility, administrators are recommended to assign users a range of UIDs and GIDs below 60000 where possible. (UIDs from 0-99 inclusive are reserved by the operating system vendor for use in future applications. Their use by end system users or vendors of layered products is not supported and may cause security related issues with future applications.)

The password file is an ASCII file that resides in the `/etc` directory. Because the encrypted passwords on a secure system are always kept in the shadow file, `/etc/passwd` has general read permission on all systems and can be used by routines that map between numerical user IDs and user names.

Blank lines are treated as malformed entries in the `passwd` file and cause consumers of the file, such as `getpwnam(3C)`, to fail.

The password file can contain entries beginning with a '+' (plus sign) or '-' (minus sign) to selectively incorporate entries from another naming service source, such as NIS or LDAP.

A line beginning with a '+' means to incorporate entries from the naming service source. There are three styles of the '+' entries in this file. A single + means to insert all the entries from the alternate naming service source at that point, while a `+name` means to insert the specific entry, if one exists, from the naming service source. A `+%netgroup` means to insert the entries for all members of the network group `netgroup` from the alternate naming service. If a `+name` entry has a non-null password, `gcos`, `home-dir`, or `login-shell` field, the value of that field overrides what is contained in the alternate naming service. The *uid* and *gid* fields cannot be overridden.

A line beginning with a '-' means to disallow entries from the alternate naming service. There are two styles of '-' entries in this file. `-name` means to disallow any subsequent entries (if any) for `name` (in this file or in a naming service), and `-%netgroup` means to disallow any subsequent entries for all members of the network group `netgroup`.

This is also supported by specifying "passwd : compat" in `nsswitch.conf(4)`. The "compat" source might not be supported in future releases. The preferred sources are `files` followed by the identifier of a name service, such as `nis` or `ldap`. This has the effect of incorporating the entire contents of the naming service's `passwd` database or password-related information after the `passwd` file.

Note that in `compat` mode, for every `/etc/passwd` entry, there must be a corresponding entry in the `/etc/shadow` file.

Appropriate precautions must be taken to lock the `/etc/passwd` file against simultaneous changes if it is to be edited with a text editor; `vipw(1B)` does the necessary locking.

#### Examples EXAMPLE 1 Sample passwd File

The following is a sample `passwd` file:

**EXAMPLE 1** Sample passwd File (Continued)

```
root:x:0:1:Super-User:/:/usr/sbin/sh
fred:6k/7KCFRPNVXg:508:10:& Fredericks:/usr2/fred:/bin/csh
```

and the sample password entry from `nsswitch.conf`:

```
passwd: files ldap
```

In this example, there are specific entries for users `root` and `fred` to assure that they can login even when the system is running `single-user`. In addition, anyone whose password information is stored on an LDAP server will be able to login with their usual password, shell, and home directory.

If the password file is:

```
root:x:0:1:Super-User:/:/usr/sbin/sh
fred:6k/7KCFRPNVXg:508:10:& Fredericks:/usr2/fred:/bin/csh
+
```

and the password entry in `nsswitch.conf` is:

```
passwd: compat
```

then all the entries listed in the NIS `passwd.byuid` and `passwd.byname` maps will be effectively incorporated after the entries for `root` and `fred`. If the password entry in `nsswitch.conf` is:

```
passwd_compat: ldap
passwd: compat
```

then all password-related entries stored on the LDAP server will be incorporated after the entries for `root` and `fred`.

The following is a sample `passwd` file when `shadow` does not exist:

```
root:q,mJzTnu8icf.:0:1:Super-User:/:/usr/sbin/sh
fred:6k/7KCFRPNVXg:508:10:& Fredericks:/usr2/fred:/bin/csh
+john:
+@documentation:no-login:
+:::Guest
```

The following is a sample `passwd` file when `shadow` does exist:

```
root:##root:0:1:Super-User:/:/usr/sbin/sh
fred:##fred:508:10:& Fredericks:/usr2/fred:/bin/csh
+john:
+@documentation:no-login:
+:::Guest
```

**EXAMPLE 1** Sample passwd File (Continued)

In this example, there are specific entries for users `root` and `fred`, to assure that they can log in even when the system is running standalone. The user `john` will have his password entry in the naming service source incorporated without change, anyone in the `netgroup` documentation will have their password field disabled, and anyone else will be able to log in with their usual password, shell, and home directory, but with a `gcos` field of `Guest`.

- Files**
- `/etc/nsswitch.conf`
  - `/etc/passwd`
  - `/etc/shadow`

**See Also** `chgrp(1)`, `chown(1)`, `finger(1)`, `groups(1)`, `login(1)`, `newgrp(1)`, `passwd(1)`, `sh(1)`, `sort(1)`, `domainname(1M)`, `getent(1M)`, `pwck(1M)`, `pwconv(1M)`, `su(1M)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `a64l(3C)`, `crypt(3C)`, `getpw(3C)`, `getpwnam(3C)`, `getspnam(3C)`, `putpwent(3C)`, `group(4)`, `hosts.equiv(4)`, `nsswitch.conf(4)`, `shadow(4)`, `environ(5)`, `unistd.h(3HEAD)`

*Oracle Solaris Administration: Common Tasks*

**Notes** The `solaris.user.manage` and `solaris.role.manage` authorizations are required to modify the `passwd` fields for users and roles respectively. These authorizations allow an administrator to set the `username`, `uid`, `gcos-field`, `home-dir`, and `login-shell` for users and roles respectively. Setting the `gid` requires the `solaris.group.delegate/assign` authorization. See `group(4)`.

**Name** path\_to\_inst – device instance number file

**Synopsis** /etc/path\_to\_inst

**Description** /etc/path\_to\_inst records mappings of physical device names to instance numbers.

The instance number of a device is encoded in its minor number, and is the way that a device driver determines which of the possible devices that it may drive is referred to by a given special file.

In order to keep instance numbers persistent across reboots, the system records them in /etc/path\_to\_inst.

This file is read only at boot time, and is updated by [add\\_drv\(1M\)](#) and [devfsadm\(1M\)](#).

Note that it is generally not necessary for the system administrator to change this file, as the system will maintain it.

The system administrator can change the assignment of instance numbers by editing this file and doing a reconfiguration reboot. However, any changes made in this file will be lost if [add\\_drv\(1M\)](#) or [devfsadm\(1M\)](#) is run before the system is rebooted.

Each instance entry is a single line of the form:

```
"physical name" instance number "driver binding name"
```

where

*physical name* is the absolute physical pathname of a device. This pathname must be enclosed in double quotes.

*instance number* is a decimal or hexadecimal number.

*driver binding name* is the name used to determine the driver for the device. This name may be a driver alias or a driver name. The driver binding name must be enclosed in double quotes.

**Examples** EXAMPLE 1 Sample path\_to\_inst Entries

Here are some sample path\_to\_inst entries:

```
"/iommu@f,e0000000" 0 "iommu"
"/iommu@f,e0000000/sbus@f,e0001000" 0 "sbus"
"/iommu@f,e0000000/sbus@f,e0001000/sbusmem@e,0" 14 "sbusmem"
"/iommu@f,e0000000/sbus@f,e0001000/sbusmem@f,0" 15 "sbusmem"
"/iommu@f,e0000000/sbus@f,e0001000/ledma@f,400010" 0 "ledma"
"/obio/serial@0,100000" 0 "zs"
"/SUNW,sx@f,80000000" 0 "SUNW,sx"
```

**Files** /etc/path\_to\_inst Mapping of physical device names to instance numbers.

**See Also** [add\\_drv\(1M\)](#), [boot\(1M\)](#), [devfsadm\(1M\)](#), [mknod\(1M\)](#)

**Warnings** If the file is removed the system may not be bootable (as it may rely on information found in this file to find the root, usr or swap device). If it does successfully boot, it will regenerate the file, but after rebooting devices may end up having different minor numbers than they did before, and special files created via [mknod\(1M\)](#) may refer to different devices than expected.

For the same reasons, changes should not be made to this file without careful consideration.

**Notes** This document does not constitute an API. `path_to_inst` may not exist or may have a different content or interpretation in a future release. The existence of this notice does not imply that any other documentation that lacks this notice constitutes an API.

**Name** pci, pcie – configuration files for PCI and PCI Express device drivers

**Description** The Peripheral Component Interconnect (PCI) bus is a little endian bus. PCI Express (PCIe) and PCI-X are successors to PCI. All three types of devices share the same configuration parameters. What is specified here for PCI devices applies to PCI-X 1.0 devices as well. All three types of devices are self-identifying, which means that these devices provide configuration parameters to the system that allow the system to identify the device and its driver. The configuration parameters are represented in the form of name-value pairs that can be retrieved using the DDI property interfaces. See [ddi\\_prop\\_lookup\(9F\)](#) for details.

The bus properties of PCI devices or logical bus properties of PCIe devices are derived from PCI configuration space, or supplied by the Fcode PROM, if it exists. Therefore, driver configuration files are not necessary for these devices.

On some occasions, drivers for PCI and PCIe devices can use driver configuration files to provide driver private properties through the global property mechanism. See [driver.conf\(4\)](#) for further details. Driver configuration files can also be used to augment or override properties for a specific instance of a driver.

All bus drivers of PCI and PCIe devices recognize the following properties:

**reg** An arbitrary length array where each element of the array consists of a 5-tuple of 32-bit values. Each array element describes a logically contiguous mappable resource on the PCI bus or PCIe device tree.

The first three values in the 5-tuple describe the PCI address of the mappable resource. The first tuple contains the following information:

---

|              |                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------|
| Bits 0 - 7   | 8-bit register number                                                                                    |
| Bits 8 - 10  | 3-bit function number                                                                                    |
| Bits 11 - 15 | 5-bit device number                                                                                      |
| Bits 16 - 23 | 8-bit bus number                                                                                         |
| Bits 24 - 25 | 2-bit address space type identifier                                                                      |
| Bits 31 - 28 | Register number extended bits 8:11 for extended config space. Zero for conventional configuration space. |

---

The address space type identifier can be interpreted as follows:

|     |                                |
|-----|--------------------------------|
| 0x0 | configuration space            |
| 0x1 | I/O space                      |
| 0x2 | 32-bit memory space<br>address |
| 0x3 | 64-bit memory space<br>address |

The bus number is a unique identifying number assigned to each PCI bus or PCIe logical bus within its domain.

The device number is a unique identifying number assigned to each device on a PCI bus or PCIe logical bus. Note that a device number is unique only within the set of device numbers for a particular bus or logical bus.

Each PCI or PCIe device can have one to eight logically independent functions, each with its own independent set of configuration registers. Each function on a device is assigned a function number. For a device with only one function, the function number must be 0.

The register number fields select a particular register within the set of configuration registers corresponding to the selected function. When the address space type identifier indicates configuration space, non-zero register number extended bits select registers in extended configuration space.

The second and third values in the `reg` property 5-tuple specify the 64-bit address of the mappable resource within the PCI or PCIe address domain. The second 32-bit tuple corresponds to the high order four bytes of the 64-bit address. The third 32-bit tuple corresponds to the low order bytes.

The fourth and fifth 32-bit values in the 5-tuple `reg` property specify the size of the mappable resource. The size is a 64-bit value, where the fourth tuple corresponds to the high order bytes of the 64-bit size and the fifth corresponds to the low order.

The driver can refer to the elements of this array by index, and construct kernel mappings to these addresses using `ddi_regs_map_setup(9F)`. The index into the array is passed as the *number* argument of `ddi_regs_map_setup(9F)`.

At a high-level interrupt context, you can use the `ddi_get*` and `ddi_put*` family of functions to access I/O and memory space. However, access to configuration space is not allowed when running at a high-interrupt level.

`interrupts` This property consists of a single-integer element array. Valid interrupt property values are 1, 2, 3, and 4. This value is derived directly from the contents of the device's configuration-interrupt-pin register.



A driver should use an index value of 0 when registering its interrupt handler with the DDI interrupt interfaces.

All PCI and PCIe devices support the `reg` property. The device number and function number as derived from the `reg` property are used to construct the address part of the device name under `/devices`.

Only devices that generate interrupts support an `interrupts` property.

Occasionally it might be necessary to override or augment the configuration information supplied by a PCI or PCIe device. This change can be achieved by writing a driver configuration file that describes a prototype device node specification containing the additional properties required.

For the system to merge the prototype node specification into an actual device node, certain conditions must be met.

- First, the name property must be identical. The value of the name property needs to match the binding name of the device. The binding name is the name chosen by the system to bind a driver to a device and is either an alias associated with the driver or the hardware node name of the device.
- Second, the parent property must identify the PCI bus or PCIe logical bus.
- Third, the unit-address property must identify the card. The format of the unit-address property is:

`DD[ , F]`

where DD is the device number and F is the function number. If the function number is 0, only DD is specified.

#### Examples EXAMPLE 1 Sample Configuration File

An example configuration file called ACME, `scsi-hba.conf` for a PCI driver called ACME, `scsi-hba` follows:

```
#
Copyright (c) 1995, ACME SCSI Host Bus Adaptor
ident "@(#)ACME,scsi-hba.conf 1.1 96/02/04"
name="ACME,scsi-hba" parent="/pci@1,0/pci@1f,4000"
 unit-address="3" scsi-initiator-id=6;
hba-advanced-mode="on";
hba-dma-speed=10;
```

In this example, a property `scsi-initiator-id` specifies the SCSI bus initiator id that the adapter should use, for just one particular instance of adapter installed in the machine. The name property identifies the driver and the parent property to identify the particular bus the

**EXAMPLE 1** Sample Configuration File *(Continued)*

card is plugged into. This example uses the parent's full path name to identify the bus. The unit-address property identifies the card itself, with device number of 3 and function number of 0.

Two global driver properties are also created: `hba-advanced-mode` (which has the string value `on`) and `hba-dma-speed` (which has the value `10 M bit/s`). These properties apply to all device nodes of the `ACME, scsi-hba`.

Configuration files for PCIe devices are similar. Shown below is an example configuration file called `ACME,pcie-widget.conf` for a PCIe driver called `ACME,pcie-widget`.

```
#
Copyright (c) 2005, ACME PCIe Widget Adapter
ident "@(#)ACME,pcie-widget.conf 1.1 05/11/14"
name="ACME,pcie-widget" parent="/pci@780" unit-address="2,1"
debug-mode=12;
```

In this example, we provide a property `debug-mode` for a particular PCIe device. As before, the logical bus is identified by the pathname of the parent of the device. The device has a device number of 2, and a function number of 1.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Architecture   | SPARC, x86      |

**See Also** [driver.conf\(4\)](#), [attributes\(5\)](#), [ddi\\_intr\\_add\\_handler\(9F\)](#), [ddi\\_prop\\_lookup\(9F\)](#), [ddi\\_regs\\_map\\_setup\(9F\)](#)

*Writing Device Drivers*

*IEEE 1275 PCI Bus Binding*

<http://playground.sun.com/1275/bindings/pci/pci-express.txt>

**Notes** PCIe devices support an extended configuration space unavailable to PCI devices. While PCIe devices can be operated using a PCI device driver, operating them using a PCIe device driver can make use of the extended properties and features made available only in the extended configuration space.

---

**Name** phones – remote host phone number database

**Synopsis** /etc/phones

**Description** The file /etc/phones contains the system-wide private phone numbers for the [tip\(1\)](#) program. /etc/phones is normally unreadable, and so may contain privileged information. The format of /etc/phones is a series of lines of the form:

```
<system-name>[\t]*<phone-number>.
```

The system name is one of those defined in the [remote\(4\)](#) file and the phone number is constructed from [0123456789-=%]. The '=' and '\*' characters are indicators to the auto call units to pause and wait for a second dial tone (when going through an exchange). The '=' is required by the DF02-AC and the '\*' is required by the BIZCOMP 1030.

Comment lines are lines containing a '#' sign in the first column of the line.

Only one phone number per line is permitted. However, if more than one line in the file contains the same system name [tip\(1\)](#) will attempt to dial each one in turn, until it establishes a connection.

**Files** /etc/phones

**See Also** [tip\(1\)](#), [remote\(4\)](#)

**Name** pkginfo – package characteristics file

**Description** pkginfo is an ASCII file that describes the characteristics of the package along with information that helps control the flow of installation. It is created by the software package developer.

Each entry in the pkginfo file is a line that establishes the value of a parameter in the following form:

```
PARAM="value"
```

There is no required order in which the parameters must be specified within the file. The PKG, NAME, ARCH, VERSION and CATEGORY parameters are mandatory. Other parameters are optional.

pkginfo provides optional parameters and an environment variable in support of the zones (multiple Solaris environments) feature. See [zones\(5\)](#).

The following parameters are mandatory:

#### ARCH

A comma-separated list of alphanumeric tokens that indicate the architecture associated with the package. The [pkgmk\(1\)](#) tool can be used to create or modify this value when actually building the package. The maximum length of a token is 16 characters and it cannot include a comma.

Solaris's installation software meaningfully uses only one architecture token of the form:

```
<instruction_set_architecture>[.<platform_group>]
```

where *platform\_group* is intended only for Solaris installation packages. Third party application software should restrict itself to ARCH values from the following Solaris-supported instruction set architectures (uname -p): sparc, i386, and ppc. Examples of Solaris' platform groups (uname -m) are sun4u for the SPARC instruction set and i86pc for the i386 instruction set. See [uname\(1\)](#) and [isalist\(1\)](#) for more details.

#### CATEGORY

A comma-separated list of categories under which a package can be displayed. A package must at least belong to the system or application category. Categories are case-insensitive and can contain only alphanumerics. Each category is limited in length to 16 characters.

#### NAME

Text that specifies the package name (maximum length of 256 ASCII characters). Use the NAME parameter as the foundation for describing the functionality and purpose of the package; spell out any acronyms and avoid internal product/project code names. The DESC parameter can then be used to expand the descriptive information. Use the NAME parameter to state as specifically as possible the use of the package, why a user would need to load it, and so on.

**PKG**

Abbreviation for the package being installed. All characters in the abbreviation must be alphanumeric. You can also use the – and + characters in the abbreviation. The first character cannot be numeric, a + or a -.

The abbreviation is limited to a maximum length of 32 characters. `install`, `new`, and `all` are reserved abbreviations. It is customary to make the first four letters unique to your company, such as the company's stock symbol.

**VERSION**

Text that specifies the current version associated with the software package. The maximum length is 256 ASCII characters and the first character cannot be a left parenthesis. The `pkgmk(1)` tool can be used to create or modify this value when actually building the package. Current Solaris software practice is to assign this parameter monotonically increasing Dewey decimal values of the form:

```
<major_revision>.<minor_revision>[.<micro_revision>]
```

where all the revision fields are integers. The versioning fields can be extended to an arbitrary string of numbers in Dewey-decimal format, if necessary.

The following parameters are optional:

**BASEDIR**

The pathname to a default directory where “relocatable” files can be installed. If blank, the package is not relocatable and any files that have relative pathnames are not installed. An administrator can override the default directory.

**CLASSES**

A space-separated list of classes defined for a package. The order of the list determines the order in which the classes are installed. Classes listed first are installed first (on a media by media basis). This parameter can be modified by the request script.

**DESC**

Text that describes the package (maximum length of 256 ASCII characters). This parameter value is used to provide the installer with a description of what the package contains and should build on the description provided in the `NAME` parameter. Try to make the two parameters work together so that a `pkginfo -l` provides a fairly comprehensive textual description of the package.

**EMAIL**

An electronic address where further information is available or bugs can be reported (maximum length of 256 ASCII characters).

**HOTLINE**

Phone number and/or mailing address where further information can be received or bugs can be reported (maximum length of 256 ASCII characters).

**INTONLY**

Indicates that the package should only be installed interactively when set to any non-null value.

**ISTATES**

A list of allowable run states for package installation (for example, “S s 1” allows run states of S, s or 1). The Solaris operating environment supports the run levels s, S, 0, 1, 2, 3, 5, and 6. Applicable run levels for this parameter are s, S, 1, 2, and 3. See [init\(1M\)](#) for details.

**MAXINST**

The maximum number of package instances that should be allowed on a machine at the same time. By default, only one instance of a package is allowed. This parameter must be set in order to have multiple instances of a package. In order to support multiple instances of packages (for example, packages that differ in their ARCH or VERSION parameter value), the value of this parameter must be high enough to allow for all instances of a given package, including multiple versions coexisting on a software server.

**ORDER**

A list of classes defining the order in which they should be put on the medium. Used by [pkgmk\(1\)](#) in creating the package. Classes not defined in this field are placed on the medium using the standard ordering procedures.

**PSTAMP**

Production stamp used to mark the [pkgmap\(4\)](#) file on the output volumes. Provides a means for distinguishing between production copies of a version if more than one is in use at a time. If PSTAMP is not defined, the default is used. The default consists of the UNIX system machine name followed by the string “YYYYMMDDHHMMSS” (year, month, date, hour, minutes, seconds).

**RSTATES**

A list of allowable run states for package removal (for example, “S s 1” allows run states of S, s or 1). The Solaris operating environment supports the run levels s, S, 0, 1, 2, 3, 5, and 6. Applicable run levels for this parameter are s, S, 1, 2, and 3. See [init\(1M\)](#) for details.

**SUNW\_ISA**

Solaris-only optional parameter that indicates a software package contains 64-bit objects if it is set to sparcv9. If this parameter is not set, the default ISA (instruction set architecture) is set to the value of the ARCH parameter.

**SUNW\_LOC**

Solaris-only optional parameter used to indicate a software package containing localization files for a given product or application. The parameter value is a comma-separated list of locales supported by a package. It is only used for packages containing localization files, typically the message catalogues. The allowable values for this string field are those found in the table of Standard Locale Names located in the [International Language Environments Guide](#).

```
SUNW_LOC="<locale_name>,<locale_name>, . . . \
<locale_name>"
```

where

```
<locale_name> ::= <language>[_<territory>]\
[.<codeset>]
<language> ::= the set of names from ISO 639
<territory> ::= the set of territories specified
in ISO 3166
<codeset> ::= is a string corresponding to the coded
character set
```

Since a value of `C` specifies the traditional UNIX system behavior (American English, `en_US`), packages belonging to the `C` locale are viewed as non-localized packages, and thus must not have `SUNW_LOC` and `SUNW_PKGLIST` included in their `pkginfo` file. See also the `SUNW_LOC` parameter in [setlocale\(3C\)](#) for more information. This keyword is not recognized by the add-on software utility Software Manager.

#### SUNW\_PKG\_DIR

A value set by `pkgadd` that contains the location of the installing package. This value is provided to any install time package procedure scripts that need to know where the installing package is located. This parameter should never be set manually from within a `pkginfo` file.

#### SUNW\_PKG\_ALLZONES

Defines whether a package, when installed, must be installed and must be identical in all zones. Assigned value can be `true` or `false`. The default value is `false`. The setting of `SUNW_PKG_ALLZONES` has the effects described below.

If set to `true`, the following conditions are in effect:

- The package must be installed in the global zone.
- The package must be installed in any non-global zone that is created.
- The package must be identical in all zones.
- The package can be installed only by the global zone administrator.
- The package cannot be installed by a non-global zone administrator.

If set to `false`, the following conditions are in effect:

- The package is not required to be installed in all zones.
- The package is not required to be identical across all zones.
- The package can be installed by the global zone administrator or by a non-global zone administrator.

Packages that must be identical across all zones must set this variable to `true`. This would include packages that deliver components that are part of the core operating system, or that are dependent on interfaces exported by the core operating system, or that deliver device drivers, or runtime libraries that use or export operating system interfaces that are not guaranteed to be stable across minor releases.

Packages that deliver components that are not part of the core operating system (such as application programs) that can be different between any two zones must set this variable to `false`.

With respect to `SUNW_PKG_ALLZONES`, keep in mind the following:

- Use of `pkgadd` in the global zone installs packages in all zones unless `-G` is specified, in which case packages are installed in the global zone only. The setting of `SUNW_PKG_ALLZONES` does not change this behavior. For example, a package that has a setting of `SUNW_PKG_ALLZONES=false` is not installed in the global zone only.
- The `SUNW_PKG_ALLZONES` attribute controls whether a package *must* be installed in all zones (and must be the same in all zones) when it is installed.
- Use of the `-G` option to `pkgadd` with a package that has `SUNW_PKG_ALLZONES=true` is an error and causes installation of that package to fail.

#### `SUNW_PKG_HOLLOW`

Defines whether a package should be visible in any non-global zone if that package is required to be installed and be identical in all zones (for example, a package that has `SUNW_PKG_ALLZONES=true`). Assigned value can be `true` or `false`. The default value is `false`. The package is not required to be installed, but if it is installed, the setting of `SUNW_PKG_HOLLOW` has the effects described below.

If set to `false`, the following conditions are in effect:

- If installed in the global zone, the package content and installation information are required in all non-global zones.
- Software delivered by the package is visible in all non-global zones. An example of such a package is the package that delivers the `truss(1)` command.

If set to `true`, the following conditions are in effect:

- The package content is not delivered on any non-global zone. However, the package installation information is required on all non-global zones.
- The package delivers software that should not be visible in all non-global zones. Examples include kernel drivers and system configuration files that work only in the global zone. This setting allows the non-global zone to resolve dependencies on packages that are installed only in the global zone without actually installing the package data.
- In the global zone, the package is recognized as having been installed, and all components of the package are installed. Directories are created, files are installed, and class action and other scripts are run as appropriate when the package is installed.
- In a non-global zone, the package is recognized as having been installed, but no components of the package are installed. No directories are created, no files are installed, and no class action or other install scripts are run when the package is installed.



- When removed from the global zone, the package is recognized as having been completely installed. Appropriate directories and files are removed, and class action or other install scripts are run when the package is removed.
- When removed from a non-global zone, the package is recognized as not having been completely installed. No directories are removed, no files are removed, and no class action or other install scripts are run when the package is removed.
- The package is recognized as being installed in all zones for purposes of dependency checking by other packages that rely on this package being installed.

If `SUNW_PKG_ALLZONES` is set to `false`, the value of this variable has no meaning. It is a package construction error to set `SUNW_PKG_ALLZONES` to `false`, then set `SUNW_PKG_HOLLOW` to `true`.

#### `SUNW_PKG_THISZONE`

Defines whether a package must be installed in the current zone only. Assigned value can be `true` or `false`. The default value is `false`. The setting of `SUNW_PKG_THISZONE` has the effects described below.

If set to `true`, the following conditions are in effect:

- The package is installed in the current zone only.
- If installed in the global zone, the package is not added to any currently existing or yet-to-be-created non-global zones. This is the same behavior that would occur if the `-G` option were specified to `pkgadd`.

If set to `false`, the following conditions are in effect:

- If `pkgadd` is run in a non-global zone, the package is installed in the current zone only.
- If `pkgadd` is run in the global zone, the package is installed in the global zone, and is also installed in all currently installed non-global zones. In addition, the package will be propagated to all future, newly installed non-global zones.

#### `SUNW_PKGLIST`

Solaris-only optional parameter used to associate a localization package to the package(s) from which it is derived. It is required whenever the `SUNW_LOC` parameter is defined. This parameter value is an comma-separated list of package abbreviations of the form:

```
SUNW_PKGLIST="pkg1[:version] , pkg2[:version] , . . ."
```

where *version* (if specified) should match the version string in the base package specified (see `VERSION` parameter in this manual page). When in use, `SUNW_PKGLIST` helps determine the order of package installation. The packages listed in the parameter are installed before the localization package in question is installed. When left blank, `SUNW_PKGLIST=""`, the package is assumed to be required for the locale to function correctly. This keyword is not recognized by the add-on software utility Software Manager.

**SUNW\_PKGTYPE**

Solaris-only parameter for Sun internal use only. Required for packages part of the Solaris operating environment releases which install into the `/`, `/usr`, `/usr/kvm`, and `/usr/openwin` file systems. The Solaris operating environment installation software must know which packages are part of which file system to properly install a server/client configuration. The currently allowable values for this parameter are `root`, `usr`, `kvm`, and `ow`. If no `SUNW_PKGTYPE` parameter is present, the package is assumed to be of `BASEDIR=/opt`. `SUNW_PKGTYPE` is optional only for packages which install into the `/opt` name space as is the case for the majority of Solaris add-on software.

**SUNW\_PKGVERS**

Solaris-only parameter indicating of version of the Solaris operating environment package interface.

```
SUNW_PKGVERS="<sunw_package_version>"
```

where *<unw\_package\_version>* has the form *x.y[.z]* and *x*, *y*, and *z* are integers. For packages built for this release and previous releases, use `SUNW_PKGVERS="1.0"`.

**SUNW\_PRODNAME**

Solaris-only parameter indicating the name of the product this package is a part of or comprises (maximum length of 256 ASCII characters). A few examples of currently used `SUNW_PRODNAME` values are: "SunOS", "OpenWindows", and "Common Desktop Environment".

**SUNW\_PRODVERS**

Solaris-only parameter indicating the version or release of the product described in `SUNW_PRODNAME` (maximum length of 256 ASCII characters). For example, where `SUNW_PRODNAME="SunOS"`, and the Solaris 2.x Beta release, this string could be "5.x BETA", while for the Solaris 2.x FCS release, the string would be "5.x". For Solaris 10, the string is "5.10". If the `SUNW_PRODNAME` parameter is NULL, so should be the `SUNW_PRODVERS` parameter.

**ULIMIT**

If set, this parameter is passed as an argument to the `ulimit(1)` command (see [limit\(1\)](#)), which establishes the maximum size of a file during installation.

**VENDOR**

Used to identify the vendor that holds the software copyright (maximum length of 256 ASCII characters).

**VSTOCK**

The vendor stock number, if any, that identifies this product (maximum length of 256 ASCII characters).

For further discussion of the zones-related parameters described above, see [Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management](#).

**Environment Variables** The following environment variables are available only to package class action scripts and to checkinstall, preinstall, postinstall scripts.

**SUNW\_PKG\_INSTALL\_ZONE**

This variable is set only during the initial installation of a zone.

If this variable is not set, the system does not support the [zones\(5\)](#) feature. In this case, the package is being installed to or removed from a system that is not configured for zones.

If the variable is set, and equal to `global`, the package is being installed to or removed from the global zone.

If the variable is set and not equal to `global`, the package is being installed to or removed from the non-global zone named by the contents of the environment variable `${SUNW_PKG_INSTALL_ZONE}`.

**PKG\_INIT\_INSTALL**

This variable is set only during an initial installation of Solaris, such as installing Solaris from a CD, DVD, or net install image.

If this variable is set and equal to `TRUE`, then the package is being installed as part of an initial installation of Solaris.

If this variable is not set, or set and not equal to `TRUE`, then the package is not being installed as part of an initial installation of Solaris.

The following code excerpt illustrates the semantics of the preceding environment variables.

```
if [$PKG_INIT_INSTALL != ""] ; then
 # Package being installed as part of initial
 # installation of Solaris.

elif [$SUNW_PKG_INSTALL_ZONE != ""] ; then

 if [$SUNW_PKG_INSTALL_ZONE != "global"] ; then
 # Package being installed as part of initial installation
 # of non-global zone $SUNW_PKG_INSTALL_ZONE
 else
 # Package being installed as part of initial installation
 # of a global zone.
 fi

else
 # Package not being installed as part of initial installation of
 # Solaris and package not being installed as part of initial
 # installation of non-global zone.
fi
```

**Examples** EXAMPLE 1 A Sample pkginfo File

Here is a sample pkginfo file:

```
SUNW_PRODNAME="SunOS"
SUNW_PRODVERS="5.5"
SUNW_PKGTYPE="usr"
SUNW_PKG_ALLZONES=false
SUNW_PKG_HOLLOW=false
PKG="SUNWesu"
NAME="Extended System Utilities"
VERSION="11.5.1"
ARCH="sparc"
VENDOR="Sun Microsystems, Inc."
HOTLINE="Please contact your local service provider"
EMAIL=""
VSTOCK="0122c3f5566"
CATEGORY="system"
ISTATES="S 2"
RSTATES="S 2"
```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE   |
|---------------------|-------------------|
| Availability        | system/core-os    |
| Interface Stability | See entries below |
| PKG value           | Committed         |
| VERSION value       | Committed         |
| NAME value          | Committed         |
| DESC value          | Committed         |
| ARCH value          | Committed         |
| CATEGORY value      | Committed         |
| BASEDIR value       | Committed         |
| ISTATES value       | Committed         |
| RSTATES value       | Committed         |
| MAXINST value       | Committed         |
| SUNW_PKG_ALLZONES   | Committed         |
| SUNW_PKG_HOLLOW     | Committed         |

| ATTRIBUTE TYPE    | ATTRIBUTE VALUE |
|-------------------|-----------------|
| SUNW_PKG_THISZONE | Committed       |
| SUNW_PRODNAME     | Committed       |
| SUNW_PRODVERS     | Committed       |
| SUNW_PKGVERS      | Committed       |
| SUNW_PKGTYPE      | Uncommitted     |
| SUNW_LOC          | Committed       |
| SUNW_PKGLIST      | Committed       |
| SUNW_PKG_DIR      | Committed       |
| PKG_INIT_INSTALL  | Uncommitted     |

**See Also** [isalist\(1\)](#), [limit\(1\)](#), [pkgmk\(1\)](#), [uname\(1\)](#), [init\(1M\)](#), [setlocale\(3C\)](#), [pkgmap\(4\)](#), [attributes\(5\)](#), [zones\(5\)](#)

*Application Packaging Developer's Guide*

*International Language Environments Guide*

*Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*

**Notes** Developers can define their own installation parameters by adding a definition to this file. A developer-defined parameter must begin with a capital letter.

Trailing white space after any parameter value is ignored. For example, `VENDOR="Sun Microsystems, Inc."` is the same as `VENDOR="Sun Microsystems, Inc. "`.

**Name** pkgmap – package contents description file

**Description** pkgmap is an ASCII file that provides a complete listing of the package contents. It is automatically generated by [pkgmk\(1\)](#) using the information in the [prototype\(4\)](#) file.

Each entry in pkgmap describes a single “deliverable object file.” A deliverable object file includes shell scripts, executable objects, data files, directories, and so forth. The entry consists of several fields of information, each field separated by a space. The fields are described below and must appear in the order shown.

*part* An optional field designating the part number in which the object resides. A part is a collection of files and is the atomic unit by which a package is processed. A developer can choose the criteria for grouping files into a part (for example, based on class). If no value is defined in this field, part 1 is assumed.

*ftype* A one-character field that indicates the file type. Valid values are listed below. File types are divided between those that are not to be modified and those that are modifiable.

Files of the following types must never be modified:

- b block special device
- c character special device
- d directory
- f a standard executable file, data file, or other type of file, the contents of which must never be modified.
- i information file (such as a file containing a copyright, list of dependencies, or package information) or installation script (such as checkinstall, class action [i. ], pre/post install/remove), the contents of which must never be modified.
- l linked file
- p named pipe
- s symbolic link
- x an exclusive directory accessible only by this package

Files of the following types can be modified:

- e An editable file, intended to be edited (selectively modified) after installation. An editable file is expected to change on installation or removal, can be shared by several packages, and must be installed by a class action script. Examples are a configuration file or a list of users.

- v* A volatile file, intended to be overwritten or appended to after installation. A volatile file is not expected to change on installation or removal, is not preserved between installations, and can be installed by a class action script. Examples are a log file or a lock file.

Following package installation, the contents of files of all types except *e* and *v* must not change. Any file that is subject to change should be marked as *e* or *v*.

*class* The installation class to which the file belongs. This name must contain only alphanumeric characters and be no longer than 12 characters. It is not specified if the *f*type is *i* (information file).

*pathname* *pathname* may contain variables of the form *\$variable* that support install-time configuration of the file. *variable* may be embedded in the pathname structure. (See [prototype\(4\)](#) for definitions of variable specifications.)

Do not use the following reserved words in *pathname*, since they are applied by [pkgadd\(1M\)](#) using a different mechanism:

```
PKG_INSTALL_ROOT
BASEDIR
CLIENT_BASEDIR
```

*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 files, packaging information files, or non-installable files.

The mode can contain a variable specification. (See [prototype\(4\)](#) for definitions of variable specifications.)

*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 or changed to the owner stored in the package database, which could be different from what is on the file system. When the question mark is used, it implies that the file is already on the file system. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what owner an installation script will be executed.

The owner can contain a variable specification. (See [prototype\(4\)](#) for definitions of variable specifications.)

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>group</i>   | 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 or changed to the owner stored in the package database, which could be different from what is on the file system. When the question mark is used, it implies that the file is already on the file system. This field is not used for linked files or non-installable files. It is used optionally with a package information file. If used, it indicates with what group an installation script will be executed.<br><br>The group can contain a variable specification. (See <a href="#">prototype(4)</a> for definitions of variable specifications.) |
| <i>size</i>    | The actual size of the file in bytes. This field is not specified for named pipes, special devices, directories or linked files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <i>cksum</i>   | The checksum of the file contents. This field is not specified for named pipes, special devices, directories, or linked files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <i>modtime</i> | The time of last modification, as reported by the <a href="#">stat(2)</a> function call. This field is not specified for named pipes, special devices, directories, or linked files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

Each pkgmap file must have one line that provides information about the number of parts, maximum size of parts that make up the package, and, optionally, the size of the package after compression (where size is given in 512-byte blocks). This line is in the following format:

```
: number_of_parts maximum_part_size compressed_pkg_size
```

Lines that begin with “#” are comment lines and are ignored.

When files are saved during installation before they are overwritten, they are normally just copied to a temporary pathname. However, for files whose mode includes execute permission (but which are not editable), the existing version is linked to a temporary pathname and the original file is removed. This allows processes which are executing during installation to be overwritten.

#### Examples EXAMPLE 1 A Sample pkgmap File

```
: 2 500
1 i pkginfo 237 1179 541296672
1 d none bin 0755 root bin
1 f none bin/INSTALL 0755 root bin 11103 17954 541295535
1 f none bin/REMOVE 0755 root bin 3214 50237 541295541
1 l none bin/UNINSTALL=bin/REMOVE
1 f none bin/cmda 0755 root bin 3580 60325 541295567
1 f none bin/cmdb 0755 root bin 49107 51255 541438368
1 f class1 bin/cmdc 0755 root bin 45599 26048 541295599
1 f class1 bin/cmdd 0755 root bin 4648 8473 541461238
1 f none bin/cmde 0755 root bin 40501 1264 541295622
```



---

**EXAMPLE 1** A Sample pkgmap File *(Continued)*

```
1 f class2 bin/cmdf 0755 root bin 2345 35889 541295574
1 f none bin/cmdg 0755 root bin 41185 47653 541461242
2 d class2 data 0755 root bin
2 p class1 data/apipe 0755 root other
2 d none log 0755 root bin
2 v none log/logfile 0755 root bin 41815 47563 541461333
2 d none save 0755 root bin
2 d none spool 0755 root bin
2 d none tmp 0755 root bin
```

**See Also** [pkgmk\(1\)](#), [pkgadd\(1M\)](#), [stat\(2\)](#), [pkginfo\(4\)](#), [prototype\(4\)](#)

*Application Packaging Developer's Guide*

**Notes** The pkgmap file may contain only one entry per unique pathname.

**Name** plot – graphics interface

**Description** Files of this format are interpreted for various devices by commands. A graphics file is a stream of plotting instructions. Each instruction consists of an ASCII letter usually followed by bytes of binary information. The instructions are executed in order. A point is designated by four bytes representing the  $x$  and  $y$  values; each value is a signed integer. The last designated point in an  $l$ ,  $m$ ,  $n$ , or  $p$  instruction becomes the “current point” for the next instruction.

- $m$  Move: the next four bytes give a new current point.
- $n$  Cont: draw a line from the current point to the point given by the next four bytes.
- $p$  Point: plot the point given by the next four bytes.
- $l$  Line: draw a line from the point given by the next four bytes to the point given by the following four bytes.
- $t$  Label: place the following ASCII string so that its first character falls on the current point. The string is terminated by a NEWLINE.
- $a$  Arc: the first four bytes give the center, the next four give the starting point, and the last four give the end point of a circular arc. The least significant coordinate of the end point is used only to determine the quadrant. The arc is drawn counter-clockwise.
- $c$  Circle: the first four bytes give the center of the circle, the next two the radius.
- $e$  Erase: start another frame of output.
- $f$  Linemod: take the following string, up to a NEWLINE, as the style for drawing further lines. The styles are “dotted,” “solid,” “longdashed,” “shortdashed,” and “dotdashed.” Effective only in `plot 4014` and `plot ver`.
- $s$  Space: the next four bytes give the lower left corner of the plotting area; the following four give the upper right corner. The plot will be magnified or reduced to fit the device as closely as possible.

Space settings that exactly fill the plotting area with unity scaling appear below. The upper limit is just outside the plotting area.

In every case the plotting area is taken to be square; points outside may be displayable on devices whose face is not square.

```
4014 space(0, 0, 3120, 3120);
ver space(0, 0, 2048, 2048);
300,300s space(0, 0, 4096, 4096);
450 space(0, 0, 4096, 4096);
```

---

|                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>                 | policy.conf – configuration file for security policy                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Synopsis</b>             | /etc/security/policy.conf                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Description</b>          | The <code>policy.conf</code> file provides the security policy configuration for user-level attributes. Each entry consists of a key/value pair in the form:<br><br>key=value<br><br>The following keys are defined:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| AUTHS_GRANTED               | Specify the default set of authorizations granted to all users. This entry is interpreted by <a href="#">chkauthattr(3C)</a> . The value is zero or more comma-separated authorizations defined in <a href="#">auth_attr(4)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| PROFS_GRANTED               | Specify the default set of profiles granted to all users. This entry is interpreted by <a href="#">chkauthattr(3C)</a> and <a href="#">getexecuser(3C)</a> . The value is zero or more comma-separated profiles defined in <a href="#">prof_attr(4)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| CONSOLE_USER                | Specify an additional default set of profiles granted to the <i>console user</i> user. This entry is interpreted by <a href="#">chkauthattr(3C)</a> and <a href="#">getexecuser(3C)</a> . The value is zero or more comma-separated profiles defined in <a href="#">prof_attr(4)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| PRIV_DEFAULT and PRIV_LIMIT | Settings for these keys determine the default privileges that users have. (See <a href="#">privileges(5)</a> .) If these keys are not set, the default privileges are taken from the inherited set. PRIV_DEFAULT determines the default set on login. PRIV_LIMIT defines the limit set on login. Users can have privileges assigned or taken away through use of <a href="#">user_attr(4)</a> . Privileges can also be assigned to profiles, in which case users who have those profiles can exercise the assigned privileges through <a href="#">pfexec(1)</a> .<br><br>For maximum future compatibility, the privilege specifications should always include <code>basic</code> or <code>all</code> . Privileges should then be removed using negation. See EXAMPLES. By assigning privileges in this way, you avoid a situation where, following an addition of a currently unprivileged operation to the basic privilege set, a user unexpectedly does not have the privileges he needs to perform that now-privileged operation. |

Note that removing privileges from the limit set requires *extreme* care, as any set-uid root program might suddenly fail because it lacks certain privilege(s). Note also that dropping `basic` privileges from the default privilege set can cause unexpected failure modes in applications.

|                                          |                                                                                                                                                                                                                                                                                                                                              |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>LOCK_AFTER_RETRIES=YES NO</code>   | Specifies whether a local account is locked after the count of failed logins for a user equals or exceeds the allowed number of retries as defined by <code>RETRIES</code> in <code>/etc/default/login</code> . The default value for users is <code>NO</code> . Individual account overrides are provided by <a href="#">user_attr(4)</a> . |
| <code>CRYPT_ALGORITHMS_ALLOW</code>      | Specify the algorithms that are allowed for new passwords and is enforced only in <a href="#">crypt_gensalt(3C)</a> .                                                                                                                                                                                                                        |
| <code>CRYPT_ALGORITHMS_DEPRECATED</code> | Specify the algorithm for new passwords that is to be deprecated. For example, to deprecate use of the traditional UNIX algorithm, specify <code>CRYPT_ALGORITHMS_DEPRECATED=__unix__</code> and change <code>CRYPT_DEFAULT=</code> to another algorithm, such as <code>CRYPT_DEFAULT=1</code> for BSD and Linux MD5.                        |
| <code>CRYPT_DEFAULT</code>               | Specify the default algorithm for new passwords. The Oracle Solaris default is the <code>crypt_sha256(4)</code> algorithm.                                                                                                                                                                                                                   |

The key/value pair must appear on a single line, and the key must start the line. Lines starting with `#` are taken as comments and ignored. Option name comparisons are case-insensitive.

Only one `CRYPT_ALGORITHMS_ALLOW` or `CRYPT_ALGORITHMS_DEPRECATED` value can be specified. Whichever is listed first in the file takes precedence. The algorithm specified for `CRYPT_DEFAULT` must either be specified for `CRYPT_ALGORITHMS_ALLOW` or not be specified for `CRYPT_ALGORITHMS_DEPRECATED`. If `CRYPT_DEFAULT` is not specified, the default is `__unix__`.

#### Examples **EXAMPLE 1** Defining a Key/Value Pair

```
AUTHS_GRANTED=solaris.date
```

#### **EXAMPLE 2** Specifying Privileges

As noted above, you should specify privileges through negation, specifying `all` for `PRIV_LIMIT` and `basic` for `PRIV_DEFAULT`, then subtracting privileges, as shown below.

```
PRIV_LIMIT=all,!sys_linkdir
PRIV_DEFAULT=basic,!file_link_any
```

The first line, above, takes away only the `sys_linkdir` privilege. The second line takes away only the `file_link` privilege. These privilege specifications are unaffected by any future

**EXAMPLE 2** Specifying Privileges (Continued)

addition of privileges that might occur.

**Files**

|                           |                                   |
|---------------------------|-----------------------------------|
| /etc/user_attr            | Defines extended user attributes. |
| /etc/security/auth_attr   | Defines authorizations.           |
| /etc/security/prof_attr   | Defines profiles.                 |
| /etc/security/policy.conf | Defines policy for the system.    |

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Availability        | system/core-os  |
| Interface Stability | Committed       |

**See Also** [login\(1\)](#), [pfexec\(1\)](#), [chkauthattr\(3C\)](#), [getexecuser\(3C\)](#), [auth\\_attr\(4\)](#), [crypt.conf\(4\)](#), [prof\\_attr\(4\)](#), [user\\_attr\(4\)](#), [attributes\(5\)](#), [privileges\(5\)](#)

**Notes** The *console user* is defined as the owner of `/dev/console`.

**Name** priv\_names – privilege definition file

**Synopsis** /etc/security/priv\_names

**Description** The priv\_names file, located in /etc/security, defines the privileges with which a process can be associated. See [privileges\(5\)](#) for the privilege definitions. In that man page, privileges correspond to privilege names in priv\_names as shown in the following examples:

| name in <a href="#">privileges(5)</a> | Name in priv_names |
|---------------------------------------|--------------------|
| PRIV_FILE_CHOWN                       | file_chown         |
| PRIV_FILE_CHOWN_SELF                  | file_chown_self    |
| PRIV_FILE_DAC_EXECUTE                 | file_dac_execute   |

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Availability        | system/core-os  |
| Interface Stability | Committed       |

**See Also** [ppriv\(1\)](#), [attributes\(5\)](#), [privileges\(5\)](#)

**Name** `proc` – /proc, the process file system

**Description** /proc is a file system that provides access to the state of each process and light-weight process (lwp) in the system. The name of each entry in the /proc directory is a decimal number corresponding to a process-ID. These entries are themselves subdirectories. Access to process state is provided by additional files contained within each subdirectory; the hierarchy is described more completely below. In this document, “/proc file” refers to a non-directory file within the hierarchy rooted at /proc. The owner of each /proc file and subdirectory is determined by the user-ID of the process.

/proc can be mounted on any mount point, in addition to the standard /proc mount point, and can be mounted several places at once. Such additional mounts are allowed in order to facilitate the confinement of processes to subtrees of the file system via `chroot(1M)` and yet allow such processes access to commands like `ps(1)`.

Standard system calls are used to access /proc files: `open(2)`, `close(2)`, `read(2)`, and `write(2)` (including `readv(2)`, `writtev(2)`, `pread(2)`, and `pwrite(2)`). Most files describe process state and can only be opened for reading. `ctl` and `lwpcctl` (control) files permit manipulation of process state and can only be opened for writing. `as` (address space) files contain the image of the running process and can be opened for both reading and writing. An open for writing allows process control; a read-only open allows inspection but not control. In this document, we refer to the process as open for reading or writing if any of its associated /proc files is open for reading or writing.

In general, more than one process can open the same /proc file at the same time. *Exclusive open* is an advisory mechanism provided to allow controlling processes to avoid collisions with each other. A process can obtain exclusive control of a target process, with respect to other cooperating processes, if it successfully opens any /proc file in the target process for writing (the `as` or `ctl` files, or the `lwpcctl` file of any lwp) while specifying `O_EXCL` in the `open(2)`. Such an open fails if the target process is already open for writing (that is, if an `as`, `ctl`, or `lwpcctl` file is already open for writing). There can be any number of concurrent read-only opens; `O_EXCL` is ignored on opens for reading. It is recommended that the first open for writing by a controlling process use the `O_EXCL` flag; multiple controlling processes usually result in chaos.

If a process opens one of its own /proc files for writing, the open succeeds regardless of `O_EXCL` and regardless of whether some other process has the process open for writing. Self-opens do not count when another process attempts an exclusive open. (A process cannot exclude a debugger by opening itself for writing and the application of a debugger cannot prevent a process from opening itself.) All self-opens for writing are forced to be close-on-exec (see the `F_SETFD` operation of `fcntl(2)`).

Data may be transferred from or to any locations in the address space of the traced process by applying `lseek(2)` to position the `as` file at the virtual address of interest followed by `read(2)` or `write(2)` (or by using `pread(2)` or `pwrite(2)` for the combined operation). The address-map files `/proc/pid/map` and `/proc/pid/xmap` can be read to determine the accessible

areas (mappings) of the address space. I/O transfers may span contiguous mappings. An I/O request extending into an unmapped area is truncated at the boundary. A write request beginning at an unmapped virtual address fails with EIO; a read request beginning at an unmapped virtual address returns zero (an end-of-file indication).

Information and control operations are provided through additional files. `<procfs.h>` contains definitions of data structures and message formats used with these files. Some of these definitions involve the use of sets of flags. The set types `sigset_t`, `fltset_t`, and `sysset_t` correspond, respectively, to signal, fault, and system call enumerations defined in `<sys/signal.h>`, `<sys/fault.h>`, and `<sys/syscall.h>`. Each set type is large enough to hold flags for its own enumeration. Although they are of different sizes, they have a common structure and can be manipulated by these macros:

```
prfillset(&set); /* turn on all flags in set */
premptyset(&set); /* turn off all flags in set */
praddset(&set, flag); /* turn on the specified flag */
prdelset(&set, flag); /* turn off the specified flag */
r = prismember(&set, flag); /* != 0 iff flag is turned on */
```

One of `prfillset()` or `premptyset()` must be used to initialize `set` before it is used in any other operation. `flag` must be a member of the enumeration corresponding to `set`.

Every process contains at least one *light-weight process*, or *lwp*. Each *lwp* represents a flow of execution that is independently scheduled by the operating system. All *lwps* in a process share its address space as well as many other attributes. Through the use of `lwpctl` and `ctl` files as described below, it is possible to affect individual *lwps* in a process or to affect all of them at once, depending on the operation.

When the process has more than one *lwp*, a representative *lwp* is chosen by the system for certain process status files and control operations. The representative *lwp* is a stopped *lwp* only if all of the process's *lwps* are stopped; is stopped on an event of interest only if all of the *lwps* are so stopped (excluding `PR_SUSPENDED` *lwps*); is in a `PR_REQUESTED` stop only if there are no other events of interest to be found; or, failing everything else, is in a `PR_SUSPENDED` stop (implying that the process is deadlocked). See the description of the `status` file for definitions of stopped states. See the `PCSTOP` control operation for the definition of "event of interest".

The representative *lwp* remains fixed (it is chosen again on the next operation) as long as all of the *lwps* are stopped on events of interest or are in a `PR_SUSPENDED` stop and the `PCRUN` control operation is not applied to any of them.

When applied to the process control file, every `/proc` control operation that must act on an *lwp* uses the same algorithm to choose which *lwp* to act upon. Together with synchronous stopping (see `PCSET`), this enables a debugger to control a multiple-*lwp* process using only the process-level status and control files if it so chooses. More fine-grained control can be achieved using the *lwp*-specific files.



The system supports two process data models, the traditional 32-bit data model in which ints, longs and pointers are all 32 bits wide (the ILP32 data model), and on some platforms the 64-bit data model in which longs and pointers, but not ints, are 64 bits in width (the LP64 data model). In the LP64 data model some system data types, notably `size_t`, `off_t`, `time_t` and `dev_t`, grow from 32 bits to 64 bits as well.

The `/proc` interfaces described here are available to both 32-bit and 64-bit controlling processes. However, many operations attempted by a 32-bit controlling process on a 64-bit target process fails with `E_OVERFLOW` because the address space range of a 32-bit process cannot encompass a 64-bit process or because the data in some 64-bit system data type cannot be compressed to fit into the corresponding 32-bit type without loss of information. Operations that fail in this circumstance include reading and writing the address space, reading the address-map files, and setting the target process's registers. There is no restriction on operations applied by a 64-bit process to either a 32-bit or a 64-bit target processes.

The format of the contents of any `/proc` file depends on the data model of the observer (the controlling process), not on the data model of the target process. A 64-bit debugger does not have to translate the information it reads from a `/proc` file for a 32-bit process from 32-bit format to 64-bit format. However, it usually has to be aware of the data model of the target process. The `pr_dmodel` field of the `status` files indicates the target process's data model.

To help deal with system data structures that are read from 32-bit processes, a 64-bit controlling program can be compiled with the C preprocessor symbol `_SYSCALL32` defined before system header files are included. This makes explicit 32-bit fixed-width data structures (like `struct stat32`) visible to the 64-bit program. See [types32.h\(3HEAD\)](#).

## Directory Structure

At the top level, the directory `/proc` contains entries each of which names an existing process in the system. These entries are themselves directories. Except where otherwise noted, the files described below can be opened for reading only. In addition, if a process becomes a *zombie* (one that has exited but whose parent has not yet performed a [wait\(3C\)](#) upon it), most of its associated `/proc` files disappear from the hierarchy; subsequent attempts to open them, or to read or write files opened before the process exited, elicits the error `ENOENT`.

Although process state and consequently the contents of `/proc` files can change from instant to instant, a single [read\(2\)](#) of a `/proc` file is guaranteed to return a sane representation of state; that is, the read is atomic with respect to the state of the process. No such guarantee applies to successive reads applied to a `/proc` file for a running process. In addition, atomicity is not guaranteed for I/O applied to the `as` (address-space) file for a running process or for a process whose address space contains memory shared by another running process.

A number of structure definitions are used to describe the files. These structures may grow by the addition of elements at the end in future releases of the system and it is not legitimate for a program to assume that they are not.

**Structure Of /proc/pid** A given directory `/proc/pid` contains the following entries. A process can use the invisible alias `/proc/self` if it wishes to open one of its own `/proc` files (invisible in the sense that the name “self” does not appear in a directory listing of `/proc` obtained from `ls(1)`, `getdents(2)`, or `readdir(3C)`).

**contracts** A directory containing references to the contracts held by the process. Each entry is a symlink to the contract's directory under `/system/contract`. See `contract(4)`.

**as** Contains the address-space image of the process; it can be opened for both reading and writing. `lseek(2)` is used to position the file at the virtual address of interest and then the address space can be examined or changed through `read(2)` or `write(2)` (or by using `pread(2)` or `pwrite(2)` for the combined operation).

**ctl** A write-only file to which structured messages are written directing the system to change some aspect of the process's state or control its behavior in some way. The seek offset is not relevant when writing to this file. Individual lwps also have associated `lwpctl` files in the `lwp` subdirectories. A control message may be written either to the process's `ctl` file or to a specific `lwpctl` file with operation-specific effects. The effect of a control message is immediately reflected in the state of the process visible through appropriate status and information files. The types of control messages are described in detail later. See CONTROL MESSAGES.

**status** Contains state information about the process and the representative lwp. The file contains a `pstatus` structure which contains an embedded `lwpsstatus` structure for the representative lwp, as follows:

```
typedef struct pstatus {
 int pr_flags; /* flags (see below) */
 int pr_nlwp; /* number of active lwps in the process */
 int pr_nzomb; /* number of zombie lwps in the process */
 pid_t pr_pid; /* process id */
 pid_t pr_ppid; /* parent process id */
 pid_t pr_pgid; /* process group id */
 pid_t pr_sid; /* session id */
 id_t pr_aslwpid; /* obsolete */
 id_t pr_agentid; /* lwp-id of the agent lwp, if any */
 sigset_t pr_sigpend; /* set of process pending signals */
 uintptr_t pr_brkbase; /* virtual address of the process heap */
 size_t pr_brksize; /* size of the process heap, in bytes */
 uintptr_t pr_stkbase; /* virtual address of the process stack */
 size_t pr_stksize; /* size of the process stack, in bytes */
 timestruc_t pr_utime; /* process user cpu time */
 timestruc_t pr_stime; /* process system cpu time */
 timestruc_t pr_cutime; /* sum of children's user times */
 timestruc_t pr_cstime; /* sum of children's system times */
 sigset_t pr_sigtrace; /* set of traced signals */
 fltset_t pr_fltrace; /* set of traced faults */
 sysset_t pr_sysentry; /* set of system calls traced on entry */
 sysset_t pr_sysexit; /* set of system calls traced on exit */
};
```

```

 char pr_dmodel; /* data model of the process */
 taskid_t pr_taskid; /* task id */
 projid_t pr_projid; /* project id */
 zoneid_t pr_zoneid; /* zone id */
 lwpstatus_t pr_lwp; /* status of the representative lwp */
} pstatus_t;

```

`pr_flags` is a bit-mask holding the following process flags. For convenience, it also contains the lwp flags for the representative lwp, described later.

`PR_ISSYS` process is a system process (see `PCSTOP`).

`PR_VFORKP` process is the parent of a vforked child (see `PCWATCH`).

`PR_FORK` process has its inherit-on-fork mode set (see `PCSET`).

`PR_RLC` process has its run-on-last-close mode set (see `PCSET`).

`PR_KLC` process has its kill-on-last-close mode set (see `PCSET`).

`PR_ASYNC` process has its asynchronous-stop mode set (see `PCSET`).

`PR_MSACCT` Set by default in all processes to indicate that microstate accounting is enabled. However, this flag has been deprecated and no longer has any effect. Microstate accounting may not be disabled; however, it is still possible to toggle the flag.

`PR_MSFOURK` Set by default in all processes to indicate that microstate accounting is enabled for processes that this parent forks(). However, this flag has been deprecated and no longer has any effect. It is possible to toggle this flag; however, it is not possible to disable microstate accounting.

`PR_BPTADJ` process has its breakpoint adjustment mode set (see `PCSET`).

`PR_PTRACE` process has its ptrace-compatibility mode set (see `PCSET`).

`pr_nlwp` is the total number of active lwps in the process. `pr_nzomb` is the total number of zombie lwps in the process. A zombie lwp is a non-detached lwp that has terminated but has not been reaped with `thr_join(3C)` or `pthread_join(3C)`.

`pr_pid`, `pr_ppid`, `pr_pgid`, and `pr_sid` are, respectively, the process ID, the ID of the process's parent, the process's process group ID, and the process's session ID.

`pr_aslwpid` is obsolete and is always zero.

`pr_agentid` is the lwp-ID for the `/proc` agent lwp (see the `PCAGENT` control operation). It is zero if there is no agent lwp in the process.

`pr_sigpend` identifies asynchronous signals pending for the process.

`pr_brkbase` is the virtual address of the process heap and `pr_brksize` is its size in bytes. The address formed by the sum of these values is the process break (see [brk\(2\)](#)). `pr_stkbase` and `pr_stksize` are, respectively, the virtual address of the process stack and its size in bytes. (Each lwp runs on a separate stack; the distinguishing characteristic of the process stack is that the operating system grows it when necessary.)

`pr_utime`, `pr_stime`, `pr_cutime`, and `pr_cstime` are, respectively, the user CPU and system CPU time consumed by the process, and the cumulative user CPU and system CPU time consumed by the process's children, in seconds and nanoseconds.

`pr_sigtrace` and `pr_fltttrace` contain, respectively, the set of signals and the set of hardware faults that are being traced (see [PCSTRACE](#) and [PCSFAULT](#)).

`pr_sysentry` and `pr_sysexit` contain, respectively, the sets of system calls being traced on entry and exit (see [PCSENTRY](#) and [PCSEXIT](#)).

`pr_dmodel` indicates the data model of the process. Possible values are:

```
PR_MODEL_ILP32 process data model is ILP32.
PR_MODEL_LP64 process data model is LP64.
PR_MODEL_NATIVE process data model is native.
```

The `pr_taskid`, `pr_projid`, and `pr_zoneid` fields contain respectively, the numeric IDs of the task, project, and zone in which the process was running.

The constant `PR_MODEL_NATIVE` reflects the data model of the controlling process, *that is*, its value is `PR_MODEL_ILP32` or `PR_MODEL_LP64` according to whether the controlling process has been compiled as a 32-bit program or a 64-bit program, respectively.

`pr_lwp` contains the status information for the representative lwp:

```
typedef struct lwpstatus {
 int pr_flags; /* flags (see below) */
 id_t pr_lwpid; /* specific lwp identifier */
 short pr_why; /* reason for lwp stop, if stopped */
 short pr_what; /* more detailed reason */
 short pr_cursig; /* current signal, if any */
 siginfo_t pr_info; /* info associated with signal or fault */
 sigset_t pr_lwppend; /* set of signals pending to the lwp */
 sigset_t pr_lwphold; /* set of signals blocked by the lwp */
 struct sigaction pr_action; /* signal action for current signal */
 stack_t pr_altstack; /* alternate signal stack info */
 uintptr_t pr_oldcontext; /* address of previous ucontext */
 short pr_syscall; /* system call number (if in syscall) */
 short pr_nsysarg; /* number of arguments to this syscall */
 int pr_errno; /* errno for failed syscall */
 long pr_sysarg[PRSYSARGS]; /* arguments to this syscall */
};
```

```

long pr_rval1; /* primary syscall return value */
long pr_rval2; /* second syscall return value, if any */
char pr_clname[PRCLSZ]; /* scheduling class name */
timestruc_t pr_tstamp; /* real-time time stamp of stop */
timestruc_t pr_utime; /* lwp user cpu time */
timestruc_t pr_stime; /* lwp system cpu time */
uintptr_t pr_ustack; /* stack boundary data (stack_t) address */
ulong_t pr_instr; /* current instruction */
prgregset_t pr_reg; /* general registers */
prfpregset_t pr_fpreg; /* floating-point registers */
} lwpstatus_t;

```

`pr_flags` is a bit-mask holding the following lwp flags. For convenience, it also contains the process flags, described previously.

|                         |                                                                                                  |
|-------------------------|--------------------------------------------------------------------------------------------------|
| <code>PR_STOPPED</code> | The lwp is stopped.                                                                              |
| <code>PR_ISTOP</code>   | The lwp is stopped on an event of interest (see <code>PCSTOP</code> ).                           |
| <code>PR_DSTOP</code>   | The lwp has a stop directive in effect (see <code>PCSTOP</code> ).                               |
| <code>PR_STEP</code>    | The lwp has a single-step directive in effect (see <code>PCRUN</code> ).                         |
| <code>PR_ASLEEP</code>  | The lwp is in an interruptible sleep within a system call.                                       |
| <code>PR_PCINVAL</code> | The lwp's current instruction ( <code>pr_instr</code> ) is undefined.                            |
| <code>PR_DETACH</code>  | This is a detached lwp (see <code>pthread_create(3C)</code> and <code>pthread_join(3C)</code> ). |
| <code>PR_DAEMON</code>  | This is a daemon lwp (see <code>pthread_create(3C)</code> ).                                     |
| <code>PR_ASLEEP</code>  | This flag is obsolete and is never set.                                                          |
| <code>PR_AGENT</code>   | This is the /proc agent lwp for the process.                                                     |

`pr_lwpid` names the specific lwp.

`pr_why` and `pr_what` together describe, for a stopped lwp, the reason for the stop. Possible values of `pr_why` and the associated `pr_what` are:

|                           |                                                                                                                                                                                                                                                                                                              |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>PR_REQUESTED</code> | indicates that the stop occurred in response to a stop directive, normally because <code>PCSTOP</code> was applied or because another lwp stopped on an event of interest and the asynchronous-stop flag (see <code>PCSET</code> ) was not set for the process. <code>pr_what</code> is unused in this case. |
| <code>PR_SIGNALED</code>  | indicates that the lwp stopped on receipt of a signal (see <code>PCSTRACE</code> ); <code>pr_what</code> holds the signal number that caused the stop (for a newly-stopped lwp, the same value is in <code>pr_cursig</code> ).                                                                               |
| <code>PR_FAULTED</code>   | indicates that the lwp stopped on incurring a hardware fault (see <code>PCSFAULT</code> ); <code>pr_what</code> holds the fault number that caused the stop.                                                                                                                                                 |

|                            |                                                                                                                                                                                   |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>PR_SYSENTRY</code>   |                                                                                                                                                                                   |
| <code>PR_SYSEXIT</code>    | indicate a stop on entry to or exit from a system call (see <code>PCSENTRY</code> and <code>PCSEXIT</code> ); <code>pr_what</code> holds the system call number.                  |
| <code>PR_JOBCONTROL</code> | indicates that the lwp stopped due to the default action of a job control stop signal (see <a href="#">sigaction(2)</a> ); <code>pr_what</code> holds the stopping signal number. |
| <code>PR_SUSPENDED</code>  | indicates that the lwp stopped due to internal synchronization of lwps within the process. <code>pr_what</code> is unused in this case.                                           |

`pr_cursig` names the current signal, that is, the next signal to be delivered to the lwp, if any. `pr_info`, when the lwp is in a `PR_SIGNALLED` or `PR_FAULTED` stop, contains additional information pertinent to the particular signal or fault (see `<sys/siginfo.h>`).

`pr_lwppend` identifies any synchronous or directed signals pending for the lwp. `pr_lwphold` identifies those signals whose delivery is being blocked by the lwp (the signal mask).

`pr_action` contains the signal action information pertaining to the current signal (see [sigaction\(2\)](#)); it is undefined if `pr_cursig` is zero. `pr_altstack` contains the alternate signal stack information for the lwp (see [sigaltstack\(2\)](#)).

`pr_oldcontext`, if not zero, contains the address on the lwp stack of a `ucontext` structure describing the previous user-level context (see [ucontext.h\(3HEAD\)](#)). It is non-zero only if the lwp is executing in the context of a signal handler.

`pr_syscall` is the number of the system call, if any, being executed by the lwp; it is non-zero if and only if the lwp is stopped on `PR_SYSENTRY` or `PR_SYSEXIT`, or is asleep within a system call (`PR_ASLEEP` is set). If `pr_syscall` is non-zero, `pr_nsysarg` is the number of arguments to the system call and `pr_sysarg` contains the actual arguments.

`pr_rval1`, `pr_rval2`, and `pr_errno` are defined only if the lwp is stopped on `PR_SYSEXIT` or if the `PR_VFORKP` flag is set. If `pr_errno` is zero, `pr_rval1` and `pr_rval2` contain the return values from the system call. Otherwise, `pr_errno` contains the error number for the failing system call (see `<sys/errno.h>`).

`pr_clname` contains the name of the lwp's scheduling class.

`pr_tstamp`, if the lwp is stopped, contains a time stamp marking when the lwp stopped, in real time seconds and nanoseconds since an arbitrary time in the past.

`pr_utime` is the amount of user level CPU time used by this LWP.

`pr_stime` is the amount of system level CPU time used by this LWP.

`pr_ustack` is the virtual address of the `stack_t` that contains the stack boundaries for this LWP. See [getustack\(2\)](#) and [\\_stack\\_grow\(3C\)](#).

`pr_instr` contains the machine instruction to which the lwp's program counter refers. The amount of data retrieved from the process is machine-dependent. On SPARC based machines, it is a 32-bit word. On x86-based machines, it is a single byte. In general, the size is that of the machine's smallest instruction. If `PR_PCINVAL` is set, `pr_instr` is undefined; this occurs whenever the lwp is not stopped or when the program counter refers to an invalid virtual address.

`pr_reg` is an array holding the contents of a stopped lwp's general registers.

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SPARC             | On SPARC-based machines, the predefined constants <code>R_G0 ... R_G7</code> , <code>R_00 ... R_07</code> , <code>R_L0 ... R_L7</code> , <code>R_I0 ... R_I7</code> , <code>R_PC</code> , <code>R_nPC</code> , and <code>R_Y</code> can be used as indices to refer to the corresponding registers; previous register windows can be read from their overflow locations on the stack (however, see the <code>gwindows</code> file in the <code>/proc/pid/lwp/lwpid</code> subdirectory). |
| SPARC V8 (32-bit) | For SPARC V8 (32-bit) controlling processes, the predefined constants <code>R_PSR</code> , <code>R_WIM</code> , and <code>R_TBR</code> can be used as indices to refer to the corresponding special registers. For SPARC V9 (64-bit) controlling processes, the predefined constants <code>R_CCR</code> , <code>R_ASI</code> , and <code>R_FPRS</code> can be used as indices to refer to the corresponding special registers.                                                           |
| x86 (32-bit)      | For 32-bit x86 processes, the predefined constants listed below can be used as indices to refer to the corresponding registers.                                                                                                                                                                                                                                                                                                                                                          |
|                   | SS<br>UESP<br>EFL<br>CS<br>EIP<br>ERR<br>TRAPNO<br>EAX<br>ECX<br>EDX<br>EBX<br>ESP<br>EBP<br>ESI<br>EDI<br>DS<br>ES<br>GS                                                                                                                                                                                                                                                                                                                                                                |

The preceding constants are listed in `<sys/regset.h>`.

Note that a 32-bit process can run on an x86 64-bit system, using the constants listed above.

x86 (64-bit)

To read the registers of a 32- *or* a 64-bit process, a 64-bit x86 process should use the predefined constants listed below.

```
REG_GSBASE
REG_FSBASE
REG_DS
REG_ES
REG_GS
REG_FS
REG_SS
REG_RSP
REG_RFL
REG_CS
REG_RIP
REG_ERR
REG_TRAPNO
REG_RAX
REG_RCX
REG_RDX
REG_RBX
REG_RBP
REG_RSI
REG_RDI
REG_R8
REG_R9
REG_R10
REG_R11
REG_R12
REG_R13
REG_R14
REG_R15
```

The preceding constants are listed in `<sys/regset.h>`.

`pr_fpreg` is a structure holding the contents of the floating-point registers.

SPARC registers, both general and floating-point, as seen by a 64-bit controlling process are the V9 versions of the registers, even if the target process is a 32-bit (V8) process. V8 registers are a subset of the V9 registers.

If the `lwp` is not stopped, all register values are undefined.



`psinfo` Contains miscellaneous information about the process and the representative lwp needed by the `ps(1)` command. `psinfo` remains accessible after a process becomes a *zombie*. The file contains a `psinfo` structure which contains an embedded `lwpsinfo` structure for the representative lwp, as follows:

```
typedef struct psinfo {
 int pr_flag; /* process flags (DEPRECATED: see below) */
 int pr_nlwp; /* number of active lwps in the process */
 int pr_nzomb; /* number of zombie lwps in the process */
 pid_t pr_pid; /* process id */
 pid_t pr_ppid; /* process id of parent */
 pid_t pr_pgid; /* process id of process group leader */
 pid_t pr_sid; /* session id */
 uid_t pr_uid; /* real user id */
 uid_t pr_euid; /* effective user id */
 gid_t pr_gid; /* real group id */
 gid_t pr_egid; /* effective group id */
 uintptr_t pr_addr; /* DEPRECATED was address of process */
 size_t pr_size; /* size of process image in Kbytes */
 size_t pr_rssize; /* resident set size in Kbytes */
 dev_t pr_ttydev; /* controlling tty device (or PRNODEV) */
 ushort_t pr_pctcpu; /* % of recent cpu time used by all lwps */
 ushort_t pr_pctmem; /* % of system memory used by process */
 timestruc_t pr_start; /* process start time, from the epoch */
 timestruc_t pr_time; /* cpu time for this process */
 timestruc_t pr_ctime; /* cpu time for reaped children */
 char pr_fname[PRFNSZ]; /* name of exec'ed file */
 char pr_psargs[PRARGSZ]; /* initial characters of arg list */
 int pr_wstat; /* if zombie, the wait() status */
 int pr_argc; /* initial argument count */
 uintptr_t pr_argv; /* address of initial argument vector */
 uintptr_t pr_envp; /* address of initial environment vector */
 char pr_dmodel; /* data model of the process */
 lwpsinfo_t pr_lwp; /* information for representative lwp */
 taskid_t pr_taskid; /* task id */
 projid_t pr_projid; /* project id */
 poolid_t pr_poolid; /* pool id */
 zoneid_t pr_zoneid; /* zone id */
 ctid_t pr_contract; /* process contract id */
} psinfo_t;
```

Some of the entries in `psinfo`, such as `pr_addr`, used to refer to internal kernel data structures addresses. In this release the value of `pr_addr` is always 0 unless running with all privilege.

`psinfo_t.pr_flag` is a deprecated interface that should no longer be used. Applications currently relying on the `SSYS` bit in `pr_flag` should migrate to checking `PR_ISSYS` in the `pstatus` structure's `pr_flags` field.

`pr_pctcpu` and `pr_pctmem` are 16-bit binary fractions in the range 0.0 to 1.0 with the binary point to the right of the high-order bit (1.0 == 0x8000). `pr_pctcpu` is the summation over all lwps in the process.

`pr_lwp` contains the `ps(1)` information for the representative lwp. If the process is a *zombie*, `pr_nlwp`, `pr_nzomb`, and `pr_lwp.pr_lwpid` are zero and the other fields of `pr_lwp` are undefined:

```
typedef struct lwpsinfo {
 int pr_flag; /* lwp flags (DEPRECATED: see below) */
 id_t pr_lwpid; /* lwp id */
 uintptr_t pr_addr; /* DEPRECATED was internal address of lwp */
 uintptr_t pr_wchan; /* DEPRECATED was wait addr for sleeping lwp */
 char pr_stype; /* synchronization event type */
 char pr_state; /* numeric lwp state */
 char pr_sname; /* printable character for pr_state */
 char pr_nice; /* nice for cpu usage */
 short pr_syscall; /* system call number (if in syscall) */
 char pr_oldpri; /* pre-SVR4, low value is high priority */
 char pr_cpu; /* pre-SVR4, cpu usage for scheduling */
 int pr_pri; /* priority, high value = high priority */
 ushort_t pr_pctcpu; /* % of recent cpu time used by this lwp */
 timestruc_t pr_start; /* lwp start time, from the epoch */
 timestruc_t pr_time; /* cpu time for this lwp */
 char pr_clname[PRCLSZ]; /* scheduling class name */
 char pr_name[PRFNSZ]; /* name of system lwp */
 processorid_t pr_onpro; /* processor which last ran this lwp */
 processorid_t pr_bindpro; /* processor to which lwp is bound */
 psetid_t pr_bindpset; /* processor set to which lwp is bound */
 lgrp_id_t pr_lgrp /* home lgroup */
} lwpsinfo_t;
```

Some of the entries in `lwpsinfo`, such as `pr_stype`, `pr_state`, and `pr_name`, refer to internal kernel data structures and should not be expected to retain their meanings across different versions of the operating system.

In prior releases the `lwpsinfo` `pr_addr` and `pr_wchan` fields contained kernel memory addresses for the corresponding kernel data structures in this release those fields are now always 0 unless running with all privilege.

`lwpsinfo_t.pr_flag` is a deprecated interface that should no longer be used.

`pr_pctcpu` is a 16-bit binary fraction, as described above. It represents the CPU time used by the specific lwp. On a multi-processor machine, the maximum value is  $1/N$ , where  $N$  is the number of CPUs.

`pr_contract` is the id of the process contract of which the process is a member. See [contract\(4\)](#) and [process\(4\)](#).

`cred` Contains a description of the credentials associated with the process:

```
typedef struct prcred {
 uid_t pr_euid; /* effective user id */
 uid_t pr_ruid; /* real user id */
 uid_t pr_suid; /* saved user id (from exec) */
 gid_t pr_egid; /* effective group id */
 gid_t pr_rgid; /* real group id */
 gid_t pr_sgid; /* saved group id (from exec) */
 int pr_ngroups; /* number of supplementary groups */
 gid_t pr_groups[1]; /* array of supplementary groups */
} prcred_t;
```

The array of associated supplementary groups in `pr_groups` is of variable length; the `cred` file contains all of the supplementary groups. `pr_ngroups` indicates the number of supplementary groups. (See also the PCSCRED and PCSCREDX control operations.)

`priv` Contains a description of the privileges associated with the process:

```
typedef struct prpriv {
 uint32_t pr_nsets; /* number of privilege set */
 uint32_t pr_setsize; /* size of privilege set */
 uint32_t pr_infosize; /* size of supplementary data */
 priv_chunk_t pr_sets[1]; /* array of sets */
} prpriv_t;
```

The actual dimension of the `pr_sets[]` field is

```
pr_sets[pr_nsets][pr_setsize]
```

which is followed by additional information about the process state `pr_infosize` bytes in size.

The full size of the structure can be computed using `PRIV_PPRIV_SIZE(prpriv_t*)`.

`sigact` Contains an array of `sigaction` structures describing the current dispositions of all signals associated with the traced process (see [sigaction\(2\)](#)). Signal numbers are displaced by 1 from array indices, so that the action for signal number  $n$  appears in position  $n-1$  of the array.

`auxv` Contains the initial values of the process's aux vector in an array of `auxv_t` structures (see `<sys/auxv.h>`). The values are those that were passed by the operating system as startup information to the dynamic linker.

- `ldt` This file exists only on x86-based machines. It is non-empty only if the process has established a local descriptor table (LDT). If non-empty, the file contains the array of currently active LDT entries in an array of elements of type `struct ssd`, defined in `<sys/sysi86.h>`, one element for each active LDT entry.
- `map,xmap` Contain information about the virtual address map of the process. The `map` file contains an array of `prmap` structures while the `xmap` file contains an array of `prxmap` structures. Each structure describes a contiguous virtual address region in the address space of the traced process:

```
typedef struct prmap {
 uintptr_t pr_vaddr; /* virtual address of mapping */
 size_t pr_size; /* size of mapping in bytes */
 char pr_mapname[PRMAPSZ]; /* name in /proc/pid/object */
 offset_t pr_offset; /* offset into mapped object, if any */
 int pr_mflags; /* protection and attribute flags */
 int pr_pagesize; /* pagesize for this mapping in bytes */
 int pr_shmid; /* SysV shared memory identifier */
} prmap_t;

typedef struct prxmap {
 uintptr_t pr_vaddr; /* virtual address of mapping */
 size_t pr_size; /* size of mapping in bytes */
 char pr_mapname[PRMAPSZ]; /* name in /proc/pid/object */
 offset_t pr_offset; /* offset into mapped object, if any */
 int pr_mflags; /* protection and attribute flags */
 int pr_pagesize; /* pagesize for this mapping in bytes */
 int pr_shmid; /* SysV shared memory identifier */
 dev_t pr_dev; /* device of mapped object, if any */
 uint64_t pr_ino; /* inode of mapped object, if any */
 size_t pr_rss; /* pages of resident memory */
 size_t pr_anon; /* pages of resident anonymous memory */
 size_t pr_locked; /* pages of locked memory */
 uint64_t pr_hatpagesize; /* pagesize of mapping */
} prxmap_t;
```

`pr_vaddr` is the virtual address of the mapping within the traced process and `pr_size` is its size in bytes. `pr_mapname`, if it does not contain a null string, contains the name of a file in the object directory (see below) that can be opened read-only to obtain a file descriptor for the mapped file associated with the mapping. This enables a debugger to find object file symbol tables without having to know the real path names of the executable file and shared libraries of the process. `pr_offset` is the 64-bit offset within the mapped file (if any) to which the virtual address is mapped.

`pr_mflags` is a bit-mask of protection and attribute flags:

`MA_READ` mapping is readable by the traced process.

|                            |                                                                                     |
|----------------------------|-------------------------------------------------------------------------------------|
| <code>MA_WRITE</code>      | mapping is writable by the traced process.                                          |
| <code>MA_EXEC</code>       | mapping is executable by the traced process.                                        |
| <code>MA_SHARED</code>     | mapping changes are shared by the mapped object.                                    |
| <code>MA_ISM</code>        | mapping is intimate shared memory (shared MMU resources)                            |
| <code>MAP_NORESERVE</code> | mapping does not have swap space reserved (mapped with <code>MAP_NORESERVE</code> ) |
| <code>MA_SHM</code>        | mapping System V shared memory                                                      |

A contiguous area of the address space having the same underlying mapped object may appear as multiple mappings due to varying read, write, and execute attributes. The underlying mapped object does not change over the range of a single mapping. An I/O operation to a mapping marked `MA_SHARED` fails if applied at a virtual address not corresponding to a valid page in the underlying mapped object. A write to a `MA_SHARED` mapping that is not marked `MA_WRITE` fails. Reads and writes to private mappings always succeed. Reads and writes to unmapped addresses fail.

`pr_pagesize` is the page size for the mapping, currently always the system pagesize.

`pr_shmid` is the shared memory identifier, if any, for the mapping. Its value is `-1` if the mapping is not System V shared memory. See [shmget\(2\)](#).

`pr_dev` is the device of the mapped object, if any, for the mapping. Its value is `PRNODEV (-1)` if the mapping does not have a device.

`pr_ino` is the inode of the mapped object, if any, for the mapping. Its contents are only valid if `pr_dev` is not `PRNODEV`.

`pr_rss` is the number of resident pages of memory for the mapping. The number of resident bytes for the mapping may be determined by multiplying `pr_rss` by the page size given by `pr_pagesize`.

`pr_anon` is the number of resident anonymous memory pages (pages which are private to this process) for the mapping.

`pr_locked` is the number of locked pages for the mapping. Pages which are locked are always resident in memory.

`pr_hatpagesize` is the size, in bytes, of the HAT (MMU) translation for the mapping. `pr_hatpagesize` may be different than `pr_pagesize`. The possible values are hardware architecture specific, and may change over a mapping's lifetime.

`rmap` Contains information about the reserved address ranges of the process. The file contains an array of `prmap` structures, as defined above for the `map` file. Each structure describes a

contiguous virtual address region in the address space of the traced process that is reserved by the system in the sense that an `mmap(2)` system call that does not specify `MAP_FIXED` do not use any part of it for the new mapping. Examples of such reservations include the address ranges reserved for the process stack and the individual thread stacks of a multi-threaded process.

**cwd** A symbolic link to the process's current working directory. See `chdir(2)`. A `readlink(2)` of `/proc/pid/cwd` yields a null string. However, it can be opened, listed, and searched as a directory, and can be the target of `chdir(2)`.

**root** A symbolic link to the process's root directory. `/proc/pid/root` can differ from the system root directory if the process or one of its ancestors executed `chroot(2)` as super user. It has the same semantics as `/proc/pid/cwd`.

**fd** A directory containing references to the open files of the process. Each entry is a decimal number corresponding to an open file descriptor in the process.

If an entry refers to a regular file, it can be opened with normal file system semantics but, to ensure that the controlling process cannot gain greater access than the controlled process, with no file access modes other than its read/write open modes in the controlled process. If an entry refers to a directory, it can be accessed with the same semantics as `/proc/pid/cwd`. An attempt to open any other type of entry fails with `EACCESS`.

**object** A directory containing read-only files with names corresponding to the `pr_mapname` entries in the `map` and `pagedata` files. Opening such a file yields a file descriptor for the underlying mapped file associated with an address-space mapping in the process. The file name `a.out` appears in the directory as an alias for the process's executable file.

The `object` directory makes it possible for a controlling process to gain access to the object file and any shared libraries (and consequently the symbol tables) without having to know the actual path names of the executable files.

**path** A directory containing symbolic links to files opened by the process. The directory includes one entry for `cwd` and `root`. The directory also contains a numerical entry for each file descriptor in the `fd` directory, and entries matching those in the `object` directory. If this information is not available, any attempt to read the contents of the symbolic link fails. This is most common for files that do not exist in the filesystem namespace (such as FIFOs and sockets), but can also happen for regular files. For the file descriptor entries, the path may be different from the one used by the process to open the file.

**pagedata** Opening the page data file enables tracking of address space references and modifications on a per-page basis.

A `read(2)` of the page data file descriptor returns structured page data and atomically clears the page data maintained for the file by the system. That is to say, each read returns data collected since the last read; the first read returns data collected since the file was opened. When the call completes, the read buffer contains the following structure as its header and

thereafter contains a number of section header structures and associated byte arrays that must be accessed by walking linearly through the buffer.

```
typedef struct prpageheader {
 timestruc_t pr_tstamp; /* real time stamp, time of read() */
 ulong_t pr_nmap; /* number of address space mappings */
 ulong_t pr_npage; /* total number of pages */
} prpageheader_t;
```

The header is followed by `pr_nmap` `prasmmap` structures and associated data arrays. The `prasmmap` structure contains the following elements:

```
typedef struct prasmmap {
 uintptr_t pr_vaddr; /* virtual address of mapping */
 ulong_t pr_npage; /* number of pages in mapping */
 char pr_mapname[PRMAPSZ]; /* name in /proc/pid/object */
 offset_t pr_offset; /* offset into mapped object, if any */
 int pr_mflags; /* protection and attribute flags */
 int pr_pagesize; /* pagesize for this mapping in bytes */
 int pr_shmid; /* SysV shared memory identifier */
} prasmmap_t;
```

Each section header is followed by `pr_npage` bytes, one byte for each page in the mapping, plus 0-7 null bytes at the end so that the next `prasmmap` structure begins on an eight-byte aligned boundary. Each data byte may contain these flags:

`PG_REFERENCED`    page has been referenced.  
`PG_MODIFIED`      page has been modified.

If the read buffer is not large enough to contain all of the page data, the read fails with `E2BIG` and the page data is not cleared. The required size of the read buffer can be determined through `fstat(2)`. Application of `lseek(2)` to the page data file descriptor is ineffective; every read starts from the beginning of the file. Closing the page data file descriptor terminates the system overhead associated with collecting the data.

More than one page data file descriptor for the same process can be opened, up to a system-imposed limit per traced process. A read of one does not affect the data being collected by the system for the others. An open of the page data file fails with `ENOMEM` if the system-imposed limit would be exceeded.

**watch** Contains an array of `prwatch` structures, one for each watched area established by the `PCWATCH` control operation. See `PCWATCH` for details.

**usage** Contains process usage information described by a `prusage` structure which contains at least the following fields:

```
typedef struct prusage {
 id_t pr_lwpid; /* lwp id. 0: process or defunct */
```

```

int pr_count; /* number of contributing lwps */
timestruc_t pr_tstamp; /* real time stamp, time of read() */
timestruc_t pr_create; /* process/lwp creation time stamp */
timestruc_t pr_term; /* process/lwp termination time stamp */
timestruc_t pr_rtime; /* total lwp real (elapsed) time */
timestruc_t pr_utime; /* user level CPU time */
timestruc_t pr_stime; /* system call CPU time */
timestruc_t pr_ttime; /* other system trap CPU time */
timestruc_t pr_tftime; /* text page fault sleep time */
timestruc_t pr_dftime; /* data page fault sleep time */
timestruc_t pr_kftime; /* kernel page fault sleep time */
timestruc_t pr_ltime; /* user lock wait sleep time */
timestruc_t pr_slptime; /* all other sleep time */
timestruc_t pr_wtime; /* wait-cpu (latency) time */
timestruc_t pr_stoptime; /* stopped time */
ulong_t pr_minf; /* minor page faults */
ulong_t pr_majf; /* major page faults */
ulong_t pr_nswap; /* swaps */
ulong_t pr_inblk; /* input blocks */
ulong_t pr_oublk; /* output blocks */
ulong_t pr_msnd; /* messages sent */
ulong_t pr_mrcv; /* messages received */
ulong_t pr_sigs; /* signals received */
ulong_t pr_vctx; /* voluntary context switches */
ulong_t pr_ictx; /* involuntary context switches */
ulong_t pr_sysc; /* system calls */
ulong_t pr_ioch; /* chars read and written */
} prusage_t;

```

Microstate accounting is now continuously enabled. While this information was previously an estimate, if microstate accounting were not enabled, the current information is now never an estimate represents time the process has spent in various states.

**lstatus** Contains a `prheader` structure followed by an array of `lwpstatus` structures, one for each active lwp in the process (see also `/proc/pid/lwp/lwpid/lwpstatus`, below). The `prheader` structure describes the number and size of the array entries that follow.

```

typedef struct prheader {
 long pr_nent; /* number of entries */
 size_t pr_entsize; /* size of each entry, in bytes */
} prheader_t;

```

The `lwpstatus` structure may grow by the addition of elements at the end in future releases of the system. Programs must use `pr_entsize` in the file header to index through the array. These comments apply to all `/proc` files that include a `prheader` structure (`lpsinfo` and `lusage`, below).



- lpsinfo** Contains a `prheader` structure followed by an array of `lwpsinfo` structures, one for each active and zombie lwp in the process. See also `/proc/pid/lwp/lwpid/lwpsinfo`, below.
- lusage** Contains a `prheader` structure followed by an array of `prusage` structures, one for each active lwp in the process, plus an additional element at the beginning that contains the summation over all defunct lwps (lwps that once existed but no longer exist in the process). Excluding the `pr_lwpid`, `pr_tstamp`, `pr_create`, and `pr_term` entries, the entry-by-entry summation over all these structures is the definition of the process usage information obtained from the usage file. (See also `/proc/pid/lwp/lwpid/lwpusage`, below.)
- lwp** A directory containing entries each of which names an active or zombie lwp within the process. These entries are themselves directories containing additional files as described below. Only the `lwpsinfo` file exists in the directory of a zombie lwp.

**Structure Of** A given directory `/proc/pid/lwp/lwpid` contains the following entries:

`/proc/pid/lwp/  
lwpid`

- lwpid** Write-only control file. The messages written to this file affect the specific lwp rather than the representative lwp, as is the case for the process's `ctl` file.
- lwpstatus** Lwp-specific state information. This file contains the `lwpsatus` structure for the specific lwp as described above for the representative lwp in the process's `status` file.
- lwpsinfo** Lwp-specific `ps(1)` information. This file contains the `lwpsinfo` structure for the specific lwp as described above for the representative lwp in the process's `psinfo` file. The `lwpsinfo` file remains accessible after an lwp becomes a zombie.
- lwpusage** This file contains the `prusage` structure for the specific lwp as described above for the process's usage file.
- gwindows** This file exists only on SPARC based machines. If it is non-empty, it contains a `gwindows_t` structure, defined in `<sys/regset.h>`, with the values of those SPARC register windows that could not be stored on the stack when the lwp stopped. Conditions under which register windows are not stored on the stack are: the stack pointer refers to nonexistent process memory or the stack pointer is improperly aligned. If the lwp is not stopped or if there are no register windows that could not be stored on the stack, the file is empty (the usual case).
- xregs** Extra state registers. The extra state register set is architecture dependent; this file is empty if the system does not support extra state registers. If the file is non-empty, it contains an architecture dependent structure of type `prxregset_t`, defined in `<procfs.h>`, with the values of the lwp's extra state registers. If the lwp is not stopped, all register values are undefined. See also the PCSXREG control operation, below.
- asrs** This file exists only for 64-bit SPARC V9 processes. It contains an `asrset_t` structure, defined in `<sys/regset.h>`, containing the values of the lwp's platform-dependent ancillary state registers. If the lwp is not stopped, all register values are undefined. See also the PCSASRS control operation, below.

**templates** A directory which contains references to the active templates for the lwp, named by the contract type. Changes made to an active template descriptor do not affect the original template which was activated, though they do affect the active template. It is not possible to activate an active template descriptor. See [contract\(4\)](#).

**Control Messages** Process state changes are effected through messages written to a process's `ctl` file or to an individual lwp's `lwpctl` file. All control messages consist of a `long` that names the specific operation followed by additional data containing the operand, if any.

Multiple control messages may be combined in a single [write\(2\)](#) (or [writev\(2\)](#)) to a control file, but no partial writes are permitted. That is, each control message, operation code plus operand, if any, must be presented in its entirety to the [write\(2\)](#) and not in pieces over several system calls. If a control operation fails, no subsequent operations contained in the same [write\(2\)](#) are attempted.

Descriptions of the allowable control messages follow. In all cases, writing a message to a control file for a process or lwp that has terminated elicits the error `ENOENT`.

**PCSTOP PCDSTOP  
PCWSTOP PCTWSTOP** When applied to the process control file, `PCSTOP` directs all lwps to stop and waits for them to stop, `PCDSTOP` directs all lwps to stop without waiting for them to stop, and `PCWSTOP` simply waits for all lwps to stop. When applied to an lwp control file, `PCSTOP` directs the specific lwp to stop and waits until it has stopped, `PCDSTOP` directs the specific lwp to stop without waiting for it to stop, and `PCWSTOP` simply waits for the specific lwp to stop. When applied to an lwp control file, `PCSTOP` and `PCWSTOP` complete when the lwp stops on an event of interest, immediately if already so stopped; when applied to the process control file, they complete when every lwp has stopped either on an event of interest or on a `PR_SUSPENDED` stop.

`PCTWSTOP` is identical to `PCWSTOP` except that it enables the operation to time out, to avoid waiting forever for a process or lwp that may never stop on an event of interest. `PCTWSTOP` takes a `long` operand specifying a number of milliseconds; the wait terminates successfully after the specified number of milliseconds even if the process or lwp has not stopped; a timeout value of zero makes the operation identical to `PCWSTOP`.

An “event of interest” is either a `PR_REQUESTED` stop or a stop that has been specified in the process's tracing flags (set by `PCSTRACE`, `PCSFAULT`, `PCSENTRY`, and `PCSEXIT`). `PR_JOBCONTROL` and `PR_SUSPENDED` stops are specifically not events of interest. (An lwp may stop twice due to a stop signal, first showing `PR_SIGNALED` if the signal is traced and again showing `PR_JOBCONTROL` if the lwp is set running without clearing the signal.) If `PCSTOP` or `PCDSTOP` is applied to an lwp that is stopped, but not on an event of interest, the stop directive takes effect when the lwp is restarted by the competing mechanism. At that time, the lwp enters a `PR_REQUESTED` stop before executing any user-level code.

A write of a control message that blocks is interruptible by a signal so that, for example, an [alarm\(2\)](#) can be set to avoid waiting forever for a process or lwp that may never stop on an

event of interest. If PCSTOP is interrupted, the lwp stop directives remain in effect even though the `write(2)` returns an error. (Use of PCTWSTOP with a non-zero timeout is recommended over PCWSTOP with an `alarm(2)`.)

A system process (indicated by the PR\_ISSYS flag) never executes at user level, has no user-level address space visible through `/proc`, and cannot be stopped. Applying one of these operations to a system process or any of its lwps elicits the error EBUSY.

- PCRUN Make an lwp runnable again after a stop. This operation takes a long operand containing zero or more of the following flags:
- PRCSIG clears the current signal, if any (see PCCSIG).
  - PRCFAULT clears the current fault, if any (see PCCFAULT).
  - PRSTEP directs the lwp to execute a single machine instruction. On completion of the instruction, a trace trap occurs. If FLTTRACE is being traced, the lwp stops; otherwise, it is sent SIGTRAP. If SIGTRAP is being traced and is not blocked, the lwp stops. When the lwp stops on an event of interest, the single-step directive is cancelled, even if the stop occurs before the instruction is executed. This operation requires hardware and operating system support and may not be implemented on all processors. It is implemented on SPARC and x86-based machines.
  - PR SABORT is meaningful only if the lwp is in a PR\_SYSENTRY stop or is marked PR\_ASLEEP; it instructs the lwp to abort execution of the system call (see PCSENTRY and PCSEXIT).
  - PRSTOP directs the lwp to stop again as soon as possible after resuming execution (see PCDSTOP). In particular, if the lwp is stopped on PR\_SIGNALLED or PR\_FAULTED, the next stop shows PR\_REQUESTED, no other stop intervenes, and the lwp does not execute any user-level code.

When applied to an lwp control file, PCRUN clears any outstanding directed-stop request and makes the specific lwp runnable. The operation fails with EBUSY if the specific lwp is not stopped on an event of interest or has not been directed to stop or if the agent lwp exists and this is not the agent lwp (see PCAGENT).

When applied to the process control file, a representative lwp is chosen for the operation as described for `/proc/pid/status`. The operation fails with EBUSY if the representative lwp is not stopped on an event of interest or has not been directed to stop or if the agent lwp exists. If PRSTEP or PRSTOP was requested, the representative lwp is made runnable and its outstanding directed-stop request is cleared; otherwise all outstanding directed-stop requests are cleared and, if it was stopped on an event of interest, the representative lwp is marked PR\_REQUESTED. If, as a consequence, all lwps are in the PR\_REQUESTED or PR\_SUSPENDED stop state, all lwps showing PR\_REQUESTED are made runnable.

**PCSTRACE** Define a set of signals to be traced in the process. The receipt of one of these signals by an lwp causes the lwp to stop. The set of signals is defined using an operand `sigset_t` contained in the control message. Receipt of SIGKILL cannot be traced; if specified, it is silently ignored.

If a signal that is included in an lwp's held signal set (the signal mask) is sent to the lwp, the signal is not received and does not cause a stop until it is removed from the held signal set, either by the lwp itself or by setting the held signal set with PCSHOLD.

**PCCSIG** The current signal, if any, is cleared from the specific or representative lwp.

**PCSSIG** The current signal and its associated signal information for the specific or representative lwp are set according to the contents of the operand `siginfo` structure (see `<sys/siginfo.h>`). If the specified signal number is zero, the current signal is cleared. The semantics of this operation are different from those of `kill(2)` in that the signal is delivered to the lwp immediately after execution is resumed (even if it is being blocked) and an additional PR\_SIGNALLED stop does not intervene even if the signal is traced. Setting the current signal to SIGKILL terminates the process immediately.

**PCKILL** If applied to the process control file, a signal is sent to the process with semantics identical to those of `kill(2)`. If applied to an lwp control file, a directed signal is sent to the specific lwp. The signal is named in a long operand contained in the message. Sending SIGKILL terminates the process immediately.

**PCUNKILL** A signal is deleted, that is, it is removed from the set of pending signals. If applied to the process control file, the signal is deleted from the process's pending signals. If applied to an lwp control file, the signal is deleted from the lwp's pending signals. The current signal (if any) is unaffected. The signal is named in a long operand in the control message. It is an error (EINVAL) to attempt to delete SIGKILL.

**PCSHOLD** Set the set of held signals for the specific or representative lwp (signals whose delivery is blocked if sent to the lwp). The set of signals is specified with a `sigset_t` operand. SIGKILL and SIGSTOP cannot be held; if specified, they are silently ignored.

**PCSFault** Define a set of hardware faults to be traced in the process. On incurring one of these faults, an lwp stops. The set is defined via the operand `fltset_t` structure. Fault names are defined in `<sys/fault.h>` and include the following. Some of these may not occur on all processors; there may be processor-specific faults in addition to these.

**FLTILL** illegal instruction

**FLTPRIV** privileged instruction

**FLTBPT** breakpoint trap

**FLTTRACE** trace trap (single-step)

**FLTWATCH** watchpoint trap

|           |                                 |
|-----------|---------------------------------|
| FLTACCESS | memory access fault (bus error) |
| FLTBOUNDS | memory bounds violation         |
| FLTIOVF   | integer overflow                |
| FLTIZDIV  | integer zero divide             |
| FLTFPE    | floating-point exception        |
| FLTSTACK  | unrecoverable stack fault       |
| FLTPAGE   | recoverable page fault          |

When not traced, a fault normally results in the posting of a signal to the lwp that incurred the fault. If an lwp stops on a fault, the signal is posted to the lwp when execution is resumed unless the fault is cleared by PCCFAULT or by the PRCFAULT option of PCRUN. FLTPAGE is an exception; no signal is posted. The `pr_info` field in the `lwpsstatus` structure identifies the signal to be sent and contains machine-specific information about the fault.

**PCCFAULT** The current fault, if any, is cleared; the associated signal is not sent to the specific or representative lwp.

**PCSENTRY PCSEXIT** These control operations instruct the process's lwps to stop on entry to or exit from specified system calls. The set of system calls to be traced is defined via an operand `sysset_t` structure.

When entry to a system call is being traced, an lwp stops after having begun the call to the system but before the system call arguments have been fetched from the lwp. When exit from a system call is being traced, an lwp stops on completion of the system call just prior to checking for signals and returning to user level. At this point, all return values have been stored into the lwp's registers.

If an lwp is stopped on entry to a system call (`PR_SYSENTRY`) or when sleeping in an interruptible system call (`PR_ASLEEP` is set), it may be instructed to go directly to system call exit by specifying the `PR_ABORT` flag in a `PCRUN` control message. Unless exit from the system call is being traced, the lwp returns to user level showing `EINTR`.

**PCWATCH** Set or clear a watched area in the controlled process from a `prwatch` structure operand:

```
typedef struct prwatch {
 uintptr_t pr_vaddr; /* virtual address of watched area */
 size_t pr_size; /* size of watched area in bytes */
 int pr_wflags; /* watch type flags */
} prwatch_t;
```

`pr_vaddr` specifies the virtual address of an area of memory to be watched in the controlled process. `pr_size` specifies the size of the area, in bytes. `pr_wflags` specifies the type of memory access to be monitored as a bit-mask of the following flags:

**WA\_READ** read access

|              |                                      |
|--------------|--------------------------------------|
| WA_WRITE     | write access                         |
| WA_EXEC      | execution access                     |
| WA_TRAPAFTER | trap after the instruction completes |

If `pr_wflags` is non-empty, a watched area is established for the virtual address range specified by `pr_vaddr` and `pr_size`. If `pr_wflags` is empty, any previously-established watched area starting at the specified virtual address is cleared; `pr_size` is ignored.

A watchpoint is triggered when an lwp in the traced process makes a memory reference that covers at least one byte of a watched area and the memory reference is as specified in `pr_wflags`. When an lwp triggers a watchpoint, it incurs a watchpoint trap. If `FLTWATCH` is being traced, the lwp stops; otherwise, it is sent a `SIGTRAP` signal; if `SIGTRAP` is being traced and is not blocked, the lwp stops.

The watchpoint trap occurs before the instruction completes unless `WA_TRAPAFTER` was specified, in which case it occurs after the instruction completes. If it occurs before completion, the memory is not modified. If it occurs after completion, the memory is modified (if the access is a write access).

Physical i/o is an exception for watchpoint traps. In this instance, there is no guarantee that memory before the watched area has already been modified (or in the case of `WA_TRAPAFTER`, that the memory following the watched area has not been modified) when the watchpoint trap occurs and the lwp stops.

`pr_info` in the `lwpsatus` structure contains information pertinent to the watchpoint trap. In particular, the `si_addr` field contains the virtual address of the memory reference that triggered the watchpoint, and the `si_code` field contains one of `TRAP_RWATCH`, `TRAP_WWATCH`, or `TRAP_XWATCH`, indicating read, write, or execute access, respectively. The `si_trapafter` field is zero unless `WA_TRAPAFTER` is in effect for this watched area; non-zero indicates that the current instruction is not the instruction that incurred the watchpoint trap. The `si_pc` field contains the virtual address of the instruction that incurred the trap.

A watchpoint trap may be triggered while executing a system call that makes reference to the traced process's memory. The lwp that is executing the system call incurs the watchpoint trap while still in the system call. If it stops as a result, the `lwpsatus` structure contains the system call number and its arguments. If the lwp does not stop, or if it is set running again without clearing the signal or fault, the system call fails with `EFAULT`. If `WA_TRAPAFTER` was specified, the memory reference is completed and the memory is modified (if the access was a write access) when the watchpoint trap occurs.

If more than one of `WA_READ`, `WA_WRITE`, and `WA_EXEC` is specified for a watched area, and a single instruction incurs more than one of the specified types, only one is reported when the watchpoint trap occurs. The precedence is `WA_EXEC`, `WA_READ`, `WA_WRITE` (`WA_EXEC` and

WA\_READ take precedence over WA\_WRITE), unless WA\_TRAPAFTER was specified, in which case it is WA\_WRITE, WA\_READ, WA\_EXEC (WA\_WRITE takes precedence).

PCWATCH fails with EINVAL if an attempt is made to specify overlapping watched areas or if pr\_wflags contains flags other than those specified above. It fails with ENOMEM if an attempt is made to establish more watched areas than the system can support (the system can support thousands).

The child of a `vfork(2)` borrows the parent's address space. When a `vfork(2)` is executed by a traced process, all watched areas established for the parent are suspended until the child terminates or performs an `exec(2)`. Any watched areas established independently in the child are cancelled when the parent resumes after the child's termination or `exec(2)`. PCWATCH fails with EBUSY if applied to the parent of a `vfork(2)` before the child has terminated or performed an `exec(2)`. The PR\_VFORKP flag is set in the `pstatus` structure for such a parent process.

Certain accesses of the traced process's address space by the operating system are immune to watchpoints. The initial construction of a signal stack frame when a signal is delivered to an lwp does not trigger a watchpoint trap even if the new frame covers watched areas of the stack. Once the signal handler is entered, watchpoint traps occur normally. On SPARC based machines, register window overflow and underflow does not trigger watchpoint traps, even if the register window save areas cover watched areas of the stack.

Watched areas are not inherited by child processes, even if the traced process's inherit-on-fork mode, PR\_FORK, is set (see PCSET, below). All watched areas are cancelled when the traced process performs a successful `exec(2)`.

PCSET PCUNSET PCSET sets one or more modes of operation for the traced process. PCUNSET unsets these modes. The modes to be set or unset are specified by flags in an operand `long` in the control message:

- PR\_FORK (inherit-on-fork): When set, the process's tracing flags and its inherit-on-fork mode are inherited by the child of a `fork(2)`, `fork1(2)`, or `vfork(2)`. When unset, child processes start with all tracing flags cleared.
- PR\_RLC (run-on-last-close): When set and the last writable `/proc` file descriptor referring to the traced process or any of its lwps is closed, all of the process's tracing flags and watched areas are cleared, any outstanding stop directives are canceled, and if any lwps are stopped on events of interest, they are set running as though PCRUN had been applied to them. When unset, the process's tracing flags and watched areas are retained and lwps are not set running on last close.
- PR\_KLC (kill-on-last-close): When set and the last writable `/proc` file descriptor referring to the traced process or any of its lwps is closed, the process is terminated with SIGKILL.

- PR\_ASYNC** (asynchronous-stop): When set, a stop on an event of interest by one lwp does not directly affect any other lwp in the process. When unset and an lwp stops on an event of interest other than `PR_REQUESTED`, all other lwps in the process are directed to stop.
- PR\_MSACCT** (microstate accounting): Microstate accounting is now continuously enabled. This flag is deprecated and no longer has any effect upon microstate accounting. Applications may toggle this flag; however, microstate accounting remains enabled regardless.
- PR\_MSFOURK** (inherit microstate accounting): All processes now inherit microstate accounting, as it is continuously enabled. This flag has been deprecated and its use no longer has any effect upon the behavior of microstate accounting.
- PR\_BPTADJ** (breakpoint trap pc adjustment): On x86-based machines, a breakpoint trap leaves the program counter (the EIP) referring to the breakpointed instruction plus one byte. When `PR_BPTADJ` is set, the system adjusts the program counter back to the location of the breakpointed instruction when the lwp stops on a breakpoint. This flag has no effect on SPARC based machines, where breakpoint traps leave the program counter referring to the breakpointed instruction.
- PR\_PTRACE** (ptrace-compatibility): When set, a stop on an event of interest by the traced process is reported to the parent of the traced process by `wait(3C)`, `SIGTRAP` is sent to the traced process when it executes a successful `exec(2)`, `setuid/setgid` flags are not honored for execs performed by the traced process, any exec of an object file that the traced process cannot read fails, and the process dies when its parent dies. This mode is deprecated; it is provided only to allow `ptrace(3C)` to be implemented as a library function using `/proc`.

It is an error (`EINVAL`) to specify flags other than those described above or to apply these operations to a system process. The current modes are reported in the `pr_flags` field of `/proc/pid/status` and `/proc/pid/lwp/lwp/status`.

- PCSREG** Set the general registers for the specific or representative lwp according to the operand `prgset_t` structure.

On SPARC based systems, only the condition-code bits of the processor-status register (`R_PSR`) of SPARC V8 (32-bit) processes can be modified by `PCSREG`. Other privileged registers cannot be modified at all.

On x86-based systems, only certain bits of the flags register (EFL) can be modified by `PCSREG`: these include the condition codes, direction-bit, and overflow-bit.

`PCSREG` fails with `EBUSY` if the lwp is not stopped on an event of interest.



- PCSVADDR** Set the address at which execution resumes for the specific or representative lwp from the operand long. On SPARC based systems, both %pc and %npc are set, with %npc set to the instruction following the virtual address. On x86-based systems, only %eip is set. PCSVADDR fails with EBUSY if the lwp is not stopped on an event of interest.
- PCSFREG** Set the floating-point registers for the specific or representative lwp according to the operand `prfprgset_t` structure. An error (EINVAL) is returned if the system does not support floating-point operations (no floating-point hardware and the system does not emulate floating-point machine instructions). PCSFREG fails with EBUSY if the lwp is not stopped on an event of interest.
- PCSXREG** Set the extra state registers for the specific or representative lwp according to the architecture-dependent operand `prxregset_t` structure. An error (EINVAL) is returned if the system does not support extra state registers. PCSXREG fails with EBUSY if the lwp is not stopped on an event of interest.
- PCSASRS** Set the ancillary state registers for the specific or representative lwp according to the SPARC V9 platform-dependent operand `asrset_t` structure. An error (EINVAL) is returned if either the target process or the controlling process is not a 64-bit SPARC V9 process. Most of the ancillary state registers are privileged registers that cannot be modified. Only those that can be modified are set; all others are silently ignored. PCSASRS fails with EBUSY if the lwp is not stopped on an event of interest.
- PCAGENT** Create an agent lwp in the controlled process with register values from the operand `prgprgset_t` structure (see PCSREG, above). The agent lwp is created in the stopped state showing PR\_REQUESTED and with its held signal set (the signal mask) having all signals except SIGKILL and SIGSTOP blocked.

The PCAGENT operation fails with EBUSY unless the process is fully stopped via `/proc`, that is, unless all of the lwps in the process are stopped either on events of interest or on PR\_SUSPENDED, or are stopped on PR\_JOBCONTROL and have been directed to stop via PCDSTOP. It fails with EBUSY if an agent lwp already exists. It fails with ENOMEM if system resources for creating new lwps have been exhausted.

Any PCRUN operation applied to the process control file or to the control file of an lwp other than the agent lwp fails with EBUSY as long as the agent lwp exists. The agent lwp must be caused to terminate by executing the `SYS_lwp_exit` system call trap before the process can be restarted.

Once the agent lwp is created, its lwp-ID can be found by reading the process status file. To facilitate opening the agent lwp's control and status files, the directory name `/proc/pid/lwp/agent` is accepted for lookup operations as an invisible alias for `/proc/pid/lwp/lwpid`, `lwpid` being the lwp-ID of the agent lwp (invisible in the sense that the name "agent" does not appear in a directory listing of `/proc/pid/lwp` obtained from `ls(1)`, `getdents(2)`, or `readdir(3C)`).

The purpose of the agent lwp is to perform operations in the controlled process on behalf of the controlling process: to gather information not directly available via `/proc` files, or in general to make the process change state in ways not directly available via `/proc` control operations. To make use of an agent lwp, the controlling process must be capable of making it execute system calls (specifically, the `SYS_lwp_exit` system call trap). The register values given to the agent lwp on creation are typically the registers of the representative lwp, so that the agent lwp can use its stack.

The agent lwp is not allowed to execute any variation of the `SYS_fork` or `SYS_exec` system call traps. Attempts to do so yield `ENOTSUP` to the agent lwp.

Symbolic constants for system call trap numbers like `SYS_lwp_exit` and `SYS_lwp_create` can be found in the header file `<sys/syscall.h>`.

**PCREAD PCWRITE** Read or write the target process's address space via a `priovec` structure operand:

```
typedef struct priovec {
 void *pio_base; /* buffer in controlling process */
 size_t pio_len; /* size of read/write request in bytes */
 off_t pio_offset; /* virtual address in target process */
} priovec_t;
```

These operations have the same effect as `pread(2)` and `pwrite(2)`, respectively, of the target process's address space file. The difference is that more than one `PCREAD` or `PCWRITE` control operation can be written to the control file at once, and they can be interspersed with other control operations in a single write to the control file. This is useful, for example, when planting many breakpoint instructions in the process's address space, or when stepping over a breakpointed instruction. Unlike `pread(2)` and `pwrite(2)`, no provision is made for partial reads or writes; if the operation cannot be performed completely, it fails with `EIO`.

- PCNICE** The traced process's `nice(2)` value is incremented by the amount in the operand `long`. Only a process with the `{PRIV_PROC_PRIOCNTL}` privilege asserted in its effective set can better a process's priority in this way, but any user may lower the priority. This operation is not meaningful for all scheduling classes.
- PCSCRED** Set the target process credentials to the values contained in the `prcred_t` structure operand (see `/proc/pid/cred`). The effective, real, and saved user-IDs and group-IDs of the target process are set. The target process's supplementary groups are not changed; the `pr_ngroups` and `pr_groups` members of the structure operand are ignored. Only the privileged processes can perform this operation; for all others it fails with `EPERM`.
- PCSCREDX** Operates like `PCSCRED` but also sets the supplementary groups; the length of the data written with this control operation should be “`sizeof(prcred_t) + sizeof(gid_t) * (#groups - 1)`”.

**PCSPRIV** Set the target process privilege to the values contained in the `prpriv_t` operand (see `/proc/pid/priv`). The effective, permitted, inheritable, and limit sets are all changed. Privilege flags can also be set. The process is made privilege aware unless it can relinquish privilege awareness. See [privileges\(5\)](#).

The limit set of the target process cannot be grown. The other privilege sets must be subsets of the intersection of the effective set of the calling process with the new limit set of the target process or subsets of the original values of the sets in the target process.

If any of the above restrictions are not met, `EPERM` is returned. If the structure written is improperly formatted, `EINVAL` is returned.

**Programming Notes** For security reasons, except for the `psinfo`, `usage`, `lpsinfo`, `lusage`, `lwpsinfo`, and `lwpsusage` files, which are world-readable, and except for privileged processes, an open of a `/proc` file fails unless both the user-ID and group-ID of the caller match those of the traced process and the process's object file is readable by the caller. The effective set of the caller is a superset of both the inheritable and the permitted set of the target process. The limit set of the caller is a superset of the limit set of the target process. Except for the world-readable files just mentioned, files corresponding to `setuid` and `setgid` processes can be opened only by the appropriately privileged process.

A process that is missing the basic privilege `{PRIV_PROC_INFO}` cannot see any processes under `/proc` that it cannot send a signal to.

A process that has `{PRIV_PROC_OWNER}` asserted in its effective set can open any file for reading. To manipulate or control a process, the controlling process must have at least as many privileges in its effective set as the target process has in its effective, inheritable, and permitted sets. The limit set of the controlling process must be a superset of the limit set of the target process. Additional restrictions apply if any of the uids of the target process are 0. See [privileges\(5\)](#).

Even if held by a privileged process, an open process or lwp file descriptor (other than file descriptors for the world-readable files) becomes invalid if the traced process performs an [exec\(2\)](#) of a `setuid`/`setgid` object file or an object file that the traced process cannot read. Any operation performed on an invalid file descriptor, except [close\(2\)](#), fails with `EAGAIN`. In this situation, if any tracing flags are set and the process or any lwp file descriptor is open for writing, the process is directed to stop and its run-on-last-close flag is set (see `PCSET`). This enables a controlling process (if it has permission) to reopen the `/proc` files to get new valid file descriptors, close the invalid file descriptors, unset the run-on-last-close flag (if desired), and proceed. Just closing the invalid file descriptors causes the traced process to resume execution with all tracing flags cleared. Any process not currently open for writing via `/proc`, but that has left-over tracing flags from a previous open, and that executes a `setuid`/`setgid` or unreadable object file, are not stopped but have all its tracing flags cleared.

To wait for one or more of a set of processes or lwps to stop or terminate, /proc file descriptors (other than those obtained by opening the `cwd` or root directories or by opening files in the `fd` or object directories) can be used in a `poll(2)` system call. When requested and returned, either of the polling events `POLLPRI` or `POLLWRNORM` indicates that the process or lwp stopped on an event of interest. Although they cannot be requested, the polling events `POLLHUP`, `POLLERR`, and `POLLNVAL` may be returned. `POLLHUP` indicates that the process or lwp has terminated. `POLLERR` indicates that the file descriptor has become invalid. `POLLNVAL` is returned immediately if `POLLPRI` or `POLLWRNORM` is requested on a file descriptor referring to a system process (see `PCSTOP`). The requested events may be empty to wait simply for termination.

|              |                             |                                              |
|--------------|-----------------------------|----------------------------------------------|
| <b>Files</b> | /proc                       | directory (list of processes)                |
|              | /proc/ <i>pid</i>           | specific process directory                   |
|              | /proc/self                  | alias for a process's own directory          |
|              | /proc/ <i>pid</i> /as       | address space file                           |
|              | /proc/ <i>pid</i> /ctl      | process control file                         |
|              | /proc/ <i>pid</i> /status   | process status                               |
|              | /proc/ <i>pid</i> /lstatus  | array of lwp status structs                  |
|              | /proc/ <i>pid</i> /psinfo   | process <code>ps(1)</code> info              |
|              | /proc/ <i>pid</i> /lpsinfo  | array of lwp <code>ps(1)</code> info structs |
|              | /proc/ <i>pid</i> /map      | address space map                            |
|              | /proc/ <i>pid</i> /xmap     | extended address space map                   |
|              | /proc/ <i>pid</i> /rmap     | reserved address map                         |
|              | /proc/ <i>pid</i> /cred     | process credentials                          |
|              | /proc/ <i>pid</i> /priv     | process privileges                           |
|              | /proc/ <i>pid</i> /sigact   | process signal actions                       |
|              | /proc/ <i>pid</i> /auxv     | process aux vector                           |
|              | /proc/ <i>pid</i> /ldt      | process LDT (x86 only)                       |
|              | /proc/ <i>pid</i> /usage    | process usage                                |
|              | /proc/ <i>pid</i> /lusage   | array of lwp usage structs                   |
|              | /proc/ <i>pid</i> /path     | symbolic links to process open files         |
|              | /proc/ <i>pid</i> /pagedata | process page data                            |
|              | /proc/ <i>pid</i> /watch    | active watchpoints                           |

|                                            |                                           |
|--------------------------------------------|-------------------------------------------|
| <code>/proc/pid/cwd</code>                 | alias for the current working directory   |
| <code>/proc/pid/root</code>                | alias for the root directory              |
| <code>/proc/pid/fd</code>                  | directory (list of open files)            |
| <code>/proc/pid/fd/*</code>                | aliases for process's open files          |
| <code>/proc/pid/object</code>              | directory (list of mapped files)          |
| <code>/proc/pid/object/a.out</code>        | alias for process's executable file       |
| <code>/proc/pid/object/*</code>            | aliases for other mapped files            |
| <code>/proc/pid/lwp</code>                 | directory (list of lwps)                  |
| <code>/proc/pid/lwp/lwpid</code>           | specific lwp directory                    |
| <code>/proc/pid/lwp/agent</code>           | alias for the agent lwp directory         |
| <code>/proc/pid/lwp/lwpid/lwpctl</code>    | lwp control file                          |
| <code>/proc/pid/lwp/lwpid/lwpstatus</code> | lwp status                                |
| <code>/proc/pid/lwp/lwpid/lwpsinfo</code>  | lwp <code>ps(1)</code> info               |
| <code>/proc/pid/lwp/lwpid/lwpusage</code>  | lwp usage                                 |
| <code>/proc/pid/lwp/lwpid/gwindows</code>  | register windows (SPARC only)             |
| <code>/proc/pid/lwp/lwpid/xregs</code>     | extra state registers                     |
| <code>/proc/pid/lwp/lwpid/asrs</code>      | ancillary state registers (SPARC V9 only) |

**See Also** [ls\(1\)](#), [ps\(1\)](#), [chroot\(1M\)](#), [alarm\(2\)](#), [brk\(2\)](#), [chdir\(2\)](#), [chroot\(2\)](#), [close\(2\)](#), [creat\(2\)](#), [dup\(2\)](#), [exec\(2\)](#), [fcntl\(2\)](#), [fork\(2\)](#), [fork1\(2\)](#), [fstat\(2\)](#), [getdents\(2\)](#), [getustack\(2\)](#), [kill\(2\)](#), [lseek\(2\)](#), [mmap\(2\)](#), [nice\(2\)](#), [open\(2\)](#), [poll\(2\)](#), [pread\(2\)](#), [ptrace\(3C\)](#), [pwrite\(2\)](#), [read\(2\)](#), [readlink\(2\)](#), [readv\(2\)](#), [shmget\(2\)](#), [sigaction\(2\)](#), [sigaltstack\(2\)](#), [vfork\(2\)](#), [write\(2\)](#), [writev\(2\)](#), [\\_stack\\_grow\(3C\)](#), [readdir\(3C\)](#), [pthread\\_create\(3C\)](#), [pthread\\_join\(3C\)](#), [siginfo.h\(3HEAD\)](#), [signal.h\(3HEAD\)](#), [thr\\_create\(3C\)](#), [thr\\_join\(3C\)](#), [types32.h\(3HEAD\)](#), [ucontext.h\(3HEAD\)](#), [wait\(3C\)](#), [contract\(4\)](#), [process\(4\)](#), [lfcompile\(5\)](#), [privileges\(5\)](#)

**Diagnostics** Errors that can occur in addition to the errors normally associated with file system access:

|        |                                                                                                                                                                                |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| E2BIG  | Data to be returned in a <a href="#">read(2)</a> of the page data file exceeds the size of the read buffer provided by the caller.                                             |
| EACCES | An attempt was made to examine a process that ran under a different uid than the controlling process and <code>{PRIV_PROC_OWNER}</code> was not asserted in the effective set. |

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EAGAIN     | The traced process has performed an <code>exec(2)</code> of a <code>setuid/setgid</code> object file or of an object file that it cannot read; all further operations on the process or <code>lwp</code> file descriptor (except <code>close(2)</code> ) elicit this error.                                                                                                                                                                                                                                                                                                                                                                                  |
| EBUSY      | <code>PCSTOP</code> , <code>PCDSTOP</code> , <code>PCWSTOP</code> , or <code>PCTWSTOP</code> was applied to a system process; an exclusive <code>open(2)</code> was attempted on a <code>/proc</code> file for a process already open for writing; <code>PCRUN</code> , <code>PCSREG</code> , <code>PCSVADDR</code> , <code>PCSFREG</code> , or <code>PCSXREG</code> was applied to a process or <code>lwp</code> not stopped on an event of interest; an attempt was made to mount <code>/proc</code> when it was already mounted; <code>PCAGENT</code> was applied to a process that was not fully stopped or that already had an agent <code>lwp</code> . |
| EINVAL     | In general, this means that some invalid argument was supplied to a system call. A non-exhaustive list of conditions eliciting this error includes: a control message operation code is undefined; an out-of-range signal number was specified with <code>PCSSIG</code> , <code>PCKILL</code> , or <code>PCUNKILL</code> ; <code>SIGKILL</code> was specified with <code>PCUNKILL</code> ; <code>PCSFREG</code> was applied on a system that does not support floating-point operations; <code>PCSXREG</code> was applied on a system that does not support extra state registers.                                                                           |
| EINTR      | A signal was received by the controlling process while waiting for the traced process or <code>lwp</code> to stop via <code>PCSTOP</code> , <code>PCWSTOP</code> , or <code>PCTWSTOP</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| EIO        | A <code>write(2)</code> was attempted at an illegal address in the traced process.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| ENOENT     | The traced process or <code>lwp</code> has terminated after being opened. The basic privilege <code>{PRIV_PROC_INFO}</code> is not asserted in the effective set of the calling process and the calling process cannot send a signal to the target process.                                                                                                                                                                                                                                                                                                                                                                                                  |
| ENOMEM     | The system-imposed limit on the number of page data file descriptors was reached on an <code>open</code> of <code>/proc/pid/pagedata</code> ; an attempt was made with <code>PCWATCH</code> to establish more watched areas than the system can support; the <code>PCAGENT</code> operation was issued when the system was out of resources for creating <code>lwps</code> .                                                                                                                                                                                                                                                                                 |
| ENOSYS     | An attempt was made to perform an unsupported operation (such as <code>creat(2)</code> , <code>link(2)</code> , or <code>unlink(2)</code> ) on an entry in <code>/proc</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| E_OVERFLOW | A 32-bit controlling process attempted to read or write the <code>as</code> file or attempted to read the <code>map</code> , <code>rmap</code> , or <code>pagedata</code> file of a 64-bit target process. A 32-bit controlling process attempted to apply one of the control operations <code>PCSREG</code> , <code>PCSXREG</code> , <code>PCSVADDR</code> , <code>PCWATCH</code> , <code>PCAGENT</code> , <code>PCREAD</code> , <code>PCWRITE</code> to a 64-bit target process.                                                                                                                                                                           |
| EPERM      | The process that issued the <code>PCSCRED</code> or <code>PCSCREDX</code> operation did not have the <code>{PRIV_PROC_SETID}</code> privilege asserted in its effective set, or the process that issued the <code>PCNICE</code> operation did not have the <code>{PRIV_PROC_PRIOCNTL}</code> in its effective set.                                                                                                                                                                                                                                                                                                                                           |

---

An attempt was made to control a process of which the E, P, and I privilege sets were not a subset of the effective set of the controlling process or the limit set of the controlling process is not a superset of limit set of the controlled process.

Any of the uids of the target process are 0 or an attempt was made to change any of the uids to 0 using PCSCRED and the security policy imposed additional restrictions. See [privileges\(5\)](#).

**Notes** Descriptions of structures in this document include only interesting structure elements, not filler and padding fields, and may show elements out of order for descriptive clarity. The actual structure definitions are contained in `<procfs.h>`.

**Bugs** Because the old [ioctl\(2\)](#)-based version of `/proc` is currently supported for binary compatibility with old applications, the top-level directory for a process, `/proc/pid`, is not world-readable, but it is world-searchable. Thus, anyone can open `/proc/pid/psinfo` even though [ls\(1\)](#) applied to `/proc/pid` fails for anyone but the owner or an appropriately privileged process. Support for the old [ioctl\(2\)](#)-based version of `/proc` will be dropped in a future release, at which time the top-level directory for a process will be made world-readable.

On SPARC based machines, the types `gregset_t` and `fpregset_t` defined in `<sys/regset.h>` are similar to but not the same as the types `prgregset_t` and `prfpregset_t` defined in `<procfs.h>`.

**Name** process – process contract type

**Synopsis** /system/contract/process

**Description** Process contracts allow processes to create a fault boundary around a set of subprocesses and observe events which occur within that boundary.

Process contracts are managed using the [contract\(4\)](#) file system and the [libcontract\(3LIB\)](#) library. The process contract type directory is /system/contract/process.

**CREATION** A process contract is created when an LWP that has an active process contract template calls [fork\(2\)](#). Initially, the child process created by `fork()` is the only resource managed by the contract. When an LWP that does not have an active process contract template calls `fork()`, the child process created by `fork()` is added as a resource to the process contract of which the parent was a member.

**EVENTTYPES** The following events types are defined:

`CT_PR_EV_EMPTY`

The last member of the process contract exited.

`CT_PR_EV_FORK`

A new process has been added to the process contract.

`CT_PR_EV_EXIT`

A member of the process contract exited.

`CT_PR_EV_CORE`

A process failed and dumped core. This could also occur if the process would have dumped core had appropriate [coreadm\(1M\)](#) options been enabled and core file size was unlimited.

`CT_PR_EV_SIGNAL`

A process received a fatal signal from a process, other than the owner of the process contract, that is a member of a different process contract.

`CT_PR_EV_HWERR`

A process was killed because of an uncorrectable hardware error.

**TERMS** The following common contract terms, defined in [contract\(4\)](#), have process-contract specific attributes:

critical event set

The default value for the critical event set is `(CT_PR_EV_EMPTY | CT_PR_EV_HWERR)`.

An attempt by a user without the `{PRIV_CONTRACT_EVENT}` privilege in its effective set to add an event, other than `CT_PR_EV_EMPTY`, to the critical event set which is not present in the fatal set, or if the `CT_PR_PGONLY` parameter is set and the same user attempts to add any event, other than `CT_PR_EV_EMPTY`, to the critical event set, fails.



### informative event set

The default value for the informative event set is `(CT_PR_EV_CORE | CT_PR_EV_SIGNAL)`.

The following contract terms can be read from or written to a process contract template using the named `libcontract(3LIB)` interfaces. These contract terms are in addition to those described in `contract(4)`.

### creator's aux

Auxiliary contract description. The purpose of this field is to provide the contract creator with a way to differentiate process contracts it creates under the same service FMRI. Use `ct_pr_tmpl_set_svc_aux(3CONTRACT)` to set this term. The default value is an empty string. The contents of this field should be limited to 7-bit ASCII values.

### fatal event set

Defines a set of events which, when generated, causes all members of the process contract to be killed with SIGKILL, or the intersection of the contract and the containing process group if the `CT_PR_PGRPONLY` parameter is set. Set this term with `ct_pr_tmpl_set_fatal(3CONTRACT)`. The fatal event set is restricted to `CT_PR_EV_CORE`, `CT_PR_EV_SIGNAL`, and `CT_PR_EV_HWERR`. For `CT_PR_EV_CORE` and `CT_PR_EV_SIGNAL` events, the scope of SIGKILL is limited to those processes which the contract author or the event source could have normally sent signals to.

The default value for the fatal event set is `CT_PR_EV_HWERR`.

If a user without the `{PRIV_CONTRACT_EVENT}` privilege in its effective set removes an event from the fatal event set which is present in the critical event set, the corresponding event is automatically removed from the critical event set and added to the informative event set.

### parameter set

Defines miscellaneous other settings. Use `ct_pr_tmpl_set_param(3CONTRACT)` to set this term.

The default parameter set is empty.

The value is a bit vector comprised of some or all of:

#### CT\_PR\_INHERIT

If set, indicates that the process contract is to be inherited by the process contract the contract owner is a member of if the contract owner exits before explicitly abandoning the process contract.

If not set, the process contract is automatically abandoned when the owner exits.

#### CT\_PR\_NOORPHAN

If set, all processes in a process contract are sent SIGKILL if the process contract is abandoned, either explicitly or because the holder died and `CT_PR_INHERIT` was not set. The scope of SIGKILL is limited to those processes which the contract author or the event source could have normally sent signals to.

If this is not set and the process contract is abandoned, the process contract is orphaned, that is, continues to exist without owner.

#### CT\_PR\_PGRPONLY

If set, only those processes within the same process group and process contract as a fatal error-generating process are killed.

If not set, all processes within the process contract are killed if a member process encounters an error specified in the fatal set.

If a user without the {PRIV\_CONTRACT\_EVENT} privilege in its effective set adds CT\_PR\_PGRPONLY to a template's parameter set, any events other than CT\_PR\_EV\_EMPTY are automatically removed from the critical event set and added to the informative event set.

#### CT\_PR\_REGENT

If set, the process contract can inherit unabandoned contracts left by exiting member processes.

If not set, indicates that the process contract should not inherit contracts from member processes. If a process exits before abandoning a contract it owns and is a member of a process contract which does not have CT\_PR\_REGENT set, the system automatically abandons the contract.

If a regent process contract has inherited contracts and is abandoned by its owner, its inherited contracts are abandoned.

#### service FMRI

Specifies the service FMRI associated with the process contract. Use `ct_pr_tmpl_set_svc_fmri(3CONTRACT)` to set this term. The default is to inherit the value from the creator's process contract. When this term is uninitialized, `ct_pr_tmpl_get_svc_fmri(3CONTRACT)` returns the token string `inherited:` to indicate the value has not been set and is inherited. Setting the service FMRI to `inherited:` clears the current (B value and the term is inherited from the creator's process contract. To set this term a process must have {PRIV\_CONTRACT\_IDENTITY} in its effective set.

#### transfer contract

Specifies the ID of an empty process contract held by the caller whose inherited process contracts are to be transferred to the newly created contract. Use `ct_pr_tmpl_set_t_transfer(3CONTRACT)` to set the transfer contract. Attempts to specify a contract not held by the calling process, or a contract which still has processes in it, fail.

The default transfer term is `0`, that is, no contract.

**STATUS** In addition to the standard items, the status object read from a status file descriptor contains the following items to obtain this information respectively:

### service contract ID

Specifies the process contract id which defined the service FMRI term. Use [ct\\_pr\\_status\\_get\\_svc\\_ctid\(3CONTRACT\)](#) to read the term's value. It can be used to determine if the service FMRI was inherited as in the example below.

```
ctid_t ctid; /* our contract id */
int fd; /* fd of ctid's status file */

ct_stathdl(Bt status;
ctid_t svc_ctid;

if (ct_status_read(fd, CTD_FIXED, &status) == 0) {
 if (ct_pr_status_get_svc_ctid(status, &svc_ctid) == 0) {
 if (svc_ctid == ctid)
 /* not inherited */
 else
 /* inherited */
 }
 ct_status_free(status);
}
```

If CTD\_ALL is specified, the following items are also available:

### Member list

The PIDs of processes which are members of the process contract. Use [ct\\_pr\\_status\\_get\\_members\(3CONTRACT\)](#) for this information.

### Inherited contract list

The IDs of contracts which have been inherited by the process contract. Use [ct\\_pr\\_status\\_get\\_contracts\(3CONTRACT\)](#) to obtain this information.

### Service FMRI (term)

Values equal to the terms used when the contract was written. The Service FMRI term of the process contract of a process en(Btering a zone has the value `svc:/system/zone_enter:default` when read from the non-global zone.

### contract creator

Specifies the process that created the process contract. Use [ct\\_pr\\_status\\_get\\_svc\\_creator\(3CONTRACT\)](#) to read the term's value.

### creator's aux (term)

Values equal to the terms used when the contract was written.

The following standard status items have different meanings in some situations:

### Ownership state

If the process contract has a state of CTS\_OWNED or CTS\_INHERITED and is held by an entity in the global zone, but contains processes in a non-global zone, it appears to have the state CTS\_OWNED when observed by processes in the non-global zone.

### Contract holder

If the process contract has a state of `CTS_OWNED` or `CTS_INHERITED` and is held by an entity in the global zone, but contains processes in a non-global zone, it appears to be held by the non-global zone's `zsched` when observed by processes in the non-global zone.

**EVENTS** In addition to the standard items, an event generated by a process contract contains the following information:

### Generating PID

The process ID of the member process which experienced the event, or caused the contract event to be generated (in the case of `CT_PR_EV_EMPTY`). Use `ct_pr_event_get_pid(3CONTRACT)` to obtain this information.

If the event type is `CT_PR_EV_FORK`, the event contains:

### Parent PID

The process ID which forked [Generating PID]. Use `ct_pr_event_get_ppid(3CONTRACT)` to obtain this information.

If the event type is `CT_PR_EV_EXIT`, the event contains:

### Exit status

The exit status of the process. Use `ct_pr_event_get_exitstatus(3CONTRACT)` to obtain this information.

If the event type is `CT_PR_EV_CORE`, the event can contain:

### Process core name

The name of the per-process core file. Use `ct_pr_event_get_pcorefile(3CONTRACT)` to obtain this information.

### Global core name

The name of the process's zone's global core file. Use `ct_pr_event_get_gcorefile(3CONTRACT)` to obtain this information.

### Zone core name

The name of the system-wide core file in the global zone. Use `ct_pr_event_get_zcorefile(3CONTRACT)` to obtain this information.

See [coreadm\(1M\)](#) for more information about per-process, global, and system-wide core files.

If the event type is `CT_PR_EV_SIGNAL`, the event contains:

### Signal

The number of the signal which killed the process. Use `ct_pr_event_get_signal(3CONTRACT)` to obtain this information.

It can contain:

sender

The PID of the process which sent the signal. Use `ct_pr_event_get_sender(3CONTRACT)` to obtain this information.

**Files** `/usr/include/sys/contract/process.h`  
Contains definitions of event-type macros.

**See Also** `ctrun(1)`, `ctstat(1)`, `ctwatch(1)`, `coreadm(1M)`, `close(2)`, `fork(2)`, `ioctl(2)`, `open(2)`, `poll(2)`, `ct_pr_event_get_exitstatus(3CONTRACT)`, `ct_pr_event_get_gcorefile(3CONTRACT)`, `ct_pr_event_get_pcorefile(3CONTRACT)`, `ct_pr_event_get_pid(3CONTRACT)`, `ct_pr_event_get_ppid(3CONTRACT)`, `ct_pr_event_get_signal(3CONTRACT)`, `ct_pr_event_get_zcorefile(3CONTRACT)`, `ct_pr_status_get_contracts(3CONTRACT)`, `ct_pr_status_get_members(3CONTRACT)`, `ct_pr_status_get_param(3CONTRACT)`, `ct_pr_tmpl_set_fatal(3CONTRACT)`, `ct_pr_tmpl_set_param(3CONTRACT)`, `ct_pr_tmpl_set_transfer(3CONTRACT)`, `ct_tmpl_set_cookie(3CONTRACT)`, `ct_tmpl_set_critical(3CONTRACT)`, `ct_tmpl_set_informative(3CONTRACT)`, `libcontract(3LIB)`, `contract(4)`, `privileges(5)`

**Name** prof\_attr – profile description database

**Synopsis** /etc/security/prof\_attr

**Description** /etc/security/prof\_attr is a local source for execution profile names, descriptions, and other attributes of execution profiles. The prof\_attr file can be used with other profile sources, including the prof\_attr NIS map. Programs use the [getprofattr\(3C\)](#) routines to gain access to this information.

The search order for multiple prof\_attr sources is specified in the /etc/nsswitch.conf file, as described in the [nsswitch.conf\(4\)](#) man page.

An execution profile is a mechanism used to bundle together the commands and authorizations needed to perform a specific function. An execution profile can also contain other execution profiles. Each entry in the prof\_attr database consists of one line of text containing five fields separated by colons (:). Line continuations using the backslash (\) character are permitted. The format of each entry is:

*profname:res1:res2:desc:attr*

*profname* The name of the profile. Profile names are case-sensitive.

*res1* The characters R0 in this field indicate it is read only and not modifiable by the tools that update this database.

*res2* Reserved for future use.

*desc* A long description. This field should explain the purpose of the profile, including what type of user would be interested in using it. The long description should be suitable for displaying in the help text of an application.

*attr* An optional list of semicolon-separated (;) key-value pairs that describe the security attributes to apply to the object upon execution. Zero or more keys can be specified. The following keys are currently interpreted by the system:

*help* is assigned the name of a file ending in .htm or .html.

*auths* specifies a comma-separated list of authorization names chosen from those names defined in the [auth\\_attr\(4\)](#) database. Authorization names can be specified using the asterisk (\*) character as a wildcard. For example, *solaris.printer.\** would mean all of Sun's authorizations for printing.

*audit\_flags* specifies per-user Audit preselection flags as colon-separated always-audit-flags and never-audit-flags. For example, *audit\_flags=always-audit-flags:never-audit-flags*. See [audit\\_flags\(5\)](#).

*profiles* specifies a comma-separated list of profile names chosen from those names defined in the prof\_attr database.

`privs` specifies a comma-separated list of privileges names chosen from those names defined in the `priv_names(4)` database. These privileges can then be used for executing commands with `pfexec(1)`.

`defaultpriv` and `limitpriv` have the same semantics as in `user_attr(4)`. If they are not specified in the `user_attr` database, the assigned profiles are searched until a match is found.

`audit_flags` specifies per-user audit preselection flags as a colon-separated list of `always-audit-flags` and `never-audit-flags` values, for example, `audit_flags=always-audit-flags:never-audit-flags`. See `audit_flags(5)`.

#### Examples EXAMPLE 1 Allowing Execution of All Commands

The following entry allows the user to execute all commands:

```
All:::Use this profile to give a :help=All.html
```

#### EXAMPLE 2 Consulting the Local prof\_attr File First

With the following `nsswitch.conf` entry, the local `prof_attr` file is consulted before the NIS map:

```
prof_attr: files nis
```

#### Files /etc/nsswitch.conf

`/etc/security/prof_attr`      Locally added entries. Make sure that the shipped header remains intact.

`/etc/security/prof_attr.d/*`    Entries added by package installation.

**Notes** The root user is usually defined in local databases because root needs to be able to log in and do system maintenance in single-user mode and at other times when the network name service databases are not available. So that the profile definitions for root can be located at such times, root's profiles should be defined in the local `prof_attr` file, and the order shown in the example `nsswitch.conf(4)` file entry under EXAMPLES is highly recommended.

Because the list of legal keys is likely to expand, any code that parses this database must be written to ignore unknown key-value pairs without error. When any new keywords are created, the names should be prefixed with a unique string, such as the company's stock symbol, to avoid potential naming conflicts.

Each application has its own requirements for whether the `help` value must be a relative pathname ending with a filename or the name of a file. The only known requirement is for the name of a file.

The following characters are used in describing the database format and must be escaped with a backslash if used as data: colon (:), semicolon (;), equals (=), and backslash (\).

The following authorizations are required to set various fields:

|             |                                                                 |                                   |
|-------------|-----------------------------------------------------------------|-----------------------------------|
| prof        | namename of profile                                             | solaris.profile.manage            |
| desc        | description of profile                                          | solaris.profile.manage            |
| help        | html help file name<br>of profile                               | solaris.profile.manage            |
| auths       | authorizations granted                                          | solaris.auth.assign/delegate      |
| profiles    | profiles granted                                                | solaris.profile.assign/delegate   |
| privs       | privileges granted                                              | solaris.privilege.assign/delegate |
| limitpriv   | the limit set of<br>privileges for the<br>command process       | solaris.privilege.assign/delegate |
| defaultpriv | the inheritable set of<br>privileges for the<br>command process | solaris.privilege.assign/delegate |

The value of `limitpriv` that can be set by an authorized user for a given command are limited to the `limitpriv` privileges that are granted to the user.

The value of the `defaultpriv` that can be set by an authorized user for a given command are limited to the `defaultpriv` privileges granted to the user.

The `solaris.auth.assign` authorization allows the authorized user to grant any authorization to another user. The `solaris.auth.delegate` allows the authorized user to grant only the user's authorizations to another user. The same principle applies to profiles, privileges.

**See Also** [auths\(1\)](#), [pfexec\(1\)](#), [profiles\(1\)](#), [getauthattr\(3C\)](#), [getprofattr\(3C\)](#), [getuserattr\(3C\)](#), [auth\\_attr\(4\)](#), [exec\\_attr\(4\)](#), [priv\\_names\(4\)](#), [user\\_attr\(4\)](#), [audit\\_flags\(5\)](#)



**Name** profile – setting up an environment for user at login time

**Synopsis** /etc/profile  
\$HOME/.profile

**Description** All users who have the shell, [sh\(1\)](#), as their login command have the commands in these files executed as part of their login sequence.

/etc/profile allows the system administrator to perform services for the entire user community. Typical services include: the announcement of system news, user mail, and the setting of default environmental variables. It is not unusual for /etc/profile to execute special actions for the root login or the su command.

The file \$HOME/.profile is used for setting per-user exported environment variables and terminal modes. The following example is typical (except for the comments):

```
Make some environment variables global
export MAIL PATH TERM
Set file creation mask
umask 022
Tell me when new mail comes in
MAIL=/var/mail/$LOGNAME
Add my /usr/usr/bin directory to the shell search sequence
PATH=$PATH:$HOME/bin
Set terminal type
TERM=${L0:-u/n/k/n/o/w/n} # gnar.invalid
while :
do
 if [-f ${TERMINFO:-/usr/share/lib/terminfo}/?/$TERM]
 then break
 elif [-f /usr/share/lib/terminfo/?/$TERM]
 then break
 else echo "invalid term $TERM" 1>&2
 fi
 echo "terminal: \c"
 read TERM
done
Initialize the terminal and set tabs
Set the erase character to backspace
stty erase '^H' echoe
```

**Files** \$HOME/.profile      user-specific environment  
/etc/profile            system-wide environment

**See Also** [env\(1\)](#), [login\(1\)](#), [mail\(1\)](#), [sh\(1\)](#), [stty\(1\)](#), [tput\(1\)](#), [su\(1M\)](#), [terminfo\(4\)](#), [environ\(5\)](#), [term\(5\)](#)

*Solaris Advanced User's Guide*

**Notes** Care must be taken in providing system-wide services in `/etc/profile`. Personal `.profile` files are better for serving all but the most global needs.

**Name** project – project file

**Description** The project file is a local source of project information. The project file can be used in conjunction with other project sources, including the NIS maps `project.byname` and `project.bynumber` and the LDAP database `project`. Programs use the [getproject\(3PROJECT\)](#) routines to access this information.

The project file contains a one-line entry for each project recognized by the system, of the form:

```
projname:projid:comment:user-list:group-list:attributes
```

where the fields are defined as:

*projname* The name of the project. The name must be a string that consists of alphanumeric characters, underline (`_`) characters, hyphens (`-`), and periods (`.`). The period, which is reserved for projects with special meaning to the operating system, can be used only in the names of default projects for users. *projname* cannot contain colons (`:`) or newline characters.

*projid* The project's unique numerical ID (PROJID) within the system. The maximum value of the *projid* field is `MAXPROJID`. Project IDs below 100 are reserved for the use of the operating system.

*comment* The project's description.

*user-list* A comma-separated list of users allowed in the project. With the exception of the special projects referred to below, an empty field indicates no users are allowed.

*group-list* A comma-separated list of groups of users allowed in the project. With the exception of the special projects referred to below, an empty field indicates no groups are allowed.

*attributes* A semicolon-separated list of name value pairs. Each pair has the following format:

```
name[=value]
```

where *name* is the arbitrary string specifying the key's name and *value* is the optional key value. An explanation of the valid name-value pair syntax is provided in the `USAGE` section of this page. The expected most frequent use of the attribute field is for the specification of resource controls. See [resource\\_controls\(5\)](#) for a description of the resource controls supported in the current release of the Solaris operating system. You can also use the attribute field for resource caps (see [rcapd\(1M\)](#)) and for the `project.pool` attribute (see [setproject\(3PROJECT\)](#)).

Null entries (empty fields) in the *user-list* and *group-list* fields, which normally mean “no users” and “no groups”, respectively, have a different meaning in the entries for three special projects, `user.username`, `group.groupname`, and `default`. See [getproject\(3PROJECT\)](#) for a description of these projects.

Wildcards can be used in *user-list* and *group-list* fields of the project database entry. The asterisk (\*), allows all users or groups to join the project. The exclamation mark followed by the asterisk (!\*), excludes all users or groups from the project. The exclamation mark (!) followed by a username or groupname excludes the specified user or group from the project. See EXAMPLES.

Malformed entries cause routines that read this file to halt, in which case project assignments specified further along are never made. Blank lines are treated as malformed entries in the project file, and cause [getproject\(3PROJECT\)](#) and derived interfaces to fail.

#### Examples EXAMPLE 1 Sample project File

The following is a sample project file:

```
system:0:System:::
user.root:1:Super-User:::
noproject:2:No Project:::
default:3:::
group.staff:10:::
beatles:100:The Beatles:john,paul,george,ringo::task.max-lwps=
 (privileged,100,signal=SIGTERM),(privileged,110,deny);
 process.max-file-descriptor
```

The two line breaks in the line that begins with `beatles` are not valid in a project file. They are shown here only to allow the example to display on a printed or displayed page. Each entry must be on one and only one line.

An example project entry for [nsswitch.conf\(4\)](#) is:

```
project: files nis
```

With these entries, the project `beatles` has members `john`, `paul`, `george`, and `ringo`. All projects listed in the NIS project table are effectively incorporated after the entry for `beatles`.

The `beatles` project has two values set on the `task.max-lwps` resource control. When a task in the `beatles` project requests (via one of its member processes) its 100th and 110th LWPs, an action associated with the encountered threshold triggers. Upon the request for the 100th LWP, the process making the request is sent the signal `SIGTERM` and is granted the request for an additional lightweight process (LWP). At this point, the threshold for 110 LWPs becomes the active threshold. When a request for the 110th LWP in the task is made, the requesting process is denied the request--no LWP is created. Since the 110th LWP is never granted, the threshold remains active, and all subsequent requests for an 110th LWP fails. (If LWPs are

**EXAMPLE 1** Sample project File (Continued)

given up, then subsequent requests succeeds, unless they would take the total number of LWPs across the task over 110.) The `process.max-file-descriptor` resource control is given no values. This means that processes entering this project only has the system resource control value on this `rctl`.

**EXAMPLE 2** Project Entry with Wildcards

The following entries use wildcards:

```
notroot:200:Shared Project:*,!root::
notused:300:Unused Project::!*:
```

In this example, any user except “root” is a member of project `notroot`. For the project `notused`, all groups are excluded.

**Usage** The `project` database offers a reasonably flexible attribute mechanism in the final name-value pair field. Name-value pairs are separated from one another with the semicolon (;) character. The name is in turn distinguished from the (optional) value by the equals (=) character. The value field can contain multiple values separated by the comma (,) character, with grouping support (into further values lists) by parentheses. Each of these values can be composed of the upper and lower case alphabetic characters, the digits '0' through '9', and the punctuation characters hyphen (-), plus (+), period (.), slash (/), and underscore (\_). Example resource control value specifications are provided in **EXAMPLES**, above, and in [resource\\_controls\(5\)](#) and [getprojent\(3PROJECT\)](#).

**See Also** [newtask\(1\)](#), [projects\(1\)](#), [prctl\(1\)](#), [getprojent\(3PROJECT\)](#), [setrctl\(2\)](#), [unistd.h\(3HEAD\)](#), [nsswitch.conf\(4\)](#), [resource\\_controls\(5\)](#)

**Notes** The `solaris.project.assign` authorizations allow an administrator to add a user to any project. The `solaris.project.delegate` authorization allows an administrator to only add users to the projects of which the administrator is a member.

**Name** protocols – protocol name database

**Synopsis** /etc/inet/protocols

/etc/protocols

**Description** The protocols file is a local source of information regarding the known protocols used in the DARPA Internet. The protocols file can be used in conjunction with or instead of other protocols sources, including the NIS maps “protocols.byname” and “protocols.bynumber”. Programs use the `getprotobyname(3SOCKET)` routine to access this information.

The protocols file has one line for each protocol. The line has the following format:

*official-protocol-name protocol-number aliases*

Items are separated by any number of blanks and/or TAB characters. A ‘#’ indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. Protocol names may contain any printable character other than a field delimiter, NEWLINE, or comment character.

**Examples** EXAMPLE 1 A Sample Database

The following is a sample database:

```
#
Internet (IP) protocols
#
ip 0 IP # internet protocol, pseudo protocol number
icmp 1 ICMP # internet control message protocol
ggp 3 GGP # gateway-gateway protocol
tcp 6 TCP # transmission control protocol
egp 8 EGP # exterior gateway protocol
pup 12 PUP # PARC universal packet protocol
udp 17 UDP # user datagram protocol

#
Internet (IPv6) extension headers
#
hopopt 0 HOPOPT # Hop-by-hop options for IPv6
ipv6 41 IPv6 # IPv6 in IP encapsulation
ipv6-route 43 IPv6-Route # Routing header for IPv6
ipv6-frag 44 IPv6-Frag # Fragment header for IPv6
esp 50 ESP # Encap Security Payload for IPv6
ah 51 AH # Authentication Header for IPv6
ipv6-icmp 58 IPv6-ICMP # IPv6 internet control message protocol
ipv6-nonxt 59 IPv6-NoNxt # No next header extension header for IPv6
ipv6-opts 60 IPv6-Opts # Destination Options for IPv6
```

**Files** /etc/nsswitch.conf configuration file for name-service switch

**See Also** [getprotobyname\(3SOCKET\)](#), [nsswitch.conf\(4\)](#)

**Notes** /etc/inet/protocols is the official SVR4 name of the protocols file. The symbolic link /etc/protocols exists for BSD compatibility.

**Name** prototype – package information file

**Description** prototype is an ASCII file used to specify package information. Each entry in the file describes a single deliverable object. An object can be a data file, directory, source file, executable object, and so forth. This file is generated by the package developer.

Entries in a prototype file consist of several fields of information separated by white space. Comment lines begin with a “#” and are ignored. The fields are described below and must appear in the order shown.

*part* An optional field designating the part number in which the object resides. A part is a collection of files and is the atomic unit by which a package is processed. A developer can choose criteria for grouping files into a part (for example, based on class). If this field is not used, part 1 is assumed.

*ftype* A one-character field that indicates the file type. Valid values are:

- b block special device
- c character special device
- d directory
- e a file to be edited upon installation or removal (can be shared by several packages)
- f a standard executable or data file
- i installation script or information file
- l linked file
- p named pipe
- s symbolic link
- v volatile file (one whose contents are expected to change, like a log file)
- x an exclusive directory accessible only by this package

*class* The installation class to which the file belongs. This name can be no longer than 64 characters. The field is not specified for installation scripts. (admin and all classes beginning with capital letters are reserved class names.)

*pathname* The pathname where the file resides on the target machine, for example, /usr/bin/mail or bin/ras/proc. Relative pathnames (those that do not begin with a slash) indicate that the file is relocatable. The form

*path1=path2*

can be used for two purposes: to define a link and to define local pathnames.



For linked files, *path1* indicates the destination of the link and *path2* indicates the source file. (This format is mandatory for linked files.)

For local pathnames, *path1* indicates the pathname an object should have on the machine where the entry is to be installed and *path2* indicates either a relative or fixed pathname to a file on the host machine which contains the actual contents.

A pathname can contain a variable specification of the form *\$variable*. If *variable* begins with a lower case letter, it is a build variable. If *variable* begins with an upper case letter, it is an install variable. Build variables are bound at build time. If an install variable is known at build time, its definition is inserted into the `pkginfo(4)` file so that it is available at install time. If an install variable is not known at build time, it is bound at install time.

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>major</i> | The major device number. The field is only specified for block or character special devices.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <i>minor</i> | The minor device number. The field is only specified for block or character special devices.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <i>mode</i>  | The octal mode of the file (for example, 0664). A question mark (?) indicates that the mode is left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.<br><br>The mode can be a variable specification of the form <i>\$variable</i> . If <i>variable</i> begins with a lower case letter, it is a build variable. If <i>variable</i> begins with an upper case letter, it is an install variable. Build variables are bound at build time. If an install variable is known at build time, its definition is inserted into the <code>pkginfo(4)</code> file so that it is available at install time. If an install variable is not known at build time, it is bound at install time.                                                      |
| <i>owner</i> | 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 is left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.<br><br>The owner can be a variable specification of the form <i>\$variable</i> . If <i>variable</i> begins with a lower case letter, it is a build variable. If <i>variable</i> begins with an upper case letter, it is an install variable. Build variables are bound at build time. If an install variable is known at build time, its definition is inserted into the <code>pkginfo(4)</code> file so that it is available at install time. If an install variable is not known at build time, it is bound at install time. |
| <i>group</i> | 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                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

is left unchanged, implying that the file already exists on the target machine. This field is not used for linked files or packaging information files.

The group can be a variable specification of the form  $\$variable$ . If *variable* begins with a lower case letter, it is a build variable. If *variable* begins with an upper case letter, it is an install variable. Build variables are bound at build time. If an install variable is known at build time, its definition is inserted into the [pkginfo\(4\)](#) file so that it is available at install time. If an install variable is not known at build time, it is bound at install time.

An exclamation point (!) at the beginning of a line indicates that the line contains a command. These commands are used to incorporate files in other directories, to locate objects on a host machine, and to set permanent defaults. The following commands are available:

|                          |                                                                                                                                                                                                                                                                              |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>search</code>      | Specifies a list of directories (separated by white space) to search for when looking for file contents on the host machine. The base name of the <i>path</i> field is appended to each directory in the ordered list until the file is located. Searches are not recursive. |
| <code>include</code>     | Specifies a pathname which points to another prototype file to include. Note that <code>search</code> requests do not span <code>include</code> files.                                                                                                                       |
| <code>default</code>     | Specifies a list of attributes (mode, owner, and group) to be used by default if attribute information is not provided for prototype entries which require the information. The defaults do not apply to entries in <code>include</code> prototype files.                    |
| <code>param=value</code> | Places the indicated parameter in the current environment. Spans to subsequent included prototype files.                                                                                                                                                                     |

The above commands can have variable substitutions embedded within them, as demonstrated in the two example prototype files below.

Before files are overwritten during installation, they are copied to a temporary pathname. The exception to this rule is files whose mode includes execute permission, unless the file is editable (that is, *ftype* is *e*). For files which meet this exception, the existing version is linked to a temporary pathname, and the original file is removed. This allows processes which are executing during installation to be overwritten.

#### Examples

```
EXAMPLE 1 Example 1:
!PROJDIR=/usr/proj
!BIN=$PROJDIR/bin
!CFG=$PROJDIR/cfg
!LIB=$PROJDIR/lib
!HDRS=$PROJDIR/hdrs
!search /usr/myname/usr/bin /usr/myname/src /usr/myname/hdrs
i pkginfo=/usr/myname/wrap/pkginfo
```

**EXAMPLE 1** Example 1: *(Continued)*

```

i depend=/usr/myname/wrap/depend
i version=/usr/myname/wrap/version
d none /usr/wrap 0755 root bin
d none /usr/wrap/usr/bin 0755 root bin
! search $BIN
f none /usr/wrap/bin/INSTALL 0755 root bin
f none /usr/wrap/bin/REMOVE 0755 root bin
f none /usr/wrap/bin/addpkg 0755 root bin
!default 755 root bin
f none /usr/wrap/bin/audit
f none /usr/wrap/bin/listpkg
f none /usr/wrap/bin/pkgmk
the following file starts out zero length but grows
v none /usr/wrap/logfile=/dev/null 0644 root bin
the following specifies a link (dest=src)
l none /usr/wrap/src/addpkg=/usr/wrap/bin/rmpkg
! search $SRC
!default 644 root other
f src /usr/wrap/src/INSTALL.sh
f src /usr/wrap/src/REMOVE.sh
f src /usr/wrap/src/addpkg.c
f src /usr/wrap/src/audit.c
f src /usr/wrap/src/listpkg.c
f src /usr/wrap/src/pkgmk.c
d none /usr/wrap/data 0755 root bin
d none /usr/wrap/save 0755 root bin
d none /usr/wrap/spool 0755 root bin
d none /usr/wrap/tmp 0755 root bin
d src /usr/wrap/src 0755 root bin

```

**EXAMPLE 2** Example 2:

```

this prototype is generated by 'pkgproto' to refer
to all prototypes in my src directory
!PROJDIR=/usr/dew/projx
!include $PROJDIR/src/cmd/prototype
!include $PROJDIR/src/cmd/audmerg/protofile
!include $PROJDIR/src/lib/proto

```

**See Also** [pkgmk\(1\)](#), [pkginfo\(4\)](#)

*Application Packaging Developer's Guide*

**Notes** Normally, if a file is defined in the prototype file but does not exist, that file is created at the time of package installation. However, if the file pathname includes a directory that does not exist, the file is not created. For example, if the prototype file has the following entry:

```
f none /usr/dev/bin/command
```

and that file does not exist, it is created if the directory `/usr/dev/bin` already exists or if the prototype also has an entry defining the directory:

```
d none /usr/dev/bin
```

**Name** pseudo – configuration files for pseudo device drivers

**Description** Pseudo devices are devices that are implemented entirely in software. Drivers for pseudo devices must provide driver configuration files to inform the system of each pseudo device that should be created.

Configuration files for pseudo device drivers must identify the parent driver explicitly as *pseudo*, and must create an integer property called *instance* which is unique to this entry in the configuration file.

Each entry in the configuration file creates a prototype devinfo node. Each node is assigned an instance number which is determined by the value of the *instance* property. This property is only applicable to children of the *pseudo* parent, and is required since pseudo devices have no hardware address from which to determine the instance number. See [driver.conf\(4\)](#) for further details of configuration file syntax.

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

Here is a configuration file called `ramdisk.conf` for a pseudo device driver that implements a RAM disk. This file creates two nodes called “ramdisk”. The first entry creates ramdisk node instance 0, and the second creates ramdisk node, instance 1, with the additional `disk-size` property set to 512.

```
#
Copyright (c) 1993, by Sun Microsystems, Inc.
#
#ident "@(#)ramdisk.conf 1.3 93/06/04 SMI"
name="ramdisk" parent="pseudo" instance=0;
name="ramdisk" parent="pseudo" instance=1 disk-size=512;
```

**See Also** [driver.conf\(4\)](#), [ddi\\_prop\\_op\(9F\)](#)

*Writing Device Drivers*

**Name** publickey – public key database

**Synopsis** /etc/publickey

**Description** /etc/publickey is a local public key database that is used for secure RPC. The /etc/publickey file can be used in conjunction with or instead of other publickey databases, including the NIS publickey map. Each entry in the database consists of a network user name (which may refer to either a user or a hostname), followed by the user's public key (in hex notation), a colon, and then the user's secret key encrypted with a password (also in hex notation).

The /etc/publickey file contains a default entry for nobody.

**See Also** [chkey\(1\)](#), [newkey\(1M\)](#), [getpublickey\(3NSL\)](#), [nsswitch.conf\(4\)](#)

**Name** queuedefs – queue description file for at, batch, and cron

**Synopsis** /etc/cron.d/queuedefs

**Description** The queuedefs file describes the characteristics of the queues managed by [cron\(1M\)](#). Each non-comment line in this file describes one queue. The format of the lines are as follows:

```
q.[njobj][nicen][nwaitw]
```

The fields in this line are:

- q* The name of the queue. a is the default queue for jobs started by [at\(1\)](#); b is the default queue for jobs started by `batch` (see [at\(1\)](#)); c is the default queue for jobs run from a [crontab\(1\)](#) file.
- njob* The maximum number of jobs that can be run simultaneously in that queue; if more than *njob* jobs are ready to run, only the first *njob* jobs will be run, and the others will be run as jobs that are currently running terminate. The default value is 100.
- nice* The [nice\(1\)](#) value to give to all jobs in that queue that are not run with a user ID of super-user. The default value is 2.
- nwait* The number of seconds to wait before rescheduling a job that was deferred because more than *njob* jobs were running in that job's queue, or because the system-wide limit of jobs executing has been reached. The default value is 60.

Lines beginning with # are comments, and are ignored.

**Examples** EXAMPLE 1 A sample file.

```
#
#
a.4j1n
b.2j2n90w
```

This file specifies that the a queue, for `at` jobs, can have up to 4 jobs running simultaneously; those jobs will be run with a [nice](#) value of 1. As no *nwait* value was given, if a job cannot be run because too many other jobs are running `cron` will wait 60 seconds before trying again to run it.

The b queue, for [batch\(1\)](#) jobs, can have up to 2 jobs running simultaneously; those jobs will be run with a [nice\(1\)](#) value of 2. If a job cannot be run because too many other jobs are running, [cron\(1M\)](#) will wait 90 seconds before trying again to run it. All other queues can have up to 100 jobs running simultaneously; they will be run with a [nice](#) value of 2, and if a job cannot be run because too many other jobs are running `cron` will wait 60 seconds before trying again to run it.

**Files** /etc/cron.d/queuedefs    queue description file for at, batch, and cron.

**See Also** [at\(1\)](#), [crontab\(1\)](#), [nice\(1\)](#), [cron\(1M\)](#)



**Name** rcmscript – script interface specification for the Reconfiguration and Coordination Manager

**Synopsis** *rcm\_scriptname* scriptinfo  
*rcm\_scriptname* register  
*rcm\_scriptname* resourceinfo *resourcename*  
*rcm\_scriptname* queryremove *resourcename*  
*rcm\_scriptname* preremove *resourcename*  
*rcm\_scriptname* postremove *resourcename*  
*rcm\_scriptname* undoremove *resourcename*

**Description** Reconfiguration and Coordination Manager (RCM) is a framework designed to coordinate device consumers during Solaris Dynamic Reconfiguration (DR). The interfaces specified in this man page allow device consumers, such as application vendors or site administrators, to act before and after DR operations take place by providing RCM scripts. You can write your own RCM scripts to shut down your applications, or to cleanly release the devices from your applications during dynamic remove operations.

An RCM script is an executable perl script, a shell script or a binary. Perl is the recommended language. Each script is run in its own address space using the user-id of the script file owner.

An RCM script is invoked on demand in response to DR as follows:

```
<scriptname> <command> [args ...]
```

Every script must implement the following RCM commands:

|              |                                      |
|--------------|--------------------------------------|
| scriptinfo   | Get script information.              |
| register     | Register devices the script handles. |
| resourceinfo | Get resource information.            |

A script might include some or all the of the following commands:

|             |                                               |
|-------------|-----------------------------------------------|
| queryremove | Queries whether the resource can be released. |
| preremove   | Releases the resource.                        |
| postremove  | Provides post-resource removal notification.  |
| undoremove  | Undo the actions done in preremove.           |

When a script's `register` command is run, the script should supply, in return data, all resource names the script or its application handles that could potentially be removed by DR. A resource name refers to a name in `/dev` path name.

Below is a high-level overview of the sequence of script invocations that occurs when dynamic removal of a script's registered resource is attempted. See the COMMANDS section for a detailed description of the commands.

1. Prior to removing the resource from the system during DR, the script's `queryremove` command is run:

```
<scriptname> queryremove <resourcename>
```

The script should check for obvious reasons why the resource can not be removed from the perspective of its service or application.

2. If the script indicates that the resource can be removed in the `queryremove` command. The script's `preremove` command is run:

```
<scriptname> preremove <resourcename>
```

The script releases the resource from the service or application represented by the script and prepares for the resource removal. Releasing the resource includes closing the resource if the resource is currently opened by its application.

3. The system then proceeds to remove the resource.
4. If the system has removed the resource successfully the script's `post remove` command is run:

```
<scriptname> postremove <resourcename>
```

Otherwise the script's `undoremove` command is run:

```
<scriptname> undoremove <resourcename>
```

For any commands the script does not implement, it must exit with exit status of 2. RCM silently returns success for the script's unimplemented commands.

A script performs the following basic steps:

- Takes RCM command and additional arguments from the command line and environment parameters.
- Processes the command.
- Writes the expected return data to stdout as `name=value` pairs delimited by newlines, where `name` is the name of the return data item that RCM expects and `value` is the value associated with the data item.

Environment The initial environment of RCM scripts is set as follows:

- Process UID is set to the UID of the script.
- Process GID is set to the GID of the script.
- PATH variable is set to `/usr/sbin:/usr/bin`.
- Current working directory is set to:

`/var/run` for scripts owned by root

`/tmp` for scripts not owned by root

- File descriptor 0 (stdin) is set to `/dev/null`
- Environment variable `RCM_ENV_DEBUG_LEVEL` is set to the debug level. Logging is discussed below.
- The following environment variables are also set where possible:

```
LANG
LC_COLLATE
LC_CTYPE
LC_MESSAGES
LC_MONETARY
LC_NUMERIC
LC_TIME
LC_ALL
TZ
```

See [environ\(5\)](#) for a description of these variables. See [gettext\(1\)](#) for details on retrieving localized messages.

All environment variable names beginning with `RCM_ENV_` are reserved for use by the RCM.

The character encoding used by the RCM and RCM scripts to exchange RCM commands, environment parameters, and name-value pairs is ASCII unless the controlling environment variables are specified otherwise.

## Commands

### **scriptinfo**

The `scriptinfo` command is invoked to gather information about the script.

Return data: If successful, the script must write the following name-value pairs to stdout and exit with status 0:

- `rcm_script_version=1`
- `rcm_script_func_info=script_func_info`
- `rcm_cmd_timeout=command_timeout_value`

where `script_func_info` is a localized human-readable message describing the functionality of the script.

The RCM monitors the execution time of RCM commands by RCM scripts. `command_timeout_value` is the maximum time in seconds the script is expected to take to process any RCM command except the `scriptinfo` command itself. If an RCM script does not process the RCM command and exit within this time, RCM sends a SIGABRT signal to the script process. RCM

then waits for a few seconds for the script to finish the processing of the current RCM command and exit. If the script does not exit within this time, RCM sends a SIGKILL signal to the script.

The `rcm_cmd_timeout` name-value pair is optional. It is only needed if the script is expected to take more than a few seconds to process any RCM command. Setting this name to a value of 0 (zero) disables the timer. If this name-value pair is not supplied, a default value is assigned by the RCM.

Upon failure, the script must specify the failure reason using the name-value pair `rcm_failure_reason` and exit with status 1.

### **register**

The `register` command is invoked to allow a script to specify the resources that it or its application handles that could potentially be removed by DR. The script has to supply all its resource names to RCM using the name-value pair `rcm_resource_name`.

**Return Data:** If successful, the script must write the following name-value pairs to stdout and exit with status 0:

```
rcm_resource_name=resourcename
rcm_resource_name=resourcename
.
.
.
```

where *resourcename* is the name of the resource the script is interested in.

Upon failure, the script must specify the failure reason using the name-value pair `rcm_failure_reason` and exit with status 1.

### **resourceinfo** *resourcename*

The `resourceinfo` command is invoked to get the usage information about *resourcename*.

**Return Data:** If successful, the script must write the following name-value pair to stdout and exit with status 0:

```
rcm_resource_usage_info=resource_usage
```

where *resource\_usage* is a localized human readable message describing the usage of the resource by the script.

Upon failure, the script must specify the failure reason using the name-value pair `rcm_failure_reason` and exit with status 1.

### **queryremove** *resourcename*

Prior to removing the resource from the system, the `queryremove` command is invoked to query the script to determine whether the script can release the given resource successfully from the service or application it represents. The script does not actually release the resource. The script might indicate that it is not able to release the resource if the resource is critical for its service or application.

Additional environment parameter:

|                            |                                                                                                                                                                                 |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>RCM_ENV_FORCE</code> | Can be one of:                                                                                                                                                                  |
| <code>FALSE</code>         | Normal request.                                                                                                                                                                 |
| <code>TRUE</code>          | Request is urgent. The script should check whether the resource can be released successfully by force, such as by using the <code>force</code> option to unmount a file system. |

**Return Data:** If the command succeeds, the script must return no data and exit with status 0.

If the script would not be able to release the resource, it must specify the reason using the name-value pair `rcm_failure_reason` and exit with status 3.

Upon any other failure, the script must specify the failure reason using the name-value pair `rcm_failure_reason` and exit with status 1.

#### **preremove** *resourcename*

The `preremove` command is invoked prior to an attempt to remove the given *resourcename*. In response to this command the script can either release the resource (including closing the device if the device is currently opened) from the service or application it represents or indicate that it can not release the resource if the resource is critical for its service or application.

Additional environment parameter:

|                            |                                                                                                                                                          |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>RCM_ENV_FORCE</code> | Can be one of:                                                                                                                                           |
| <code>FALSE</code>         | Normal request.                                                                                                                                          |
| <code>TRUE</code>          | Request is urgent. The script should make extra effort to release the resource, such as by using the <code>force</code> option to unmount a file system. |

**Return Data:** If the command succeeds, the script must return no data and exit with status 0.

If the script cannot release the resource, it must specify the reason using the name-value pair `rcm_failure_reason` and exit with status 3.

Upon any other failure, the script must specify the failure reason using the name-value pair `rcm_failure_reason` and exit with status 1.

#### **postremove** *resourcename*

The `postremove` command is invoked after the given *resourcename* has been removed.

**Return Data:** If the command succeeds, the script must return no data and exit with status 0.

Upon failure, the script must specify the failure reason using the name-value pair `rcm_failure_reason` and exit with status 1.

#### **undoremove** *resourcename*

The `undoremove` command is invoked to undo what was done in the previous `preremove` command for the given *resourcename*. The script can bring the state of the resource to the same state it was in when the script received the `preremove` command for that resource.

**Return Data:** If the command succeeds, the script must return no data and exit with status 0.

Upon failure, the script must specify the failure reason using the name-value pair `rcm_failure_reason` and exit with status 1.

**Logging** A script must log all error and debug messages by writing to stdout the name-value pairs listed below. The logged messages go to `syslogd(1M)` with the `syslog` facility of `LOG_DAEMON`. See `syslog.conf(4)`.

`rcm_log_err=message` Logs the *message* with the syslog level of `LOG_ERR`.

`rcm_log_warn=message` Logs the *message* with the syslog level of `LOG_WARNING`.

`rcm_log_info=message` Logs the *message* with the syslog level of `LOG_INFO`.

`rcm_log_debug=message` Logs the *message* with the syslog level of `LOG_DEBUG`.

A script can use the environment variable `RCM_ENV_DEBUG_LEVEL` to control the amount of information to log. `RCM_ENV_DEBUG_LEVEL` is a numeric value ranging from 0 to 9, with 0 meaning log the least amount of information and 9 meaning log the most.

**Installing or Removing RCM Scripts** You must use the following format to name a script:

*vendor,service*

where *vendor* is the stock symbol (or any distinctive name) of the vendor providing the script and *service* is the name of service the script represents.

You must be a superuser (root) to install or remove an RCM script.

Select one of the following directories where you want to place the script:

|                                                       |                                              |
|-------------------------------------------------------|----------------------------------------------|
| <code>/etc/rcm/scripts</code>                         | Scripts for specific systems                 |
| <code>/usr/platform/`uname -i`/lib/rcm/scripts</code> | Scripts for specific hardware implementation |
| <code>/usr/platform/`uname -m`/lib/rcm/scripts</code> | Scripts for specific hardware class          |
| <code>/usr/lib/rcm/scripts</code>                     | Scripts for any hardware                     |

### Installing a Script

To install a script, copy the script to the appropriate directory from the list above, change the userid and the groupid of the script to the desired values, and send SIGHUP to `rcm_daemon`. For example:

```
cp SUNW,sample.pl /usr/lib/rcm/scripts
chown user[:group] /usr/lib/rcm/scripts/SUNW,sample.pl
pkill -HUP -x -u root rcm_daemon
```

### Removing a script

Remove the script from the appropriate directory from the list above and send SIGHUP to `rcm_daemon`. For example:

```
rm /usr/lib/rcm/scripts/SUNW,sample.pl
pkill -HUP -x -u root rcm_daemon
```

## Examples

### EXAMPLE 1 Site Customization RCM Script

```
#!/usr/bin/perl -w

#
A sample site customization RCM script for a tape backup application.
#
This script registers all tape drives in the system with RCM.
When the system attempts to remove a tape drive by DR the script
does the following:
- if the tape drive is not being used for backup, it allows the
DR to continue.
- if the tape drive is being used for backup, and when DR is not
forced (RCM_ENV_FORCE=FALSE) it indicates that it cannot release
the tape drive with appropriate error message. When forced
(RCM_ENV_FORCE=TRUE) it kills the tape backup application in
order to allow the DR to continue.
#
This script does not implement the postremove and undoremove commands
since there is nothing to cleanup after DR remove operation is
completed or failed. If any cleanup is needed after the DR removal
completed, postremove command needs to implemented. If any cleanup is
```

**EXAMPLE 1** Site Customization RCM Script *(Continued)*

```
needed in the event of DR removal failure, undoremove command needs
to be implemented.
#

use strict;

my ($cmd, %dispatch);

$cmd = shift(@ARGV);

dispatch table for RCM commands
%dispatch = (
 "scriptinfo" => \&do_scriptinfo,
 "register" => \&do_register,
 "resourceinfo" => \&do_resourceinfo,
 "queryremove" => \&do_preremove,
 "preremove" => \&do_preremove
);

if (defined($dispatch{$cmd})) {
 &{$dispatch{$cmd}};
} else {
 exit (2);
}

sub do_scriptinfo
{
 print "rcm_script_version=1\n";
 print "rcm_script_func_info=Tape backup appl script for DR\n";
 exit (0);
}

sub do_register
{
 my ($dir, $f, $errmsg);

 $dir = opendir(RMT, "/dev/rmt");
 if (!$dir) {
 $errmsg = "Unable to open /dev/rmt directory: $!";
 print "rcm_failure_reason=$errmsg\n";
 exit (1);
 }

 while ($f = readdir(RMT)) {
 # ignore hidden files and multiple names for the same device
```



## EXAMPLE 1 Site Customization RCM Script (Continued)

```

 if (($f !~ /^\.\/) && ($f =~ /^[0-9]+$\/)) {
 print "rcm_resource_name=/dev/rmt/$f\n";
 }

}

closedir(RMT);
exit (0);
}

sub do_resourceinfo
{
 my ($rsrc, $unit);

 $rsrc = shift(@ARGV);
 if ($rsrc =~ /^\/dev\/rmt\/([0-9]+)\/) {
 $unit = $1;
 print "rcm_resource_usage_info=Backup Tape Unit Number $unit\n";
 exit (0);
 } else {
 print "rcm_failure_reason=Unknown tape device!\n";
 exit (1);
 }
}

sub do_preremove
{
 my ($rsrc);

 $rsrc = shift(@ARGV);

 # check if backup application is using this resource
 # if (the backup application is not running on $rsrc) {
 # allow the DR to continue
 # exit (0);
 #}
 #
 # If RCM_ENV_FORCE is FALSE deny the operation.
 # If RCM_ENV_FORCE is TRUE kill the backup application in order
 # to allow the DR operation to proceed
 #
 if ($ENV{RCM_ENV_FORCE} eq 'TRUE') {
 if ($cmd eq 'preremove') {
 # kill the tape backup application
 }
 }
}

```

**EXAMPLE 1** Site Customization RCM Script (Continued)

```

 exit (0);
 } else {
 #
 # indicate that the tape drive can not be released
 # since the device is being used for backup by the
 # tape backup application
 #
 print "rcm_failure_reason=tape backup in progress pid=...\n";
 exit (3);
 }
}

```

**Exit Status** A script must exit with following exit status values:

- 0 Operation specified by the given RCM command has been executed successfully by the script. For queryremove command it also means that the script can successfully release the resource.
- 1 An error occurred while processing the RCM command. The script should provide the error message to RCM using the name-value pair `rcm_failure_reason` before exiting.
- 2 The script does not support the given RCM command. A script must exit with this status if it cannot understand the given RCM command.
- 3 Indicates that the script cannot release the resource for preremove and queryremove commands. The script should provide a message to RCM specifying the reason for not being able to release the resource using the name-value pair `rcm_failure_reason` before exiting.

**Errors** If a script cannot successfully process an RCM command, it must supply to the RCM a message indicating the reason for failure by writing a name-value pair, in the form shown below, to stdout and exiting with the appropriate exit status.

```
rcm_failure_reason=failure_reason
```

where *failure\_reason* is a localized human readable message describing the reason for failure of the RCM command.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Committed       |

**See Also** `gettext(1)`, `cfgadm(1M)`, `cfgadm_scsi(1M)`, `cfgadm_pci(1M)`, `syslog(3C)`, `signal.h(3HEAD)`, `syslog.conf(4)`, `attributes(5)`, `environ(5)`

**Notes** RCM scripts are expected to properly handle all RCM commands that the script implements and to log all errors. Only root has permission to add or remove an RCM script. An ill-behaved RCM script can cause unexpected DR failures.

RCM commands are invoked only for the resources whose subsystems participate within the RCM framework. Currently, not all subsystems participate within the RCM framework.

**Name** rdc.cf – Availability Suite Remote Mirror software configuration file

**Description** The `rdc.cf` is an optional configuration file that supplies the `sndradm(1M)` command with details of the volume sets to be operated on. In `rdc.cf`, the volume sets and their host locations are defined in the following format:

```
post pdevice pbitmap shost sdevice sbitmap protocol mode options
```

The `rdc.cf` fields are:

`phost` (primary host)

Server on which the primary volume resides.

`pdevice` (primary device)

Primary volume partition to be copied. Specify only full path names (for example, `/dev/rdisk/c0t1d0s2`).

`pbitmap` (primary bitmap)

Volume partition in which the bitmap (scoreboard logs) of the primary partition is stored. Specify only full path names (for example, `/dev/rdisk/c0t1d0s3`).

`shost` (secondary host)

Server on which the secondary volume resides.

`sdevice` (secondary device)

Secondary volume partition. Specify only full path names (for example, `/dev/rdisk/c0t1d0s4`).

`sbitmap` (secondary bitmap)

Volume partition in which the bitmap (scoreboard logs) of the secondary file is stored. Specify only full path names (for example, `/dev/rdisk/c0t1d0s5`).

`protocol`

Network transfer protocol. Specify IP.

`mode`

Remote Mirror operating mode. Sync is the Remote Mirror mode where the I/O operation is not confirmed as complete until the remote volume has been updated. Async is the other Remote Mirror mode, in which the primary host I/O operation is confirmed as complete before updating the remote volume.

`options`

A consistency group name can be specified using the `g` character. A disk queue volume partition can be specified using the `q` character, using full path name only (`/dev/rdisk/c0t1d0s5`). Without the `q` character set will default to memory base queue. When running on a clustered system, a cluster resource group tag can be specified using the `C` character.

These options have the following syntax:

```
[g io_groupname] [q queue_volume][C ctag]
```

---

**Note** – When running on a cluster configuration, the cluster resource group tag is appended to the Remote Mirror set by default.

**Attributes** See [attributes\(5\)](#) for a description of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE               |
|---------------------|-------------------------------|
| Architecture        | x86                           |
| Availability        | storage/avs/avs-remote-mirror |
| Interface Stability | Committed                     |

**See Also** [sndradm\(1M\)](#), [sndrd\(1M\)](#), [attributes\(5\)](#)

**Name** registration\_profile – template for Basic Registration information

**Synopsis** registration.profile

**Description** The registration profile is a template for the information provided for Basic Registration. The information in this template is consumed by the [sconadm\(1M\)](#) utility. `sconadm` is the command-line alternative to the Basic Registration GUI.

An example of the registration profile template is provided in `/usr/lib/breg/data/RegistrationProfile.properties`. This file is owned by root, with read-only permissions. As root, you can copy the file to a location of your choosing and edit the file.

The registration profile template contains the following properties:

```
userName=
password=
hostName=
subscriptionKey=
portalEnabled=
proxyHostName=
proxyPort=
proxyUserName=
proxyPassword=
```

Values are not required for every property. The filling in or leaving blank a property depends on the task you intend to perform. Possible tasks are registering a new system, reregistering a system, and establishing a proxy. See [sconadm\(1M\)](#) for examples.

The properties defined for a registration template are as follows:

|                              |                                                                                                                                                                                                                                                                     |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>userName</code>        | Corresponds to Sun Online Account user name.                                                                                                                                                                                                                        |
| <code>password</code>        | Corresponds to Sun Online Account password.                                                                                                                                                                                                                         |
| <code>hostName</code>        | Hostname of the machine to be registered (defaults to Unix host name, if not provided).                                                                                                                                                                             |
| <code>subscriptionKey</code> | Enable access to all updates by providing your Sun Subscription Key for entitlement.<br><br><b>Note</b> – The Sun Subscription Key is now known as the Sun Service Plan Number. However, the name of this field ( <code>subscriptionKey</code> ) remains unchanged. |
| <code>portalEnabled</code>   | If <code>true</code> , enable local client to be managed at the Sun-hosted Update Connection Service. If <code>false</code> , disable local client to be managed at the Sun-hosted Update Connection Service.                                                       |
| <code>proxyHostName</code>   | Your HTTP web proxy host name.                                                                                                                                                                                                                                      |

proxyPort            Your HTTP web proxy port number.  
 proxyUserName        Your HTTP web proxy user name.  
 proxyPassword        Your HTTP web proxy password.

**Examples** EXAMPLE 1 Sample Registration Profile

Below are the contents of a sample registration profile. Such a profile might be used to specify a proxy server with authentication.

```

userName=
password=
hostName=
subscriptionKey=
portalEnabled=
proxyHostName=webcache.mycompany.com
proxyPort=8080
proxyUserName=myCompanyProxyUserName
proxyPassword=myCompanyProxyPassword

```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Availability        | SUNWbrg         |
| Interface Stability | Committed       |

**See Also** [sconadm\(1M\)](#), [attributes\(5\)](#)

**Name** remote – remote host description file

**Synopsis** /etc/remote

**Description** The systems known by `tip(1)` and their attributes are stored in an ASCII file which is structured somewhat like the `termcap` file. Each line in the file provides a description for a single *system*. Fields are separated by a colon ':'. Lines ending in a '\ ' character with an immediately following NEWLINE are continued on the next line.

The first entry is the name(s) of the host system. If there is more than one name for a system, the names are separated by vertical bars. After the name of the system comes the fields of the description. A field name followed by an '=' sign indicates a string value follows. A field name followed by a '#' sign indicates a following numeric value.

Entries named `tipbaudrate` are used as default entries by `tip`, as follows. When `tip` is invoked with only a phone number, it looks for an entry of the form `tipbaudrate`, where *baudrate* is the baud rate with which the connection is to be made. For example, if the connection is to be made at 300 baud, `tip` looks for an entry of the form `tip300`.

**Capabilities** Capabilities are either strings (`str`), numbers (`num`), or boolean flags (`bool`). A string capability is specified by `capability=value`; for example, 'dv=/dev/harris'. A numeric capability is specified by `capability#value`; for example, 'xa#99'. A boolean capability is specified by simply listing the capability.

`at` (`str`) Auto call unit type. The following lists valid 'at' types and their corresponding hardware:

`biz31f` Bizcomp 1031, tone dialing

`biz31w` Bizcomp 1031, pulse dialing

`biz22f` Bizcomp 1022, tone dialing

`biz22w` Bizcomp 1022, pulse dialing

`df02` DEC DF02

`df03` DEC DF03

`ventel` Ventel 212+

`v3451` Vadic 3451 Modem

`v831` Vadic 831

`hayes` Any Hayes-compatible modem

`at` Any Hayes-compatible modem

`br` (`num`) The baud rate used in establishing a connection to the remote host. This is a decimal number. The default baud rate is 300 baud.



- cm (str) An initial connection message to be sent to the remote host. For example, if a host is reached through a port selector, this might be set to the appropriate sequence required to switch to the host.
- cu (str) Call unit if making a phone call. Default is the same as the dv field.
- db (bool) Cause `tip(1)` to ignore the first hangup it sees. db (dialback) allows the user to remain in `tip` while the remote machine disconnects and places a call back to the local machine. For more information about dialback configuration, see *Oracle Solaris Administration: IP Services*.
- di (str) Disconnect message sent to the host when a disconnect is requested by the user.
- du (bool) This host is on a dial-up line.
- dv (str) Device(s) to open to establish a connection. If this file refers to a terminal line, `tip` attempts to perform an exclusive open on the device to insure only one user at a time has access to the port.
- ec (bool) Initialize the `tip` variable `echocheck` to `on`, so that `tip` will synchronize with the remote host during file transfer by waiting for the echo of the last character transmitted.
- e1 (str) Characters marking an end-of-line. The default is no characters. `tip` only recognizes `~` escapes after one of the characters in `e1`, or after a RETURN.
- es (str) The command prefix (escape) character for `tip`.
- et (num) Number of seconds to wait for an echo response when echo-check mode is on. This is a decimal number. The default value is 10 seconds.
- ex (str) Set of non-printable characters not to be discarded when scripting with beautification turned on. The default value is `"\t\n\b\f"`.
- fo (str) Character used to force literal data transmission. The default value is `'\377'`.
- fs (num) Frame size for transfers. The default frame size is equal to 1024.
- hd (bool) Initialize the `tip` variable `halfduplex` to `on`, so local echo should be performed.
- hf (bool) Initialize the `tip` variable `hardwareflow` to `on`, so hardware flow control is used.
- ie (str) Input end-of-file marks. The default is a null string (`""`).
- nb (bool) Initialize the `tip` variable `beautify` to `off`, so that unprintable characters will not be discarded when scripting.
- nt (bool) Initialize the `tip` variable `tandem` to `off`, so that XON/XOFF flow control will not be used to throttle data from the remote host.
- nv (bool) Initialize the `tip` variable `verbose` to `off`, so that verbose mode will be turned on.

- oe** (str) Output end-of-file string. The default is a null string (""). When `tip` is transferring a file, this string is sent at end-of-file.
- pa** (str) The type of parity to use when sending data to the host. This may be one of even, odd, none, zero (always set bit 8 to 0), one (always set bit 8 to 1). The default is none.
- pn** (str) Telephone number(s) for this host. If the telephone number field contains an '@' sign, `tip` searches the `/etc/phones` file for a list of telephone numbers — see [phones\(4\)](#). A '%' sign in the telephone number indicates a 5-second delay for the Ventel Modem.

For Hayes-compatible modems, if the telephone number starts with an 'S', the telephone number string will be sent to the modem without the "DT", which allows reconfiguration of the modem's S-registers and other parameters; for example, to disable auto-answer: "pn=S0=0DT5551234"; or to also restrict the modem to return only the basic result codes: "pn=S0=0X0DT5551234".

- pr** (str) Character that indicates end-of-line on the remote host. The default value is '\n'.
- ra** (bool) Initialize the `tip` variable `raise` to on, so that lower case letters are mapped to upper case before sending them to the remote host.
- rc** (str) Character that toggles case-mapping mode. The default value is '\377'.
- re** (str) The file in which to record session scripts. The default value is `tip.record`.
- rw** (bool) Initialize the `tip` variable `rawftp` to on, so that all characters will be sent as is during file transfers.
- sc** (bool) Initialize the `tip` variable `script` to on, so that everything transmitted by the remote host will be recorded.
- tb** (bool) Initialize the `tip` variable `tabexpand` to on, so that tabs will be expanded to spaces during file transfers.
- tc** (str) Indicates that the list of capabilities is continued in the named description. This is used primarily to share common capability information.

### Examples **EXAMPLE 1** Using the Capability Continuation Feature

Here is a short example showing the use of the capability continuation feature:

```
UNIX-1200:\
:dv=/dev/cua0:el=^D^U^C^S^Q^O@:du:at=ventel:ie=#$%:oe=^D
:br#1200:arpavax|ax:\
:pn=7654321%:tc=UNIX-1200
```

- Files** `/etc/remote` remote host description file.  
`/etc/phones` remote host phone number database.

**See Also** [tip\(1\)](#), [phones\(4\)](#)

*Oracle Solaris Administration: IP Services*

**Name** resolv.conf – resolver configuration file

**Synopsis** svc:/network/dns/client  
/etc/resolv.conf

**Description** The resolver is a set of routines that provide access to the Internet Domain Name System. See [resolver\(3RESOLV\)](#).

resolv.conf is a configuration file that contains the information that is read by the resolver routines the first time they are invoked by a process. The file is designed to be human readable and contains a list of keywords with values that provide various types of resolver information.

The configuration used by the resolver is managed by SMF in the svc:/network/dns/client service.

For the purposes of backwards compatibility, the resolv.conf file is regenerated from the SMF properties.

The following properties are supported and contained in the config property group of the svc:/network/dns/client service.

The following properties are single-valued or multi-valued:

nameserver

Specifies the IPv4 or IPv6 Internet address of a name server that the resolver is to query. Up to *MAXNS* name servers may be listed, one per keyword. See <resolv.h>. If there are multiple servers, the resolver library queries them in the order listed. If no name server entries are present, the resolver library queries the name server on the local machine. The resolver library follows the algorithm to try a name server until the query times out. It then tries the name servers that follow, until each query times out. It repeats all the name servers until a maximum number of retries are made.

domain

Specifies the local domain name. Most queries for names within this domain can use short names relative to the local domain. If no domain entry is present, the domain is determined from [sysinfo\(2\)](#) or from [gethostname\(3C\)](#). (Everything after the first '.' is presumed to be the domain name.) If the host name does not contain a domain part, the root domain is assumed. You can use the LOCALDOMAIN environment variable to override the domain name.

search

The search list for host name lookup. The search list is normally determined from the local domain name. By default, it contains only the local domain name. You can change the default behavior by listing the desired domain search path following the search keyword, with spaces or tabs separating the names. Most resolver queries will be attempted using each component of the search path in turn until a match is found. This process may be slow

and will generate a lot of network traffic if the servers for the listed domains are not local. Queries will time out if no server is available for one of the domains.

The search list is currently limited to six domains and a total of 256 characters.

#### sortlist

Allows addresses returned by the `libresolv-internal gethostbyname()` to be sorted. A `sortlist` is specified by IP address netmask pairs. The netmask is optional and defaults to the natural netmask of the net. The IP address and optional network pairs are separated by slashes. Up to 10 pairs may be specified. For example:

```
sortlist 130.155.160.0/255.255.240.0 130.155.0.0
```

#### options

Allows certain internal resolver variables to be modified. The syntax is

```
options option ...
```

where `option` is one of the following:

##### debug

Sets `RES_DEBUG` in the `_res.options` field.

##### ndots:n

Sets a threshold floor for the number of dots which must appear in a name given to `res_query()` before an initial absolute (as-is) query is performed. See [resolver\(3RESOLV\)](#). The default value for `n` is 1, which means that if there are any dots in a name, the name is tried first as an absolute name before any search list elements are appended to it.

##### timeout:n

##### retrans:n

Sets the amount of time the resolver will wait for a response from a remote name server before retrying the query by means of a different name server. Measured in seconds, the default is `RES_TIMEOUT`. See `<resolv.h>`. The `timeout` and `retrans` values are the starting point for an exponential back off procedure where the `timeout` is doubled for every retransmit attempt.

##### attempts:n

##### retry:n

Sets the number of times the resolver will send a query to its name servers before giving up and returning an error to the calling application. The default is `RES_DFLRETRY`. See `<resolv.h>`.

##### rotate

Sets `RES_ROTATE` in `_res.options`. The name servers are queried round-robin from among those listed. The query load is spread among all listed servers, rather than having all clients try the first listed server first every time.

**no-check-names**

Sets `RES_NOCHECKNAME` in `_res.options`. This disables the modern BIND checking of incoming host names and mail names for invalid characters such as underscore (`_`), non-ASCII, or control characters.

**inet6**

Sets `RES_USE_INET6` in `_res.options`. In the Solaris BIND port, this has no effect on `gethostbyname(3NSL)`. To retrieve IPv6 addresses or IPv4 addresses, use `getaddrinfo(3SOCKET)` instead of setting `inet6`.

The `domain` and `search` keywords are mutually exclusive. If more than one instance of these keywords is present, the last instance takes precedence.

You can override the `search` keyword of the system `resolv.conf` file on a per-process basis by setting the environment variable `LOCALDOMAIN` to a space-separated list of search domains.

You can amend the `options` keyword of the system `resolv.conf` file on a per-process basis by setting the environment variable `RES_OPTIONS` to a space-separated list of resolver options.

The keyword and value must appear on a single line. Start the line with the keyword, for example, `nameserver`, followed by the value, separated by white space.

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

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Committed       |
| Standard            | BIND 8.3.3      |

**See Also** `domainname(1M)`, `sysinfo(2)`, `gethostbyname(3NSL)`, `getnameinfo(3SOCKET)`, `getipnodebyname(3SOCKET)`, `gethostbyname(3C)`, `resolver(3RESOLV)`, `attributes(5)`

Vixie, Paul, Dunlap, Keven J., Karels, Michael J. *Name Server Operations Guide for BIND*. Internet Software Consortium, 1996.

**Name** rmtab – remote mounted file system table

**Synopsis** /etc/rmtab

**Description** rmtab contains a table of filesystems that are remotely mounted by NFS clients. This file is maintained by [mountd\(1M\)](#), the mount daemon. The data in this file should be obtained only from [mountd\(1M\)](#) using the MOUNTPROC\_DUMP remote procedure call.

The file contains a line of information for each remotely mounted filesystem. There are a number of lines of the form:

```
hostname:fsname
```

The mount daemon adds an entry for any client that successfully executes a mount request and deletes the appropriate entries for an unmount request.

Lines beginning with a hash ('#') are commented out. These lines are removed from the file by [mountd\(1M\)](#) when it first starts up. Stale entries may accumulate for clients that crash without sending an unmount request.

**Files** /etc/rmtab

**See Also** [mountd\(1M\)](#), [showmount\(1M\)](#)

**Name** rndc.conf – rndc configuration file

**Synopsis** rndc.conf

**Description** rndc.conf is the configuration file for rndc, the BIND 9 name server control utility. This file has a similar structure and syntax to named.conf. Statements are enclosed in braces and terminated with a semicolon. Clauses in the statements are also semicolon terminated. The usual comment styles are supported:

C style        /\* \*/

C++ style     // to end of line

Unix style    # to end of line

rndc.conf is much simpler than named.conf. The file uses three statements: an options statement, a server statement and a key statement.

The options statement contains five clauses. The default-server clause is followed by the name or address of a name server. This host is used when no name server is provided as an argument to rndc. The default-key clause is followed by the name of a key which is identified by a key statement. If no keyid is provided on the rndc command line, and no key clause is found in a matching server statement, this default key will be used to authenticate the server's commands and responses. The default-port clause is followed by the port to connect to on the remote name server. If no port option is provided on the rndc command line, and no port clause is found in a matching server statement, this default port will be used to connect. The default-source-address and default-source-address-v6 clauses which can be used to set the IPv4 and IPv6 source addresses respectively.

After the server keyword, the server statement includes a string which is the hostname or address for a name server. The statement has three possible clauses: key, port, and addresses. The key name must match the name of a key statement in the file. The port number specifies the port to connect to. If an addresses clause is supplied these addresses will be used instead of the server name. Each address can take an optional port. If a source-address or source-address-v6 is supplied then these will be used to specify the IPv4 and IPv6 source addresses respectively.

The key statement begins with an identifying string, the name of the key. The statement has two clauses. algorithm identifies the encryption algorithm for rndc to use; currently only HMAC-MD5 is supported. This is followed by a secret clause which contains the base-64 encoding of the algorithm's encryption key. The base-64 string is enclosed in double quotes.

There are two common ways to generate the base-64 string for the secret. The BIND 9 program `rndc-confgen\(1M\)` can be used to generate a random key, or the `mmencode` program, also known as `mimencode`, can be used to generate a base-64 string from known input. `mmencode` does not ship with BIND 9 but is available on many systems. See the `EXAMPLES` section for sample command lines for each.



```

Examples options {
 default-server localhost;
 default-key samplekey;
};

server localhost {
 key samplekey;
};

server testserver {
 key testkey;
 addresses { localhost port 5353; };
};

key samplekey {
 algorithm hmac-md5;
 secret "6FMfj430sz4lyb240Ie2iGEz9lf1lLJO+lz";
};

key testkey {
 algorithm hmac-md5;
 secret "R3HI8P6BKw9ZwXwN3VZKuQ==";
};

```

In the above example, `rndc` by default uses the server at `localhost` (127.0.0.1) and the key called `samplekey`. Commands to the `localhost` server will use the `samplekey` key, which must also be defined in the server's configuration file with the same name and secret. The key statement indicates that `samplekey` uses the HMAC-MD5 algorithm and its secret clause contains the base-64 encoding of the HMAC-MD5 secret enclosed in double quotes.

If `rndc -s testserver` is used then `rndc` connects to server on `localhost` port 5353 using the key `testkey`.

To generate a random secret with `rndc-confgen`:

```
rndc-confgen
```

A complete `rndc.conf` file, including the randomly generated key, will be written to the standard output. Commented out key and controls statements for `named.conf` are also printed.

To generate a base-64 secret with `mmencode`:

```
echo "known plaintext for a secret" | mmencode
```

**Name Server Configuration** The name server must be configured to accept `rndc` connections and to recognize the key specified in the `rndc.conf` file, using the controls statement in `named.conf`. See the sections on the controls statement in the *BIND 9 Administrator Reference Manual* for details.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE          |
|---------------------|--------------------------|
| Availability        | service/network/dns/bind |
| Interface Stability | Volatile                 |

**See Also** [rnc\(1M\)](#), [rnc - confgen\(1M\)](#), [attributes\(5\)](#)

*BIND 9 Administrator Reference Manual*

**Name** rpc – rpc program number data base

**Synopsis** /etc/rpc

**Description** The rpc file is a local source containing user readable names that can be used in place of RPC program numbers. The rpc file can be used in conjunction with or instead of other rpc sources, including the NIS maps “rpc.byname” and “rpc.bynumber”.

The rpc file has one line for each RPC program name. The line has the following format:

*name-of-the-RPC-program* *RPC-program-number* *aliases*

Items are separated by any number of blanks and/or tab characters. A “#” indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

**Examples** EXAMPLE 1 RPC Database

Below is an example of an RPC database:

```
#
rpc
#
rpcbind 100000 portmap sunrpc portmapper
rusersd 100002 rusers
nfs 100003 nfsprog
mountd 100005 mount showmount
walld 100008 rwall shutdown
sprayd 100012 spray
llockmgr 100020
nlockmgr 100021
status 100024
bootparam 100026
keyserv 100029 keyserver
```

**Files** /etc/nsswitch.conf

**See Also** [nsswitch.conf\(4\)](#)

**Name** rt\_dptbl – real-time dispatcher parameter table

**Description** The process scheduler (or dispatcher) is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the notion of scheduling classes where each class defines a scheduling policy, used to schedule processes within that class. Associated with each scheduling class is a set of priority queues on which ready to run processes are linked. These priority queues are mapped by the system configuration into a set of global scheduling priorities which are available to processes within the class. The dispatcher always selects for execution the process with the highest global scheduling priority in the system. The priority queues associated with a given class are viewed by that class as a contiguous set of priority levels numbered from 0 (lowest priority) to  $n$  (highest priority—a configuration dependent value). The set of global scheduling priorities that the queues for a given class are mapped into might not start at zero and might not be contiguous, depending on the configuration.

The real-time class maintains an in-core table, with an entry for each priority level, giving the properties of that level. This table is called the real-time dispatcher parameter table (`rt_dptbl`). The `rt_dptbl` consists of an array (`config_rt_dptbl[]`) of parameter structures (`struct rtdpent_t`), one for each of the  $n$  priority levels. The structure are accessed via a pointer, (`rt_dptbl`), to the array. The properties of a given priority level  $i$  are specified by the  $i$ th parameter structure in this array (`rt_dptbl[i]`).

A parameter structure consists of the following members. These are also described in the `/usr/include/sys/rt.h` header file.

|                         |                                                                                                                                                                                                                                                                                                                                        |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>rt_globpri</code> | The global scheduling priority associated with this priority level. The <code>rt_globpri</code> values cannot be changed with <code>dispadmin(1M)</code> .                                                                                                                                                                             |
| <code>rt_quantum</code> | The length of the time quantum allocated to processes at this level in ticks (hz). The time quantum value is only a default or starting value for processes at a particular level as the time quantum of a real-time process can be changed by the user with the <code>pricntl</code> command or the <code>pricntl</code> system call. |

In the high resolution clock mode (`hires_tick` set to 1), the value of hz is set to 1000. Increase quantum to maintain the same absolute time quantum.

An administrator can affect the behavior of the real-time portion of the scheduler by reconfiguring the `rt_dptbl`. There are two methods available for doing this: reconfigure with a loadable module at boot-time or by using `dispadmin(1M)` at run-time.

|                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>rt_dptbl</code> Loadable Module | The <code>rt_dptbl</code> can be reconfigured with a loadable module which contains a new real time dispatch table. The module containing the dispatch table is separate from the RT loadable module which contains the rest of the real time software. This is the only method that can be used to change the number of real time priority levels or the set of global scheduling priorities used by the real time class. The relevant procedure and source code is described in the <a href="#">Examples</a> section. |
|---------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

dispadmin  
Configuration File

The `rt_quantum` values in the `rt_dptbl` can be examined and modified on a running system using the `dispadmin(1M)` command. Invoking `dispadmin` for the real-time class allows the administrator to retrieve the current `rt_dptbl` configuration from the kernel's in-core table, or overwrite the in-core table with values from a configuration file. The configuration file used for input to `dispadmin` must conform to the specific format described below.

Blank lines are ignored and any part of a line to the right of a `#` symbol is treated as a comment. The first non-blank, non-comment line must indicate the resolution to be used for interpreting the time quantum values. The resolution is specified as

```
RES=res
```

where *res* is a positive integer between 1 and 1,000,000,000 inclusive and the resolution used is the reciprocal of *res* in seconds. (For example, `RES=1000` specifies millisecond resolution.) Although very fine (nanosecond) resolution may be specified, the time quantum lengths are rounded up to the next integral multiple of the system clock's resolution.

The remaining lines in the file are used to specify the `rt_quantum` values for each of the real-time priority levels. The first line specifies the quantum for real-time level 0, the second line specifies the quantum for real-time level 1. There must be exactly one line for each configured real-time priority level. Each `rt_quantum` entry must be either a positive integer specifying the desired time quantum (in the resolution given by *res*), or the value -2 indicating an infinite time quantum for that level.

### Examples EXAMPLE 1 A Sample dispadmin Configuration File

The following excerpt from a `dispadmin` configuration file illustrates the format. Note that for each line specifying a time quantum there is a comment indicating the corresponding priority level. These level numbers indicate priority within the real-time class, and the mapping between these real-time priorities and the corresponding global scheduling priorities is determined by the configuration specified in the `RT_DPTBL` loadable module. The level numbers are strictly for the convenience of the administrator reading the file and, as with any comment, they are ignored by `dispadmin` on input. `dispadmin` assumes that the lines in the file are ordered by consecutive, increasing priority level (from 0 to the maximum configured real-time priority). The level numbers in the comments should normally agree with this ordering; if for some reason they don't, however, `dispadmin` is unaffected.

```
Real-Time Dispatcher Configuration File
RES=1000

TIME QUANTUM PRIORITY
(rt_quantum)LEVEL
100# 0
100# 1
100# 2
100# 3
100# 4
```

**EXAMPLE 1** A Sample `dispadm` Configuration File *(Continued)*

```

100# 5
90 # 6
90 # 7
.. .
.. .
.. .
10# 58
10# 59

```

**EXAMPLE 2** Replacing The `rt_dptbl` Loadable Module

In order to change the size of the real time dispatch table, the loadable module which contains the dispatch table information will have to be built. It is recommended that you save the existing module before using the following procedure.

1. Place the dispatch table code shown below in a file called `rt_dptbl.c`. An example of an `rt_dptbl.c` file follows.
2. Compile the code using the given compilation and link lines supplied.

```

cc -c -O -D_KERNEL rt_dptbl.c
ld -r -o RT_DPTBL rt_dptbl.o

```

3. Copy the current dispatch table in `/usr/kernel/sched` to `RT_DPTBL.bak`.
4. Replace the current `RT_DPTBL` in `/usr/kernel/sched`.
5. You will have to make changes in the `/etc/system` file to reflect the changes to the sizes of the tables. See [system\(4\)](#). The `rt_maxpri` variable may need changing. The syntax for setting this is:

```

set RT:rt_maxpri=(class-specific value for maximum \
 real-time priority)

```

6. Reboot the system to use the new dispatch table.

Great care should be used in replacing the dispatch table using this method. If you don't get it right, the system may not behave properly.

The following is an example of a `rt_dptbl.c` file used for building the new `rt_dptbl`.

```

/* BEGIN rt_dptbl.c */
#include <sys/proc.h>
#include <sys/priocntl.h>
#include <sys/class.h>
#include <sys/disp.h>
#include <sys/rt.h>
#include <sys/rtpriocntl.h>
/*

```

## EXAMPLE 2 Replacing The rt\_dptbl Loadable Module (Continued)

```

 * This is the loadable module wrapper.
 */
#include <sys/modctl.h>
extern struct mod_ops mod_miscops;
/*
 * Module linkage information for the kernel.
 */
static struct modlmisc modlmisc = {
 &mod_miscops, "realtime dispatch table"
};
static struct modlinkage modlinkage = {
 MODREV_1, &modlmisc, 0
};
__init()
{
 return (mod_install(&modlinkage));
}
__info (struct modinfo *modinfop)
{
 return (mod_info(&modlinkage, modinfop));
}
rtdepent_t config_rt_dptbl[] = {

/* prilevel Time quantum */

100,100,
101,100,
102,100,
103,100,
104,100,
105,100,
106,100,
107,100,
108,100,
109,100,
110,80,
111,80,
112,80,
113,80,
114,80,
115,80,
116,80,
117,80,
118,80,
119,80,

```

EXAMPLE 2 Replacing The rt\_dptbl Loadable Module *(Continued)*

```
120,60,
121,60,
122,60,
123,60,
124,60,
125,60,
126,60,
127,60,
128,60,
129,60,
130,40,
131,40,
132,40,
133,40,
134,40,
135,40,
136,40,
137,40,
138,40,
139,40,
140,20,
141,20,
142,20,
143,20,
144,20,
145,20,
146,20,
147,20,
148,20,
149,20,
150,10,
151,10,
152,10,
153,10,
154,10,
155,10,
156,10,
157,10,
158,10,
159,10,

};
/*
 * Return the address of config_rt_dptbl
 */ rtdpent_t *
```



**EXAMPLE 2** Replacing The rt\_dptbl Loadable Module (Continued)

```
 rt_getdptbl()
{
 return (config_rt_dptbl);
}
```

**See Also** [priocntl\(1\)](#), [dispadmin\(1M\)](#), [priocntl\(2\)](#), [system\(4\)](#)

*Oracle Solaris Administration: Common Tasks*

*Programming Interfaces Guide*

**Name** sasl\_appname.conf – SASL options and configuration file

**Synopsis** /etc/sasl/appname.conf

**Description** The /etc/sasl/appname.conf file is a user-supplied configuration file that supports user set options for server applications.

You can modify the behavior of `libsasl` and its plug-ins for server applications by specifying option values in /etc/sasl/appname.conf file, where *appname* is the application defined name of the application. For `sendmail`, the file would be /etc/sasl/Sendmail.conf. See your application documentation for information on the application name.

Options that you set in a *appname.conf* file do not override SASL options specified by the application itself.

The format for each option setting is:

```
option_name:value.
```

You can comment lines in the file by using a leading #.

The SASL library supports the following options for server applications:

|                                |                                                                                                                                                                                                                                                                                                                             |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>auto_transition</code>   | When set to yes, plain users and login plug-ins are automatically transitioned to other mechanisms when they do a successful plaintext authentication. The default value for <code>auto_transition</code> is no.                                                                                                            |
| <code>auxprop_plugin</code>    | A space-separated list of names of auxiliary property plug-ins to use. By default, SASL will use or query all available auxiliary property plug-ins.                                                                                                                                                                        |
| <code>canon_user_plugin</code> | The name of the canonical user plug-in to use. By default, the value of <code>canon_user_plugin</code> is INTERNAL, to indicated the use of built-in plug-ins..                                                                                                                                                             |
| <code>log_level</code>         | An integer value for the desired level of logging for a server, as defined in <code>&lt;sasl.h&gt;</code> . This sets the <code>log_level</code> in the <code>sasl_server_params_t</code> struct in <code>/usr/include/sasl/saslplug.h</code> . The default value for <code>log_level</code> is 1 to indicate SASL_LOG_ERR. |
| <code>mech_list</code>         | Whitespace separated list of SASL mechanisms to allow, for example, DIGEST-MD5 GSSAPI. The <code>mech_list</code> option is used to restrict the mechanisms to a subset of the installed plug-ins. By default, SASL will use all available mechanisms.                                                                      |

|                                    |                                                                                                                                                                                                                                                                                                                                   |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pw_check</code>              | Whitespace separated list of mechanisms used to verify passwords that are used by <code>sasl_checkpass(3SASL)</code> . The default value for <code>pw_check</code> is <code>auxprop</code> .                                                                                                                                      |
| <code>reauth_timeout</code>        | This SASL option is used by the server DIGEST-MD5 plug-in. The value of <code>reauth_timeout</code> is the length in time (in minutes) that authentication information will be cached for a fast reauthorization. A value of 0 will disable reauthorization. The default value of <code>reauth_timeout</code> is 1440 (24 hours). |
| <code>server_load_mech_list</code> | A space separated list of mechanisms to load. If in the process of loading server plug-ins no desired mechanisms are included in the plug-in, the plug-in will be unloaded. By default, SASL loads all server plug-ins.                                                                                                           |
| <code>user_authid</code>           | If the value of <code>user_authid</code> is yes, then the GSSAPI will acquire the client credentials rather than use the default credentials when it creates the GSS client security context. The default value of <code>user_authid</code> is no, whereby SASL uses the default client Kerberos identity.                        |

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTETYPE       | ATTRIBUTEVALUE |
|---------------------|----------------|
| Interface Stability | Committed      |

**See Also** [attributes\(5\)](#)

**Name** sbus – configuration files for SBus device drivers

**Description** The SBus is a geographically addressed peripheral bus present on many SPARC hardware platforms. SBus devices are *self-identifying* — that is to say the SBus card itself provides information to the system so that it can identify the device driver that needs to be used. The device usually provides additional information to the system in the form of name-value pairs that can be retrieved using the DDI property interfaces. See [ddi\\_prop\\_op\(9F\)](#) for details.

The information is usually derived from a small Forth program stored in the FCode PROM on the card, so driver configuration files should be completely unnecessary for these devices. However, on some occasions, drivers for SBus devices may need to use driver configuration files to augment the information provided by the SBus card. See [driver.conf\(4\)](#) for further details.

When they are needed, configuration files for SBus device drivers should identify the parent bus driver implicitly using the *class* keyword. This removes the dependency on the particular bus driver involved since this may be named differently on different platforms.

All bus drivers of class sbus recognise the following properties:

- reg** An arbitrary length array where each element of the array consists of a 3-tuple of integers. Each array element describes a logically contiguous mappable resource on the SBus.
- The first integer of each tuple specifies the slot number the card is plugged into. The second integer of each 3-tuple specifies the offset in the slot address space identified by the first element. The third integer of each 3-tuple specifies the size in bytes of the mappable resource.
- The driver can refer to the elements of this array by index, and construct kernel mappings to these addresses using [ddi\\_map\\_regs\(9F\)](#). The index into the array is passed as the *rnumber* argument of [ddi\\_map\\_regs\(\)](#).
- You can use the `ddi_get*` and `ddi_put*` family of functions to access register space from a high-level interrupt context.
- interrupts** An arbitrary length array where each element of the array consists of a single integer. Each array element describes a possible SBus interrupt level that the device might generate.
- The driver can refer to the elements of this array by index, and register interrupt handlers with the system using [ddi\\_add\\_intr\(9F\)](#). The index into the array is passed as the *inumber* argument of [ddi\\_add\\_intr\(\)](#).
- registers** An arbitrary length array where each element of the array consists of a 3-tuple of integers. Each array element describes a logically contiguous mappable resource on the SBus.

The first integer of each tuple should be set to `-1`, specifying that any SBus slot may be matched. The second integer of each 3-tuple specifies the offset in the slot address space identified by the first element. The third integer of each 3-tuple specifies the size in bytes of the mappable resource.

The `registers` property can only be used to augment an incompletely specified `reg` property with information from a driver configuration file. It may only be specified in a driver configuration file.

All SBus devices must provide `reg` properties to the system. The first two integer elements of the `reg` property are used to construct the address part of the device name under `/devices`.

Only devices that generate interrupts need to provide `interrupts` properties.

Occasionally, it may be necessary to override or augment the configuration information supplied by the SBus device. This can be achieved by writing a driver configuration file that describes a prototype device information (`devinfo`) node specification, containing the additional properties required.

For the system to merge the information, certain conditions must be met. First, the `name` property must be the same. Second, either the first two integers (slot number and offset) of the two `reg` properties must be the same, or the second integer (offset) of the `reg` and `registers` properties must be the same.

In the event that the SBus card has no `reg` property at all, the self-identifying information cannot be used, so all the details of the card must be specified in a driver configuration file.

#### Examples **EXAMPLE 1** A sample configuration file.

Here is a configuration file for an SBus card called `SUNW,netboard`. The card already has a simple FCode PROM that creates `name` and `reg` properties, and will have a complete set of properties for normal use once the driver and firmware is complete.

In this example, we want to augment the properties given to us by the firmware. We use the same `name` property, and use the `registers` property to match the firmware `reg` property. That way we don't have to worry about which slot the card is really plugged into.

We want to add an `interrupts` property while we are developing the firmware and driver so that we can start to experiment with interrupts. The device can generate interrupts at SBus level 3. Additionally, we want to set a `debug-level` property to 4.

```
#
Copyright (c) 1992, by Sun Microsystems, Inc.
#ident "@(#)SUNW,netboard.conf 1.4 92/03/10 SMI"
#
name="SUNW,netboard" class="sbus"
 registers=-1,0x40000,64,-1,0x80000,1024
```

EXAMPLE 1 A sample configuration file. (Continued)

```
interrupts=3 debug-level=4;
```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Architecture   | SPARC           |

**See Also** [driver.conf\(4\)](#), [attributes\(5\)](#), [ddi\\_add\\_intr\(9F\)](#), [ddi\\_map\\_regs\(9F\)](#), [ddi\\_prop\\_op\(9F\)](#)

*Writing Device Drivers*

**Warnings** The wildcarding mechanism of the `registers` property matches every instance of the particular device attached to the system. This may not always be what is wanted.

**Name** sccsfile – format of an SCCS history file

**Description** An SCCS file is an ASCII file consisting of six logical parts:

|                    |                                                              |
|--------------------|--------------------------------------------------------------|
| <i>checksum</i>    | Character count used for error detection.                    |
| <i>delta table</i> | Log containing version info and statistics about each delta. |
| <i>usernames</i>   | Login names and/or group IDs of users who may add deltas.    |
| <i>flags</i>       | Definitions of internal keywords.                            |
| <i>comments</i>    | Arbitrary descriptive information about the file.            |
| <i>body</i>        | the Actual text lines intermixed with control lines.         |

Each section is described in detail below.

**Conventions** Throughout an SCCS file there are lines which begin with the ASCII SOH (start of heading) character (octal 001). This character is hereafter referred to as the *control character*, and will be represented as '^A'. If a line described below is not depicted as beginning with the control character, it cannot do so and still be within SCCS file format.

Entries of the form *dddd* represent a five digit string (a number between 00000 and 99999).

**Checksum** The checksum is the first line of an SCCS file. The form of the line is:

^A *hdddd*

The value of the checksum is the sum of all characters, except those contained in the first line. The ^Ah provides a *magic number* of (octal) 064001.

**Delta Table** The delta table consists of a variable number of entries of the form:

```
^As inserted /deleted/unchanged
^Ad type sid yr/mo/da hr:mi:se username serial-number \
 predecessor-sn
^Ai include-list
^Ax exclude-list
^Ag ignored-list
^Am mr-number
...
^Ac comments ...
...
^Ae
```

The first line (^As) contains the number of lines inserted/deleted/unchanged respectively. The second line (^Ad) contains the type of the delta (normal: D and removed: R), the SCCS ID of the delta, the date and time of creation of the delta, the user-name corresponding to the real user ID at the time the delta was created, and the serial numbers of the delta and its predecessor,

respectively. The `^Ai`, `^Ax`, and `^Ag` lines contain the serial numbers of deltas included, excluded, and ignored, respectively. These lines do not always appear.

The `^Am` lines (optional) each contain one MR number associated with the delta. The `^Ac` lines contain comments associated with the delta.

The `^Ae` line ends the delta table entry.

**User Names** The list of user-names and/or numerical group IDs of users who may add deltas to the file, separated by NEWLINE characters. The lines containing these login names and/or numerical group IDs are surrounded by the bracketing lines `^Au` and `^AU`. An empty list allows anyone to make a delta.

**Flags** Flags are keywords that are used internally (see [scs-admin\(1\)](#) for more information on their use). Each flag line takes the form:

```
^Af flag
 optional text
```

The following flags are defined in order of appearance:

- `^Af t type-of-program` Defines the replacement for the `11:10:32` ID keyword.
- `^Af v program-name` Controls prompting for MR numbers in addition to comments. If the optional text is present, it defines an MR number validity checking program.
- `^Af i` Indicates that the 'No id keywords' message is to generate an error that terminates the SCCS command. Otherwise, the message is treated as a warning only.
- `^Af b` Indicates that the `-b` option may be used with the SCCS `get` command to create a branch in the delta tree.
- `^Af m module-name` Defines the first choice for the replacement text of the `scsfile.4` ID keyword.
- `^Af f floor` Defines the "floor" release, that is, the release below which no deltas may be added.
- `^Af c ceiling` Defines the "ceiling" release, that is, the release above which no deltas may be added.
- `^Af d default-sid` The `d` flag defines the default SID to be used when none is specified on an SCCS `get` command.
- `^Af n` The `n` flag enables the SCCS `delta` command to insert a "null" delta (a delta that applies *no* changes) in those releases that are skipped when a delta is made in a *new* release (for example, when delta 5.1 is made after delta 2.7, releases 3 and 4 are skipped).



---

|                                         |                                                                                                                                                                                                                                                                                                                               |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>^Af j</code>                      | Enables the SCCS <code>get</code> command to allow concurrent edits of the same base SID.                                                                                                                                                                                                                                     |
| <code>^Af l</code> <i>lock-releases</i> | Defines a list of releases that are locked against editing.                                                                                                                                                                                                                                                                   |
| <code>^Af q</code> <i>user-defined</i>  | Defines the replacement for the ID keyword.                                                                                                                                                                                                                                                                                   |
| <code>^Af e 0 1</code>                  | The <code>e</code> flag indicates whether a source file is encoded or not. A <code>1</code> indicates that the file is encoded. Source files need to be encoded when they contain control characters, or when they do not end with a NEWLINE. The <code>e</code> flag allows files that contain binary data to be checked in. |

**Comments** Arbitrary text surrounded by the bracketing lines `^At` and `^AT`. The comments section typically will contain a description of the file's purpose.

**Body** The body consists of text lines and control lines. Text lines do not begin with the control character, control lines do. There are three kinds of control lines: *insert*, *delete*, and *end*, represented by:

```
^AI dddd
^AD dddd
^AE dddd
```

respectively. The digit string is the serial number corresponding to the delta for the control line.

**See Also** [sccs-admin\(1\)](#), [sccs-cdc\(1\)](#), [sccs-comb\(1\)](#), [sccs-delta\(1\)](#), [sccs-get\(1\)](#), [sccs-help\(1\)](#), [sccs-prs\(1\)](#), [sccs-prt\(1\)](#), [sccs-rmdel\(1\)](#), [sccs-sact\(1\)](#), [sccs-sccsdiff\(1\)](#), [sccs-unget\(1\)](#), [sccs-val\(1\)](#), [sccs\(1\)](#), [what\(1\)](#)

**Name** scsi – configuration files for SCSI target drivers

**Description** The architecture of the Solaris SCSI subsystem distinguishes two types of device drivers: SCSI target drivers, and SCSI host adapter drivers. Target drivers like [sd\(7D\)](#) and [st\(7D\)](#) manage the device on the other end of the SCSI bus. Host adapter drivers manage the SCSI bus on behalf of all the devices that share it.

Drivers for host adapters provide a common set of interfaces for target drivers. These interfaces comprise the Sun Common SCSI Architecture (SCSA) which are documented as part of the Solaris DDI/DKI. See [scsi\\_ifgetcap\(9F\)](#), [scsi\\_init\\_pkt\(9F\)](#), and [scsi\\_transport\(9F\)](#) for further details of these, and associated routines.

Depending on the interconnect (transport), SCSI target devices are either self-identifying or rely on [driver.conf\(4\)](#) entries to be recognized by the system. For self-identifying target devices the driver binding is chosen based on the IEEE-1275 like 'compatible' forms of the target devices. Currently the Fibre Channel interconnects, [fcp\(7D\)](#), [ifp\(7D\)](#), [scsi\\_vhci\(7D\)](#), [sf\(7D\)](#), and the SATA framework drivers (see [sata\(7D\)](#)) are self-identifying. You must specify other possible interconnects target devices by using the target driver [driver.conf\(4\)](#) configuration files.

**Self-Identifying** Host adapter drivers of class scsi-self-identifying that dynamically create self-identifying target device children establish a *compatible* property on each child. The compatible property is an ordered array of strings, each string is a compatible *form*. High precedence forms are defined first. For a particular device, the highest precedence form that has an established driver alias selects the driver for the device. Driver associations to compatible forms, called aliases, are administered by way of [add\\_drv\(1M\)](#), [update\\_drv\(1M\)](#), and [rem\\_drv\(1M\)](#) utilities.

The forms for self-identifying SCSI target devices are derived from the SCSI target device's INQUIRY data. A diverse set of forms is defined, allowing for flexibility in binding.

From the SCSI INQUIRY data, three types of information are extracted: `scsi_dtype`, flag bits, and SCSI\_ASCII vendor product revision.

The `scsi_dtype` is the first component of most forms. It is represented as two hex digits. For nodes that represent embedded secondary functions, such as an embedded enclosure service or media changer, additional forms are generated that contain the dtype of the secondary function followed by the dtype of the device in which the secondary function is embedded.

For forms that use flag bits, all applicable flags are concatenated (in alphabetical order) into a single flags string. Removable media is represented by a flag. For forms that use the SCSI\_ASCII INQUIRY vendor, product, and revision fields, a one-way conversion algorithm translates SCSI\_ASCII to a IEEE 1275 compatible string.

It is possible that a device might change the INQUIRY data it returns over time as a result of a device initialization sequence or in response to out-of-band management. A device node's compatible property is based on the INQUIRY data when the device node was created.

The following forms, in high to low precedence order, are defined for SCSI target device nodes.

```
scsiclass,DDEEFF.vVVVVVVV.pPPPPPPPPPPPPPP.rRRRR (1 *1&2)
scsiclass,DDEE.vVVVVVVV.pPPPPPPPPPPPPPP.rRRRR (2 *1)
scsiclass,DDFFF.vVVVVVVV.pPPPPPPPPPPPPPP.rRRRR (3 *2)
scsiclass,DD.vVVVVVVV.pPPPPPPPPPPPPPP.rRRRR (4)
scsiclass,DDEEFF.vVVVVVVV.pPPPPPPPPPPPPPP (5 *1&2)
scsiclass,DDEE.vVVVVVVV.pPPPPPPPPPPPPPP (6 *1)
scsiclass,DDFFF.vVVVVVVV.pPPPPPPPPPPPPPP (7 *2)
scsiclass,DD.vVVVVVVV.pPPPPPPPPPPPPPP (8)
scsiclass,DDEEFF (9 *1&2)
scsiclass,DDEE (10 *1)
scsiclass,DDFFF (11 *2)
scsiclass,DD (12)
scsiclass (13)
```

\*1 only produced on a secondary function node

\*2 only produced on a node with flags

where:

|                |                                                                                                                                                                                                                                                                                                                              |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| v              | Is the letter v. Denotes the beginning of VVVVVVVV.                                                                                                                                                                                                                                                                          |
| VVVVVVVV       | Translated scsi_vendor: SCSI standard INQUIRY data “Vendor identification” SCSI_ASCII field (bytes 8-15).                                                                                                                                                                                                                    |
| p              | Is the letter p. Denotes the beginning of PPPPPPPPPPPPPPP.                                                                                                                                                                                                                                                                   |
| PPPPPPPPPPPPPP | Translated scsi_product: SCSI standard INQUIRY data “Product identification” SCSI_ASCII field (bytes 16-31).                                                                                                                                                                                                                 |
| r              | Is the letter r. Denotes the beginning of RRRR.                                                                                                                                                                                                                                                                              |
| RRRR           | Translated scsi_revision: SCSI standard INQUIRY data “Product revision level” SCSI_ASCII field (bytes 32-35).                                                                                                                                                                                                                |
| DD             | Is a two digit ASCII hexadecimal number. The value of the two digits is based one the SCSI “Peripheral device type” command set associated with the node. On a primary node this is the scsi_dtype of the primary command set; on a secondary node this is the scsi_dtype associated with the embedded function command set. |
| EE             | Same encoding used for DD. This form is only generated on secondary function nodes. The DD function is embedded in an EE device.                                                                                                                                                                                             |

FFF Concatenation, in alphabetical order, of the flag characters below. The following flag characters are defined:

R Removable media: Used when `scsi_rmb` is set

Forms using FFF are only generated if there are applicable flag characters.

Solaris might create additional *compatible* forms not described. These forms are for Solaris internal use only. Any additional use of these forms is discouraged. Future releases of Solaris might not produce these forms.

`driver.conf` Configuration files for SCSI target drivers should identify the host adapter driver implicitly using the *class* keyword to remove any dependency on the particular host adapter involved.

All host adapter drivers of class `scsi` recognize the following properties:

`target` Integer-valued SCSI target identifier that this driver claims.

`lun` Integer-valued SCSI logical unit number (LUN) that this driver claims.

All SCSI target driver configuration file device definitions except stub device definitions for discovery of `dev_id` must provide `target` and `lun` properties. These properties are used to construct the address part of the device name under `/devices`. The stub device definitions for discovery of `dev_id` must be able to specify or imply the host adapter drivers that might have children that bind to the target driver. So all SCSI target driver configuration file stub device definitions must be defined by property class or parent.

The SCSI target driver configuration files shipped with Solaris have entries for LUN 0 only. For devices that support other LUNs, such as some CD changers, the system administrator can edit the driver configuration file to add entries for other LUNs.

**Examples** EXAMPLE 1 An Example Configuration File for a SCSI Target Driver

The following is an example configuration file for a SCSI target driver called `toaster.conf`.

```
#
Copyright (c) 1992, by Sun Microsystems, Inc.
#
#ident "@(#)toaster.conf 1.2 92/05/12 SMI"
name="toaster" class="scsi" target=4 lun=0;
```

Add the following lines to `sd.conf` for a six- CD changer on target 3, with LUNs 0 to 5.

```
name="sd" class="scsi" target=3 lun=1;
name="sd" class="scsi" target=3 lun=2;
name="sd" class="scsi" target=3 lun=3;
name="sd" class="scsi" target=3 lun=4;
name="sd" class="scsi" target=3 lun=5;
```

**EXAMPLE 1** An Example Configuration File for a SCSI Target Driver (Continued)

It is not necessary to add the line for LUN 0, as it already exists in the file shipped with Solaris.

**EXAMPLE 2** A Stub Device Definition of `sd.conf`

The following line is a stub device definition which implies the host adapter drivers of class `scsi-self-identifying` might have children that bind to the `sd(7D)` driver:

```
name="sd" class="scsi-self-identifying";
```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Availability        | system/kernel   |
| Interface Stability | Committed       |

**See Also** [add\\_drv\(1M\)](#), [rem\\_drv\(1M\)](#), [update\\_drv\(1M\)](#), [driver.conf\(4\)](#), [attributes\(5\)](#), [fcp\(7D\)](#), [ifp\(7D\)](#), [sata\(7D\)](#), [scsi\\_vhci\(7D\)](#), [sd\(7D\)](#), [sf\(7D\)](#), [st\(7D\)](#), [scsi\\_ifgetcap\(9F\)](#), [scsi\\_init\\_pkt\(9F\)](#), [scsi\\_transport\(9F\)](#)

*Writing Device Drivers*

*ANS X3T9.2/82-2 SMALL COMPUTER SYSTEM INTERFACE (SCSI-1)*

*ANS X3T9.2/375D Small Computer System Interface - 2 (SCSI-2)*

*ANS X3T10/994D SCSI-3 Architecture Model (SAM)*

*IEEE 1275 SCSI Target Device Binding*

**Notes** With [driver.conf\(4\)](#) configuration, you need to ensure that the target and lun values claimed by your target driver do not conflict with existing target drivers on the system. For example, if the target is a direct access device, the standard `sd.conf` file usually makes `sd` claim it before any other driver has a chance to probe it.

**Name** securenets – configuration file for NIS security

**Synopsis** /var/yp/securenets

**Description** The /var/yp/securenets file defines the networks or hosts which are allowed access to information by the Network Information Service (“NIS”).

The format of the file is as follows:

- Lines beginning with the “#” character are treated as comments.
- Otherwise, each line contains two fields separated by white space. The first field is a netmask, the second a network.
- The netmask field may be either 255.255.255.255 (IPv4),  
ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff (IPv6), or the string “host” indicating that the second field is a specific host to be allowed access.

Both [ypserv\(1M\)](#) and [ypxfrd\(1M\)](#) use the /var/yp/securenets file. The file is read when the [ypserv\(1M\)](#) and [ypxfrd\(1M\)](#) daemons begin. If /var/yp/securenets is present, [ypserv\(1M\)](#) and [ypxfrd\(1M\)](#) respond only to IP addresses in the range given. In order for a change in the /var/yp/securenets file to take effect, you must kill and restart any active daemons using [ypstop\(1M\)](#) and [ypstart\(1M\)](#).

An important thing to note for all the examples below is that the server must be allowed to access itself. You accomplish this either by the server being part of a subnet that is allowed to access the server, or by adding an individual entry, as the following:

```
host 127.0.0.1
```

**Examples** **EXAMPLE 1** Giving Access for Individual Machines

If individual machines are to be given access, the entry could be:

```
255.255.255.255 192.9.1.20
```

or

```
host 192.0.1.20
```

**EXAMPLE 2** Giving Access to an Entire Class C Network

If access is to be given to an entire class C network, the entry could be:

```
255.255.255.0 192.9.1.0
```

**EXAMPLE 3** Giving Access to a Class B Network

If access is to be given to a class B network, the entry could be:

```
255.255.0.0 9.9.0.0
```

**EXAMPLE 4** Giving Access for an Individual IPv6 Address

To allow access for an individual IPv6 address:

```
ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff fec0::111:abba:ace0:fba5e:1
```

or

```
host fec0::111:abba:ace0:fba5e:1
```

**EXAMPLE 5** Giving Access for all IPv6 Addresses Starting with fe80

To allow access for all IPv6 addresses starting with fe80:

```
ffff:: fe80::
```

**Files** /var/yp/securenets Configuration file for NIS security.

**See Also** [ypserv\(1M\)](#), [ypstart\(1M\)](#), [ypstop\(1M\)](#), [ypxfrd\(1M\)](#)

**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 may not be used without permission.

**Name** sel\_config – selection rules for copy, cut, paste, drag and drop operations

**Synopsis** /usr/dt/config/sel\_config  
/usr/share/gnome/sel\_config

**Description** The sel\_config file specifies how a system that is configured with Trusted Extensions behaves when a user transfers data between windows that have different labels. Transfer operations include cut-and-paste, copy-and-paste, and drag-and-drop. There are two types of entries in this file: automatic confirmation and automatic reply.

**Automatic Confirmation** This type of entry specifies whether a confirmation window, the selection confirmer, displays. Each entry has the form:

*relationship: confirmation*

*relationship* identifies the result of comparing the selected data's source and destination windows' labels. There are three allowed values:

*upgradesl*           The source window's label is less than the destination window's label.  
*downgradesl*        The source window's label is higher than the destination window's label.  
*disjointsl*         The source and destination windows' labels are disjoint. Neither label dominates the other.

*confirmation* specifies whether to perform automatic confirmation. Allowed values are:

*n*     Use manual confirmation, that is, display the selection confirmer window. This is the default.  
*y*     Use automatic confirmation, that is, do not display the selection confirmer window.

**Automatic Cancellation** Some applications perform *hidden* paste operations to save data in the clipboard. These paste operations can target destinations that the user is not authorized to confirm. Therefore, the selection confirmer could appear to the user for data that the user can only press Cancel to dismiss. Automatic cancellation dismisses these intermediate selection confirmers.

There must be one entry of this form:

*autocancel: value*

If value is *y* (for yes), then the intermediate selection confirmers do not display to the user. If the value is *n* (for no), intermediate selection confirmers appear, and the user must confirm each window for the initial transaction to complete. The default is *y*.

**Automatic Reply** A single user operation can involve several flows of information between the source and destination windows. The automatic reply set of entries provides a means to reduce the number of confirmations that are required of the user.

There must be one entry of this form:



autoreply: *value*

If *value* is y (for yes), then the remaining entries of the set are used as attributes for the selection data (rather than the actual contents) to complete the operation without confirmation. If *value* is n (for no), then the remaining entries are ignored.

Defaults can be specified for any *type* field that appears in the Confirmer window. Below are some sample entries for defaults.

```
replytype: TARGETS
replytype: Pixel Sets
replytype: LENGTH
replytype: Type Of Monitor
```

The TARGETS entry, when used, returns the list of target atoms that are supported by the source window. The Pixel Sets and Type Of Monitor entries are used for animation during a drag-and-drop operation. The LENGTH entry specifies the number of bytes in the selection.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Availability        | system/trusted  |
| Interface Stability | Committed       |

**See Also** [attributes\(5\)](#)

“Rules When Changing the Level of Security for Data” in *Oracle Solaris Trusted Extensions Configuration and Administration*

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

**Name** sendmail, sendmail.cf, submit.cf – sendmail configuration files

**Synopsis** /etc/mail/sendmail.cf

/etc/mail/submit.cf

**Description** The `sendmail.cf` and `submit.cf` files are the configuration files for `sendmail(1M)`. Starting with version 8.12 of `sendmail`, which was shipped with version 9 of the Solaris operating system, two configuration files are used for submission and transmission of mail, instead of only `sendmail.cf`, as before. These are:

`sendmail.cf` Remains the principal `sendmail` configuration file. Used for the Mail Transmission Agent (MTA).

`submit.cf` Used for the Mail Submission Program (MSP). The MSP is used to submit mail messages. Unlike the MTA, it does not run as an SMTP daemon.

The MSP does not require root privileges, thus the two-file model provides better security than the pre-`sendmail 8.12` model, in which the MSP ran as a daemon and required root privileges.

In the default `sendmail` configuration, `sendmail` uses `submit.cf`, as indicated in `ps(1)` output. In `ps` output, you will observe two `sendmail` invocations, such as the ones below:

```
/usr/lib/sendmail -Ac -q15m
/usr/lib/sendmail -bd -q15m
```

The first indicates the use of `submit.cf`, with the client queue (`/var/spool/clientmqueue`) being checked—and, if needed, flushed—every 15 minutes. The second invocation runs `sendmail` as a daemon, waiting for incoming SMTP connections.

As shipped, `sendmail.cf` and, in particular, `submit.cf`, are appropriate for most environments. Where a knowledgeable system administrator needs to make a change, he should use the following procedures.

For `sendmail.cf`:

1. Change directories to the directory that contains the source files for the configuration files.

```
cd /etc/mail/cf/cf
```

2. Create a copy of the `sendmail` file for your system.

```
cp sendmail.mc 'hostname'.mc
```

3. Edit `'hostname'.mc`. Make changes suitable for your system and environment.

4. Run `make` to generate the configuration file.

```
/usr/bin/make 'hostname'.cf
```

5. Copy the newly generated file to its correct location.

```
cp 'hostname'.cf /etc/mail/sendmail.cf
```

6. Restart the sendmail service.

```
svcadm restart sendmail
```

You must restart sendmail for `sendmail.cf` file changes to take effect, as indicated in step 6. Steps 4 - 6 can be automated. See Automated Rebuilding of Configuration Files below.

For `submit.cf`:

1. Change directories to the directory that contains the source files for the configuration files.

```
cd /etc/mail/cf/cf
```

2. Create a copy of the submit file for your system.

```
cp submit.mc submit-'hostname'.mc
```

3. Edit `submit-'hostname'.mc`. Make changes suitable for your system and environment.

4. Run `make` to generate the configuration file.

```
/usr/bin/make submit-'hostname'.cf
```

5. Copy the newly generated file to its correct location.

```
cp submit-'hostname'.cf /etc/mail/submit.cf
```

You do not need to restart sendmail for changes to `submit.cf` to take effect. Steps 4 and 5 can be automated. See Automated Rebuilding of Configuration Files below.

Enabling Access to  
Remote Clients

The [sendmail\(1M\)](#) man page describes how the `config/local_only` property can be set to `true` or `false` to disallow or allow, respectively, access to remote clients for unmodified systems.

Automated Rebuilding  
of Configuration Files

Setting a value for the:

```
path_to_sendmail_mc
```

property for the service instance:

```
svc:/network/smtp:sendmail
```

and setting a value for the:

```
path_to_submit_mc
```

for the service instance:

```
svc:/network/sendmail-client:default
```

results in automated (re)building of configuration files.

The value for each of these properties should be a string that represents the path name of the .mc files referred to in steps 2 and 3 of both procedures in the DESCRIPTION. Recommended values are:

```
/etc/mail/cf/cf/'hostname'.mc
/etc/mail/cf/cf/submit-'hostname'.mc
```

Each property, if set, results in the corresponding .mc file being used to (re)build the matching .cf file when the corresponding instance/service is started.

These properties persist across upgrades and patches. To prevent a patch or upgrade from clobbering your .cf file, or renaming it to .cf.old, you can set the desired properties instead.

**Files** /etc/mail/cf/README Describes sendmail configuration files.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE               |
|---------------------|-------------------------------|
| Availability        | service/network/smtp/sendmail |
| Interface Stability | Committed                     |

**See Also** [ps\(1\)](#), [sendmail\(1M\)](#), [svcadm\(1M\)](#), [make\(1S\)](#), [attributes\(5\)](#)

*Oracle Solaris Administration: Network Services*

**Name** service\_bundle – service manifest file format

**Synopsis** /usr/share/lib/xml/dtd/service\_bundle.dtd.1

**Description** The service management facility, described in [smf\(5\)](#), utilizes an XML-based file format to marshal the description of a set of services or service instances between systems. This file is known as a service bundle. The primary form of a service bundle is the inventory of services that are provided by a package, which is called a *service manifest*.

The DTD describing the service\_bundle is provided at /usr/share/lib/xml/dtd/service\_bundle.dtd.1. The attributes and tags are fully described in the commented DTD. The services supplied with the operating system, stored under /lib/svc/manifest, provide examples of correctly formed service descriptions.

service\_bundle documents can also use the XML Inclusions (XInclude) facility to merge multiple documents into one. A service\_bundle document manipulator must therefore support the functionality defined by the XInclude specification.

A complete service description consists of the following:

- A set of properties that identify the service and identify its restarter
- A set of properties that identify each instance
- A set of framework property groups that describe the framework's understanding of each instance
- A set of method property groups as required by [svc.startd\(1M\)](#), or by a delegated restarter
- Additional optional method property groups
- A set of dependency property groups
- An optional group of properties that indicate services to which dependencies on the described service were added
- A set of application property groups or application-specific typed property groups containing application configuration data
- A template that describes supporting information about this service, such as a description, links to documentation, and metadata about property groups and properties.

The document type definition for the service bundle provides markup to define each of these aspects of a service description, as well as a number of entities that identify regular features in describing a service, such as the <create\_default\_instance> tag.

**Manifest Handling During Packaging Operations** Service manifests within packages should be identified with the class manifest in SVR4 packages and with the attribute restart\_fmri=svc:/system/manifest-import:default in IPS packages. Class action scripts that install and remove service manifests are included in the SVR4 packaging subsystem. When such a package is installed, the service manifest is imported.

When a package is uninstalled, service instances in the manifest that are disabled are deleted. Any services in the manifest with no remaining instances are also deleted.

If the `-R` option is supplied to [pkg\(1\)](#), [pkgadd\(1M\)](#), or [pkgrm\(1M\)](#), the actions described in this section are done when the system is next rebooted with that alternate root path.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Availability   | system/core-os  |
| Stability      | Committed       |

**See Also** [pkgadd\(1M\)](#), [pkgrm\(1M\)](#), [svcadm\(1M\)](#), [svccfg\(1M\)](#), [svc.startd\(1M\)](#), [libscf\(3LIB\)](#), [attributes\(5\)](#), [locale\(5\)](#), [smf\(5\)](#), [smf\\_method\(5\)](#), [smf\\_template\(5\)](#)

[pkg\(1\)](#), [pkg\(5\)](#)

**Notes** Nested `service_bundle` elements must be of the same type.

**Name** service\_provider.conf – service provider configuration file

**Synopsis** service\_provider.conf

**Description** service\_provider.conf contains information about the device type that the service provider supports. This information includes the pathname of the service provider library, the library version and other library characteristics that are required by the system administrative command, `datadm(1M)`. `datadm(1M)` puts this information in the DAT static register file, `dat.conf(4)`.

The `datadm` program enumerates each device entry into a list of interface adapters, that is, interfaces to external network that are available to uDAPL consumers. This new list of interface adapters is appended to other service providers' information in the DAT static registry, `dat.conf`. You can do this if you invoke the `datadm` program with the `-a` option and the pathname of the `service_provider.conf` file.

Each entry in the `service_provider.conf` is a single line of 7 fields.

The following shows the order of the fields in a `service_provider.conf` entry:

```
"driver_name" "API_version" "threadsafe_library" | \
 nonthreadsafe_library"\
"default_version" | nondefault_version" \
 "service_provider_library_pathname"\
"service_provider_version" "service_provider_instance_data"
```

The fields are defined as follows:

|                                                          |                                                                                                                                                                                               |
|----------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>driver_name</i>                                       | Specifies a driver name in the format of <code>driver_name=value pair</code> , for example, <code>driver_name=tavor</code> .                                                                  |
| <i>API_version</i>                                       | Specifies the API version of the service provider library: For example, "u"major.minor is u1.2.                                                                                               |
| <i>threadsafe_library</i>   <i>nonthreadsafe_library</i> | Specifies a threadsafe or non-threadsafe library.                                                                                                                                             |
| <i>default_version</i>   <i>nondefault_version</i>       | Specifies a default or non-default version of library. A service provider can offer several versions of the library. If so, one version is designated as default with the rest as nondefault. |
| <i>service_provider_library_pathname</i>                 | Specifies the pathname of the library image.                                                                                                                                                  |
| <i>service_provider_version</i>                          | Specifies the version of the service provider. By convention, specify the company stock symbol as the service provider, followed by major and minor version numbers, for example, SUNW1.0.    |
| <i>service_provider_instance_data</i>                    | Specifies the service provider instance data.                                                                                                                                                 |

**Examples** EXAMPLE 1 Using a Logical Device Name

The following example `service_provider.conf` entry uses a logical device name:

```
#
Sample service_provider.conf entry showing an uDAPL 1.2 service
provider, udapl_tavor.so.1 supporting a device with a driver named
tavor
driver_name=tavor u1.2 nonthreadsafe default udapl_tavor.so.1 \
 SUNW.1.0 ""
```

## EXAMPLE 2 Using a Physical Device Name

The following example `service_provider.conf` uses a physical device name:

```
#
Sample service_provider.conf entry showing an uDAPL 1.2
service provider, udapl_tavor.so.1 supporting a device named
pci15b3,5a44 that can be located under /devices
#
pci15b3,5a44 u1.2 nonthreadsafe default \
 /usr/lib/tavor/udapl_tavor.so.1 SUNWudaplt1.0 ""
```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Committed       |

**See Also** [datadm\(1M\)](#), [dat.conf\(4\)](#), [attributes\(5\)](#)



**Name** services – Internet services and aliases

**Synopsis** /etc/inet/services

/etc/services

**Description** The `services` file is a local source of information regarding each service available through the Internet. The `services` file can be used in conjunction with or instead of other services sources, including the NIS maps “`services.byname`”. Programs use the `getservbyname(3SOCKET)` routines to access this information.

The `services` file contains an entry for each service. Each entry has the form:

*service-name port/protocol aliases*

*service-name* This is the official Internet service name.

*port/protocol* This field is composed of the port number and protocol through which the service is provided, for instance, `512/tcp`.

*aliases* This is a list of alternate names by which the service might be requested.

Fields can be separated by any number of SPACE and/or TAB characters. A number sign (#) indicates the beginning of a comment; any characters that follow the comment character up to the end of the line are not interpreted by routines which search the file.

Service names may contain any printable character other than a field delimiter, a NEWLINE, or a comment character.

Any changes to a port assignment do not affect the actual port registration of the service.

**Files** /etc/nsswitch.conf configuration file for name-service switch

**See Also** `getservbyname(3SOCKET)`, `inetd.conf(4)`, `nsswitch.conf(4)`

**Notes** /etc/inet/services is the official SVR4 name of the `services` file. The symbolic link /etc/services exists for BSD compatibility.

**Name** shadow – shadow password file

**Description** /etc/shadow is an access-restricted ASCII system file that stores users' encrypted passwords and related information. The shadow file can be used in conjunction with other shadow sources, including the NIS maps passwd.byname and passwd.byuid. Programs use the [getspnam\(3C\)](#) routines to access this information.

The fields for each user entry are separated by colons. Each user is separated from the next by a NEWLINE. Unlike the /etc/passwd file, /etc/shadow does not have general read permission.

Each entry in the shadow file has the form:

```
username:password:lastchg:min:max:warn:inactive:expire:flag
```

The fields are defined as follows:

|                 |                                                                                                                                                                                                                                                                                                                 |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>username</i> | The user's login name (UID).                                                                                                                                                                                                                                                                                    |
| <i>password</i> | An encrypted password for the user generated by <a href="#">crypt(3C)</a> , a <i>lock</i> string to indicate that the login is not accessible, or no string, which shows that there is no password for the login.<br><br>The lock string is defined as *LK* in the first four characters of the password field. |
| <i>lastchg</i>  | The number of days between January 1, 1970, and the date that the password was last modified. The <i>lastchg</i> value is a decimal number, as interpreted by <a href="#">strtol(3C)</a> .                                                                                                                      |
| <i>min</i>      | The minimum number of days required between password changes. This field must be set to 0 or above to enable password aging.                                                                                                                                                                                    |
| <i>max</i>      | The maximum number of days the password is valid.                                                                                                                                                                                                                                                               |
| <i>warn</i>     | The number of days before password expires that the user is warned.                                                                                                                                                                                                                                             |
| <i>inactive</i> | The number of days of inactivity allowed for that user. This is counted on a per-machine basis; the information about the last login is taken from the machine's last log file.                                                                                                                                 |
| <i>expire</i>   | An absolute date expressed as the number of days since the Unix Epoch (January 1, 1970). When this number is reached the login can no longer be used. For example, an <i>expire</i> value of 13514 specifies a login expiration of January 1, 2007.                                                             |
| <i>flag</i>     | Failed login count in low order four bits; remainder reserved for future use, set to zero.                                                                                                                                                                                                                      |

A value of -1 for *min*, *max*, or *warn* disables password aging.

The encrypted password consists of at most `CRYPT_MAXCIPHERTEXTLEN` characters chosen from a 64-character alphabet (., /, 0–9, A–Z, a–z). Two additional special characters: the dollar sign (\$) and the comma (,), can also be used and are defined in `crypt(3C)`. To update this file, use the `passwd(1)`, `useradd(1M)`, `usermod(1M)`, or `userdel(1M)` commands.

To make system administration manageable, `/etc/shadow` entries should appear in exactly the same order as `/etc/passwd` entries; this includes “+” and “-” entries if the `compat` source is being used (see `nsswitch.conf(4)`).

Values for the various time-related fields are interpreted as Greenwich Mean Time.

|              |                                 |                                        |
|--------------|---------------------------------|----------------------------------------|
| <b>Files</b> | <code>/etc/shadow</code>        | Shadow password file                   |
|              | <code>/etc/passwd</code>        | Password file                          |
|              | <code>/etc/nsswitch.conf</code> | Name-service switch configuration file |
|              | <code>/var/adm/lastlog</code>   | Time of last login                     |

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

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Committed       |

**See Also** `login(1)`, `passwd(1)`, `useradd(1M)`, `userdel(1M)`, `usermod(1M)`, `strtol(3C)`, `crypt(3C)`, `crypt_gensalt(3C)`, `getspnam(3C)`, `putspent(3C)`, `nsswitch.conf(4)`, `passwd(4)`, `attributes(5)`, `pam_unix_account(5)`, `pam_unix_auth(5)`

**Notes** If password aging is turned on in any name service the `passwd:` line in the `/etc/nsswitch.conf` file must have a format specified in the `nsswitch.conf(4)` man page.

If the `/etc/nsswitch.conf` `passwd` policy is not in one of the supported formats, logins will not be allowed upon password expiration, because the software does not know how to handle password updates under these conditions. See `nsswitch.conf(4)` for additional information.

The authorizations required to modify the various shadow fields are as follows:

|                       |                                                |                                        |
|-----------------------|------------------------------------------------|----------------------------------------|
| <code>password</code> | change one's own password                      | none required                          |
| <code>password</code> | change another users password                  | <code>solaris.passwd.assign</code>     |
| <code>password</code> | delete, set no login                           | <code>solaris.passwd.assign</code>     |
| <code>password</code> | set initial password for newly created account | <code>solaris.account.activate</code>  |
| <code>password</code> | lock, unlock existing account                  | <code>solaris.account.setpolicy</code> |
| <code>min</code>      | min days for password change                   | <code>solaris.account.setpolicy</code> |
| <code>max</code>      | max days for password change                   | <code>solaris.account.setpolicy</code> |
| <code>warn</code>     | warn days for password change                  | <code>solaris.account.setpolicy</code> |

|          |                         |                           |
|----------|-------------------------|---------------------------|
| inactive | inactivity days allowed | solaris.account.setpolicy |
| expire   | expiry date             | solaris.account.setpolicy |

**Name** sharetab – shared file system table

**Description** sharetab resides in directory `/etc/dfs` and contains a table of local resources shared by the `share` command.

Each line of the file consists of the following fields:

*pathname resource fstype specific\_options description*

where

*pathname*            Indicate the path name of the shared resource.

*resource*            Indicate the symbolic name by which remote systems can access the resource.

*fstype*                Indicate the file system type of the shared resource.

*specific\_options*    Indicate file-system-type-specific options that were given to the `share` command when the resource was shared.

*description*         Describe the shared resource provided by the system administrator when the resource was shared.

**See Also** [share\(1M\)](#)

**Name** shells – shell database

**Synopsis** /etc/shells

**Description** The shells file contains a list of the shells on the system. Applications use this file to determine whether a shell is valid. See [getusershell\(3C\)](#). For each shell a single line should be present, consisting of the shell's path, relative to root.

A hash mark (#) indicates the beginning of a comment; subsequent characters up to the end of the line are not interpreted by the routines which search the file. Blank lines are also ignored.

The following default shells are used by utilities: /bin/bash, /bin/csh, /bin/jsh, /bin/ksh, /bin/ksh93, /bin/pfcsh, /bin/pfksh, /bin/pfsh, /bin/sh, /bin/tcsh, /bin/zsh, /sbin/jsh, /sbin/sh, /usr/bin/bash, /usr/bin/csh, /usr/bin/jsh, /usr/bin/ksh, /usr/bin/ksh93, /usr/bin/pfcsh, /usr/bin/pfksh, /usr/bin/pfsh, and /usr/bin/sh, /usr/bin/tcsh, /usr/bin/zsh, and /usr/sfw/bin/zsh. /etc/shells overrides the default list.

Invalid shells in /etc/shells could cause unexpected behavior, such as being unable to log in by way of ftp(1).

**Files** /etc/shells list of shells on system

**See Also** [vipw\(1B\)](#), [sendmail\(1M\)](#), [getusershell\(3C\)](#), [aliases\(4\)](#)

**Name** `slp.conf` – configuration file for Service Location Protocol agents

**Synopsis** `/etc/inet/slp.conf`

**Description** `slp.conf` provides all Service Location Protocol (“SLP”) agents with their operational configuration. `slpd(1M)` reads `slp.conf` on startup. Service Agents (“SAs”) and User Agents (“UAs”) read `slp.conf` on invocation of the SA and UA library routines; configuration parameters are then cached on a per-process basis. All SA's must use the same set of properties as `slpd` on the local machine, since `slpd` acts as an SA server.

The configuration file format consists of a newline-delimited list of zero or more property definitions. Each property definition corresponds to a particular configurable SLP, network, or other parameter in one or more of the three SLP agents. The file format grammar is shown in *RFC 2234* as follows:

```

config-file = line-list
line-list = line / line line-list
line = property-line / comment-line
comment-line = ("#" / ";") 1*allchar newline
property-line = property newline
property = tag "=" value-list
tag = prop / prop "." tag
prop = 1*tagchar
value-list = value / value "," value-list
value = int / bool /
 "(" value-list ")" / string
int = 1*DIGIT
bool = "true" / "false" / "TRUE" / "FALSE"
newline = CR / (CRLF)
string = 1*stringchar
tagchar = DIGIT / ALPHA / tother / escape
tother = %x21-%x2d / %x2f /
 %x3a / %x3c-%x40 /
 %x5b-%x60 / %7b-%7e
 ; i.e., all characters except '.',
 ; and '='.
stringchar = DIGIT / ALPHA / sother / escape
sother = %x21-%x29 / %x2a-%x2b /
 %x2d-%x2f / %x3a-%x40 /
 %x5b-%x60 / %7b-%7e
 ; i.e., all characters except ',',
allchar = DIGIT / ALPHA / HTAB / SP
escape = "\" HEXDIG HEXDIG
 ; Used for reserved characters

```

The properties fall into one of the following categories:

- DA Configuration

- Static Scope Configuration
- Tracing and Logging
- Serialized Proxy Registrations
- Networking Configuration Parameters
- UA Configuration

DA Configuration The following are configuration properties and their parameters for DAs:

`net.slp.isDA`

|                 |               |
|-----------------|---------------|
| Setting Type    | Boolean       |
| Default Value   | False         |
| Range of Values | True or False |

A boolean that indicates whether `slpd(1M)` is to act as a DA. If `False`, `slpd(1M)` is not run as a DA.

`net.slp.DAHeartBeat`

|                 |                          |
|-----------------|--------------------------|
| Setting Type    | Integer                  |
| Default Value   | 10800 seconds (3 hours)  |
| Range of Values | 2000 – 259200000 seconds |

A 32-bit integer giving the number of seconds for the passive DA advertisement heartbeat. The default value is 10800 seconds. This property is ignored if `net.slp.isDA` is `False`.

`net.slp.DAAttributes`

|                 |                                        |
|-----------------|----------------------------------------|
| Setting Type    | List of Strings                        |
| Default Value   | Unassigned                             |
| Range of Values | List of Attribute Tag/Value List Pairs |

A comma-separated list of parenthesized attribute tag/value list pairs that the DA must advertise in DA advertisements. The property must be in the SLP attribute list wire format, which requires that you use a backslash (“\”) to escape reserved characters. See *RFC 2608* for more information on reserved characters, or refer to the *Oracle Solaris Administration: Network Services*.

Static Scope Configuration The following properties and their parameters allow you to configure various aspects of scope and DA handling:

`net.slp.useScopes`

|               |                                            |
|---------------|--------------------------------------------|
| Setting Type  | List of Strings                            |
| Default Value | Default, for SA and DA; unassigned for UA. |



## Range of Values      List of Strings

A list of strings indicating either the scopes that a UA or an SA is allowed to use when making requests, or the scopes a DA must support. If not present for the DA and SA, the default scope `Default` is used. If not present for the UA, then the user scoping model is in force, in which active and passive DA or SA discovery are used for scope discovery. The scope `Default` is used if no other information is available. If a DA or SA gets another scope in a request, a `SCOPE_NOT_SUPPORTED` error is returned, unless the request was multicast, in which case it is dropped. If a DA receives another scope in a registration, a `SCOPE_NOT_SUPPORTED` error will be returned. Unlike other properties, this property is read-only, so attempts to change it programmatically after the configuration file has been read are ignored.

### `net.slp.DAAddresses`

|                 |                              |
|-----------------|------------------------------|
| Setting Type    | List of Strings              |
| Default Value   | Unassigned                   |
| Range of Values | IPv4 addresses or host names |

A list of IP addresses or DNS-resolvable names that denote the DAs to use for statically configured UAs and SAs. The property is read by `slpd(1M)`, and registrations are forwarded to the DAs. The DAs are provided to UAs upon request. Unlike other properties, this property is read-only, so attempts to change it after the configuration file has been read are ignored.

The following grammar describes the property:

```

addr-list = addr / addr "," addr-list
addr = fqdn / hostnumber
fqdn = ALPHA / ALPHA *["-"] anum
anum = ALPHA / DIGIT
hostnumber = 1*3DIGIT 3["." 1*3DIGIT)

```

The following is an example using this grammar:

```
sawah,mandi,sambal
```

IP addresses can be used instead of host names in networks where DNS is not deployed, but network administrators are reminded that using IP addresses will complicate machine renumbering, since the SLP configuration property files in statically configured networks will have to be changed.

**Tracing and Logging** These properties direct tracing and logging information to be sent to `syslogd` at the `LOG_INFO` priority. These properties affect `slpd(1M)` only.

### `net.slp.traceDATraffic`

|              |         |
|--------------|---------|
| Setting Type | Boolean |
|--------------|---------|

Default Value      False  
Range of Values    True or False

Set `net.slp.traceDATraffic` to True to enable logging of DA traffic by `slpd`.

#### `net.slp.traceMsg`

Setting Type        Boolean  
Default Value       False  
Range of Values    True or False

Set `net.slp.traceMsg` to True to display details about SLP messages. The fields in all incoming messages and outgoing replies are printed by `slpd`.

#### `net.slp.traceDrop`

Setting Type        Boolean  
Default Value       False  
Range of Values    True or False

Set this property to True to display details when an SLPmessage is dropped by `slpd` for any reason.

#### `net.slp.traceReg`

Setting Type        Boolean  
Default Value       False  
Range of Values    True or False

Set this property to True to display the table of service advertisements when a registration or deregistration is processed by `slpd`.

Serialized Proxy Registrations    The following properties control reading and writing serialized registrations.

#### `net.slp.serializedRegURL`

Setting Type        String  
Default Value       Unassigned  
Range of Values    Valid URL

A string containing a URL pointing to a document, which contains serialized registrations that should be processed when the `slpd` starts up.

Networking Configuration Parameters The properties that follow allow you to set various network configuration parameters:

`net.slp.isBroadcastOnly`

|                 |               |
|-----------------|---------------|
| Setting Type    | Boolean       |
| Default Value   | False         |
| Range of Values | True or False |

A boolean that indicates if broadcast should be used instead of multicast.

`net.slp.multicastTTL`

|                 |                                   |
|-----------------|-----------------------------------|
| Setting Type    | Positive Integer                  |
| Default Value   | 255                               |
| Range of Values | A positive integer from 1 to 255. |

A positive integer less than or equal to 255 that defines the multicast TTL.

`net.slp.DAActiveDiscoveryInterval`

|                 |                           |
|-----------------|---------------------------|
| Setting Type    | Integer                   |
| Default Value   | 900 seconds (15 minutes)  |
| Range of Values | From 300 to 10800 seconds |

A 16-bit positive integer giving the number of seconds between DA active discovery queries. The default value is 900 seconds (15 minutes). If the property is set to zero, active discovery is turned off. This is useful when the DAs available are explicitly restricted to those obtained from the `net.slp.DAAddresses` property.

`net.slp.multicastMaximumWait`

|                 |                                 |
|-----------------|---------------------------------|
| Setting Type    | Integer                         |
| Default Value   | 15000 milliseconds (15 seconds) |
| Range of Values | 1000 to 60000 milliseconds      |

A 32-bit integer giving the maximum value for the sum of the `net.slp.multicastTimeouts` values and `net.slp.DADiscoveryTimeouts` values in milliseconds.

`net.slp.multicastTimeouts`

|                 |                           |
|-----------------|---------------------------|
| Setting Type    | List of Integers          |
| Default Value   | 3000, 3000, 3000, 3000    |
| Range of Values | List of Positive Integers |

A list of 32-bit integers used as timeouts, in milliseconds, to implement the multicast convergence algorithm. Each value specifies the time to wait before sending the next request, or until nothing new has been learned from two successive requests. In a fast network the aggressive values of 1000, 1250, 1500, 2000, 4000 allow better performance. The sum of the list must equal `net.slp.multicastMaximumWait`.

#### `net.slp.passiveDADetection`

|                 |               |
|-----------------|---------------|
| Setting Type    | Boolean       |
| Default Value   | True          |
| Range of Values | True or False |

A boolean indicating whether `slpd` should perform passive DA detection.

#### `net.slp.DADiscoveryTimeouts`

|                 |                                    |
|-----------------|------------------------------------|
| Setting Type    | List of Integers.                  |
| Default Value   | 2000, 2000, 2000, 2000, 3000, 4000 |
| Range of Values | List of Positive Integers          |

A list of 32-bit integers used as timeouts, in milliseconds, to implement the multicast convergence algorithm during active DA discovery. Each value specifies the time to wait before sending the next request, or until nothing new has been learned from two successive requests. The sum of the list must equal `net.slp.multicastMaximumWait`.

#### `net.slp.datagramTimeouts`

|                 |                           |
|-----------------|---------------------------|
| Setting Type    | List of Integers          |
| Default Value   | 3000, 3000, 3000          |
| Range of Values | List of Positive Integers |

A list of 32-bit integers used as timeouts, in milliseconds, to implement unicast datagram transmission to DAs. The *n*th value gives the time to block waiting for a reply on the *n*th try to contact the DA.

#### `net.slp.randomWaitBound`

|                 |                              |
|-----------------|------------------------------|
| Setting Type    | Integer                      |
| Default Value   | 1000 milliseconds (1 second) |
| Range of Values | 1000 to 3000 milliseconds    |

Sets the upper bound for calculating the random wait time before attempting to contact a DA.

`net.slp.MTU`

|                 |             |
|-----------------|-------------|
| Setting Type    | Integer     |
| Default Value   | 1400        |
| Range of Values | 128 to 8192 |

A 16-bit integer that specifies the network packet size, in bytes. The packet size includes IP and TCP or UDP headers.

`net.slp.interfaces`

|                 |                              |
|-----------------|------------------------------|
| Setting Type    | List of Strings              |
| Default Value   | Default interface            |
| Range of Values | IPv4 addresses or host names |

List of strings giving the IP addresses or host names of the network interface cards on which the DA or SA should listen on port 427 for multicast, unicast UDP, and TCP messages. The default value is unassigned, indicating that the default network interface card should be used. An example is:

```
195.42.42.42,195.42.142.1,195.42.120.1
```

The example machine has three interfaces on which the DA should listen. Note that if IP addresses are used, the property must be renumbered if the network is renumbered.

UA Configuration The following configuration parameters apply to the UA:

`net.slp.locale`

|                 |                                                                  |
|-----------------|------------------------------------------------------------------|
| Setting Type    | String                                                           |
| Default Value   | en                                                               |
| Range of Values | See <i>RFC 1766</i> for a list of the locale language tag names. |

A *RFC 1766* Language Tag for the language locale. Setting this property causes the property value to become the default locale for SLP messages.

`net.slp.maxResults`

|                 |                      |
|-----------------|----------------------|
| Setting Type    | Integer              |
| Default Value   | -1                   |
| Range of Values | -1, positive integer |

A 32 bit-integer that specifies the maximum number of results to accumulate and return for a synchronous request before the timeout, or the maximum number of results to return

through a callback if the request results are reported asynchronously. Positive integers and -1 are legal values. If the value of `net.slp.maxResults` is -1, all results should be returned.

`net.slp.typeHint`

|                 |                    |
|-----------------|--------------------|
| Setting Type    | List of Strings    |
| Default Value   | Unassigned         |
| Range of Values | Service type names |

A list of service type names. In the absence of any DAs, UAs perform SA discovery to find scopes. If the `net.slp.typeHint` property is set, only SA's advertising types on the list respond. Note that UAs set this property programmatically. It is not typically set in the configuration file. The default is unassigned, meaning do not restrict the type.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE     |
|---------------------|---------------------|
| Availability        | service/network/slp |
| CSI                 | Enabled             |
| Interface Stability | Committed           |

**See Also** [slpd\(1M\)](#), [slpd.reg\(4\)](#), [slp\\_api\(3SLP\)](#), [slp\(7P\)](#)

*Oracle Solaris Administration: Network Services*

Alvestrand, H. *RFC 1766: Tags for the Identification of Languages*. Network Working Group. March 1995.

Crocker, D., Overell, P. *RFC 2234, Augmented BNF for Syntax Specifications: ABNF*. The Internet Society. 1997.

Kempf, J. and Guttman, E. *RFC 2614, An API for Service Location*. The Internet Society. June 1999.

**Name** slpd.reg – serialized registration file for the service location protocol daemon (slpd)

**Synopsis** /etc/inet/slpd.reg

**Description** The serialized registration file contains a group of registrations that `slpd(1M)` registers when it starts. These registrations are primarily for older service programs that do not internally support SLP and cannot be converted. The character format of the registration file is required to be ASCII. To use serialized registrations, set the `net.slp.serializedRegURL` property in `slp.conf(4)` to point at a valid `slpd.reg` file. The syntax of the serialized registration file, in ABNF format (see *RFC 2234*), is as follows:

```

ser-file = reg-list
reg-list = reg / reg reg-list
reg = creg / ser-reg
creg = comment-line ser-reg
comment-line = ("#" / ";") 1*allchar newline
ser-reg = url-props [slist] [attr-list] newline
url-props = surl "," lang "," ltime ["," type] newline
surl = ;The registration's URL. See
 ; [8] for syntax.
lang = 1*8ALPHA ["-" 1*8ALPHA]
 ;RFC 1766 Language Tag see [6].
ltime = 1*5DIGIT
 ; A positive 16-bit integer
 ; giving the lifetime
 ; of the registration.
type = ; The service type name, see [7]
 ; and [8] for syntax.
slist = "scopes" "=" scope-list newline
scope-list = scope-name / scope-name "," scope-list
scope = ; See grammar of [7] for
 ; scope-name syntax.
attr-list = attr-def / attr-def attr-list
attr-def = (attr / keyword) newline
keyword = attr-id
attr = attr-id "=" attr-val-list
attr-id = ;Attribute id, see [7] for syntax.
attr-val-list = attr-val / attr-val "," attr-val-list
attr-val = ;Attribute value, see [7] for syntax
allchar = char / WSP
char = DIGIT / ALPHA / other
other = %x21-%x2f / %x3a-%x40 /
 %x5b-%x60 / %7b-%7e
 ; All printable, nonwhitespace US-ASCII
 ; characters.
newline = CR / (CRLF)

```

The syntax for attributes and attribute values requires that you use a backslash to escape special characters, in addition to non-ASCII characters, as specified in *RFC 2608*. The `slpd`

command handles serialized registrations exactly as if they were registered by an SA. In the `url-props` production, the `type` token is optional. If the `type` token is present for a service: URL, a warning is signalled, and the `type` name is ignored. If the maximum lifetime of 65535 seconds is specified, the registration is taken to be permanent, and it is continually refreshed by the DA or SA server until it exits.

Scopes can be included in a registration by including an attribute definition with tag `scopes` followed by a comma-separated list of scope names immediately after the `url-props` production. If the optional `scope-list` is present, the registrations are made in the indicated scopes; otherwise, they are registered in the scopes with which the DA or SA server was configured through the `net.slp.useScopes` property. If any conflicts occur between the scope list and the `net.slp.useScopes` property, an error message is issued by way of `syslog(3C)`. Refer to information regarding `LOG_INFO` in `syslog(3C)`.

Service advertisements are separated by a single blank line. Additionally, the file must end with a single blank line.

#### Examples **EXAMPLE 1** Using a Serialized Registration File

The following serialized registration file shows an instance of the service type `foo`, with a lifetime of 65535 seconds, in the `en` locale, with scope `somescope`:

```
register foo
service:foo://fooserver/foopath,en,65535
scopes=somescope
description=bogus
security=kerberos_v5
location=headquarters

next registration...
```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                    |
|---------------------|------------------------------------|
| Availability        | service/network/slp                |
| CSI                 | Enabled                            |
| Interface Stability | Committed                          |
| Standard            | See <a href="#">standards(5)</a> . |

**See Also** [slpd\(1M\)](#), [slp\\_api\(3SLP\)](#), [syslog\(3C\)](#), [slp.conf\(4\)](#), [attributes\(5\)](#)

Crocker, D. and Overall, P., *RFC 2234, Augmented BNF for Syntax Specifications: ABNF*, The Internet Society, November 1997.

Guttman, E., Perkins, C., Veizades, J., and Day, M., *RFC 2608, Service Location Protocol, Version 2*, The Internet Society, June 1999.



Kempf, J. and Guttman, E., *RFC 2614, An API for Service Location*, The Internet Society, June 1999.

**Name** smb – configuration properties for the Oracle Solaris SMB server and client

**Description** The behavior of the Oracle Solaris SMB server and client is defined by property values that are stored in the Service Management Facility, [smf\(5\)](#).

An authorized user can use the [sharectl\(1M\)](#) command to set global values for these properties in SMF.

The following list describes the properties:

**ads\_site**

Specifies the Active Directory site. Sites provide a mechanism to partition or delegate administration and policy management, which are typically used in large or complex domains.

The value should not be set if you do not have a local Active Directory site. By default, no value is set.

**autohome\_map**

Specifies the full path for the SMB autohome map file, `smbautohome`. The Oracle Solaris SMB server can automatically share home directories when a SMB client connects. The default path is `/etc`.

**client\_lmauth\_level**

Specifies the LAN Manager (LM) authentication level for the Oracle Solaris SMB client. The LM compatibility level controls the type of user authentication to use in workgroup mode or domain mode. The default value is 3.

The following describes the behavior at each level.

Oracle Solaris SMB Client

```

+---+-----+
| 1 | Uses LM and NTLM |
| | authentication |
+---+-----+
| 2 | Uses NTLM authentication |
| | |
+---+-----+
| 3 | Uses NTLMv2 authentication |
| | |
+---+-----+
| 4 | Uses NTLMv2 authentication |
+---+-----+
| 5 | Uses NTLMv2 authentication |
| | |
+---+-----+

```

`client_signing_required`

When SMB client signing is required, the Oracle Solaris SMB client signs all packets, and is not able to connect to servers configured with signing not enabled and not required. When SMB client signing is not required (`false`), the Oracle Solaris SMB client only signs packets if required by the server. Valid values are `true` and `false`. The default value is `false`.

`disposition`

A value that controls whether the Oracle Solaris SMB server disconnects the share or proceed if the map command fails. The disposition property only has meaning when the map property has been set. Otherwise it has no effect.

```
disposition = [continue | terminate]
```

`continue`

Proceed with share connection if the map command fails. This is the default in the event that disposition is not specified.

`terminate`

Disconnect the share if the map command fails.

`ddns_enable`

Enables or disables dynamic DNS updates. A value of `true` enables dynamic updates, while a value of `false` disables dynamic updates. By default, the value is `false`.

For Active Directory DNS servers, when the system is joined to the AD domain, secure updates (defined in *RFC 3007*) are used.

`dns_suffix`

Specifies a primary DNS suffix for this system. Some domain member systems have a primary DNS suffix that does not match the DNS name of the Active Directory domain (of which they are members). This creates a disjoint namespace. By default, no value is set.

`enforce_vczero`

Specifies the behavior on receiving a connection on virtual circuit (VC) zero. The SMB protocol default is to enforce VC zero behavior. When a client connects using VC zero the server disconnects all previous connections from that client. The default behavior can cause sessions to be disconnected when the DNS contains aliases for the server and shares are mapped using different names. To avoid disconnection, the default behavior can be changed by setting `enforce_vczero` to `false`. The default value is `true`.

`ipv6_enabled`

Enables IPv6 Internet protocol support within the Oracle Solaris SMB Service. Valid values are `true` and `false`. The default value is `false`.

`keep_alive`

Specifies the number of seconds before an idle SMB connection is dropped by the Oracle Solaris SMB server. If set to `0`, idle connections are not dropped. Valid values are `0` and from `20` seconds and above. The default value is `5400` seconds.

**map**

The value is a command to be executed on the Oracle Solaris SMB server when a SMB client connects to the share. The command can take the following arguments, which is substituted when the command is exec'd as described below:

**%U**

Windows username.

**%D**

Name of the domain or workgroup of %U.

**%h**

The server hostname.

**%M**

The client hostname, or "" if not available.

**%L**

The server NetBIOS name.

**%m**

The client NetBIOS name, or "" if not available. This option is only valid for NetBIOS connections (port 139).

**%I**

The IP address of the client machine.

**%i**

The local IP address to which the client is connected.

**%S**

The name of the share.

**%P**

The root directory of the share.

**%u**

The UID of the Unix user.

**max\_connections**

Specifies the maximum number of concurrent open connections on the SMB server. The value is an upper limit and it has no effect when the number of concurrent connections is below the limit. The only reason to change this value is to enforce a lower or higher limit on the number of concurrent connections. The default value is 32768.

**max\_workers**

Specifies the maximum number of worker threads that are launched to process incoming SMB requests. The SMB `max_mpx` value, which indicates to a SMB client the maximum number of outstanding SMB requests that it can have pending on the Oracle Solaris SMB server, is derived from the `max_workers` value. To ensure compatibility with older versions

of Windows the lower 8-bits of `max_mpx` must not be zero. If the lower byte of `max_workers` is zero, 64 is added to the value. Thus the minimum value is 64 and the default value, which appears in `sharctl(1M)` as 1024, is 1088.

#### `netbios_scope`

Specifies the NetBIOS scope identifier, which identifies logical NetBIOS networks that are on the same physical network. When you specify a NetBIOS scope identifier, the Oracle Solaris SMB server filters the number of machines that are listed in the browser display to make it easier to find other hosts. The value is a text string that represents a domain name. By default, no value is set.

#### `pdcc`

Specifies the preferred IP address for the domain controller with which the Oracle Solaris SMB server communicates. This property is sometimes used when there are multiple domain controllers to indicate which one is preferred. If the specified domain controller responds, it is chosen even if the other domain controllers are also available. By default, no value is set.

#### `restrict_anonymous`

Disables anonymous access to IPC\$ on the Oracle Solaris SMB server, which requires that the SMB client be authenticated to get access to MSRPC services through IPC\$. A value of `true` disables anonymous access to IPC\$, while a value of `false` enables anonymous access.

#### `server_lmauth_level`

Specifies the LAN Manager (LM) authentication level for the Oracle Solaris SMB server. The LM compatibility level controls the type of user authentication to use in workgroup mode or domain mode. The default value is 3.

The following describes the behavior at each level:

```

 Oracle Solaris SMB Server
+-----+
| 2 | Accepts LM, NTLM and NTLMv2 |
| | authentication |
+---+-----+
| 3 | Accepts LM, NTLM and NTLMv2 |
| | authentication |
+---+-----+
| 4 | Accepts NTLM and NTLMv2 |
| | authentication |
+---+-----+
| 5 | Accepts NTLMv2 |
| | authentication |
+---+-----+

```

#### `server_signing_enabled`

Enables SMB server signing. When server signing is enabled but not required, packets signing is at the discretion of the client. When Oracle Solaris SMB server signing is not enabled and not required, packets are signed if, and only if, the client requires signing.

When `server_signing_required=true`, `server_signing_enabled` is treated as `true`. Valid values are `true` and `false`. The default value is `false`.

#### `server_signing_required`

When SMB server signing is required, all packets must be signed or they are rejected. Clients that do not support signing are unable to connect to the Oracle Solaris SMB server. Valid values are `true` and `false`. The default value is `false`.

#### `system_comment`

Specifies an optional description for the system, which is a text string. This property value might appear in various places, such as Network Neighborhood or Network Places on Windows clients. By default, no value is set.

#### `unmap`

The value is a command to be executed on the Oracle Solaris SMB server when a SMB client disconnects the share. The command can take the same substitutions listed on the `map` property.

#### `wins_exclude`

Specifies a comma-separated list of network interfaces that should not be registered with WINS for the Oracle Solaris SMB server. NetBIOS host announcements are not made on excluded interfaces.

#### `wins_server_1`

Specifies the IP address of the primary WINS server. By default, no value is set.

#### `wins_server_2`

Specifies the IP address of the secondary WINS server. By default, no value is set.

**Attributes** See the [attributes\(5\)](#) man page for descriptions of the following attributes:

| ATTRIBUTETYPE       | ATTRIBUTEVALUE          |
|---------------------|-------------------------|
| Availability        | service/file-system/smb |
| Interface Stability | Uncommitted             |

**See Also** [sharectl\(1M\)](#), [smbadm\(1M\)](#), [smbd\(1M\)](#), [smbstat\(1M\)](#), [attributes\(5\)](#), [smf\(5\)](#)

*RFC 3007*

**Name** smbautohome – SMB autohome configuration

**Synopsis** smbautohome

**Description** The Solaris SMB server can automatically share home directories when an SMB client connects. The autohome map file, `/etc/smbautohome`, uses the search options and rules to determine whether to share a home directory when an SMB client connects to the server.

For example, the following entries specify the autohome rules for a particular environment:

```
+nsswitch dn=ad,dn=sun,dn=com,ou=users
jane /home/?/& dn=ad,dn=sun,dn=com,ou=users
```

The `nsswitch` autohome entry uses the naming service to match users to home directories. The second autohome entry specifies that the home directory for user `jane` is `/home/j/jane`.

**autohome Map Entry Format** A map entry, which is also referred to as a mapping, uses the following format:

```
key location [options]
```

*key* is a user name, *location* is the fully qualified path for the user's home directory, and *options* specifies the share options, for example, an AD container or description. See [share\\_smb\(1M\)](#) for information on share options.

If you intend to publish the share in Active Directory (AD), you *must* specify an AD container name, which is specified as a comma-separated list of attribute name-value pairs. The attributes use the LDAP distinguished name (DN) or relative distinguished name (RDN) format.

The DN or RDN must be specified in LDAP format by using the following attribute types:

- `cn=` represents the common name
- `ou=` represents the organizational unit
- `dc=` represents the domain component

The attribute type that is used to describe an object's RDN is called a *naming attribute*. AD uses the naming attributes as follows:

- `cn` for the user object class
- `ou` for the OU (organizational unit) object class
- `dc` for the `domainDns` object class

**autohome Map Key Substitution** The autohome feature supports the following wildcard substitutions for the value of the key field:

- The ampersand character (`&`) is expanded to the value of the key field for the entry in which it occurs. In the following example, `&` expands to `jane`:

```
jane /home/&
```

- The question mark character (`?`) is expanded to the value of the first character in the key field for the entry in which it occurs. In the following example, `?` expands to `j`:

```
jane /home/?/?&
```

**Wildcard Rule** When supplied in the key field, the asterisk character (\*) is recognized as the “catch-all” entry. Such an entry matches any key not previously matched.

For example, the following entry would map any user to a home directory in /home in which the home directory name was the same as the user name:

```
* /home/&
```

The wildcard rule is *only* applied if an appropriate rule is not matched by another map entry.

**NSSwitch Map** The `nsswitch` map is used to request that the home directory be obtained from a password database, such as the local, NIS, or LDAP databases. If an AD path is appended, it is used to publish shares.

```
+nsswitch
```

Like the “catch-all” entry, the `nsswitch` map is *only* searched if an appropriate rule is not matched by another map entry.

The wildcard and `nsswitch` rules are mutually exclusive. Do not include an `nsswitch` rule if a wildcard rule has already been defined.

**Files** /etc/smbautohome

**Attributes** See the [attributes\(5\)](#) man page for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE         |
|---------------------|-------------------------|
| Availability        | service/file-system/smb |
| Interface Stability | Uncommitted             |

**See Also** [sharectl\(1M\)](#), [smbadm\(1M\)](#), [smbd\(1M\)](#), [smbstat\(1M\)](#), [smb\(4\)](#), [attributes\(5\)](#)



**Name** smhba.conf – configuration file for the SMHBAAPI library

**Description** The `/etc/smhba.conf` file is used to specify the Vendor-Specific Libraries that are installed on the system. This file is used by the Common Library to load the individual VSLs when `HBA_LoadLibrary(3HBAAPI)` is called. If changes are made to the file while the library is in use, the library should be freed and reloaded. A version 1 VSL is compatible only with a version 1 Common Library. A version 2 VSL is compatible with both a version 1 and a version 2 Common Library.

Each VSL entry is a single line of the form:

```
"name" "library path"
```

where:

*name* is the description of library. The library name should be prepended with the domain of the manufacturer of the library.

*library path* is the absolute path to the shared object library file.

**Examples** EXAMPLE 1 Contents of `/etc/smhba.conf`

```
#
This file contains names and references to SM-HBA libraries
#
Format:
#
<library name> <library pathname>
#
The library name should be prepended with the domain of
the manufacturer or driver author.
com.sun.sashba /usr/lib/libsun_sas.so.1
com.sun.sashba64 /usr/lib/64/libsun_sas.so.1
```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------|
| Interface Stability | Committed                                                                                     |
| Standard            | ANSI INCITS 428 Storage Management Host Bus Adapter Application Programming Ingerface(SM-HBA) |

**See Also** [HBA\\_LoadLibrary\(3HBAAPI\)](#), [libSMHBAAPI\(3LIB\)](#), [attributes\(5\)](#)

**Notes** The SMHBAAPI library is provided in both 32- and 64-bit versions, but only one configuration file is specified. As a result, both 32- and 64-bit VSL libraries must be specified within the same file. When using the 32-bit Common Library, the 64-bit VSLs will fail to load.

When using the 64-bit Common Library, the 32-bit VSLs will fail to load. These failures are silently ignored by the Common Library during normal usage, but can result in warning messages when running client applications in a debugger.

**Name** s ndr – SNDR parameter values

**Synopsis** /etc/default/s ndr

**Description** The s ndr file resides in /etc/default and provides startup parameters for the [s ndr\(1M\)](#) and [s ndr syncd\(1M\)](#) daemons.

The s ndr file format is ASCII and comment lines begin with the crosshatch (#) character. Parameters consist of a keyword followed by an equal (=) sign followed by the parameter value of the form:

keyword=value

The following parameters are currently supported in the s ndr file:

SNDR\_THREADS=num

Sets the maximum number of connections allowed to the server over connection-oriented transports. By default, the number of connections is 16.

SNDR\_LISTEN\_BACKLOG=num

Sets connection queue length for the RDC TCP over a connection-oriented transport. The default value is 10 entries.

SNDR\_TRANSPORT=string

Sets the transport used for the RDC connection. If IPv6 is installed, the default value is "/dev/tcp" or "/dev/tcp6."

**Attributes** See [attributes\(5\)](#) for a description of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE               |
|---------------------|-------------------------------|
| Architecture        | x86                           |
| Availability        | storage/avs/avs-remote-mirror |
| Interface Stability | Committed                     |

**See Also** [s ndr\(1M\)](#), [s ndr syncd\(1M\)](#)

**Name** sock2path.d – socket mapping file that maps sockets to transport providers

**Synopsis** /etc/sock2path.d

**Description** /etc/sock2path.d is a directory containing files with mappings between the [socket\(3SOCKET\)](#) call parameters and the transport provider driver. The mapping file format is described in the [soconfig\(1M\)](#) manual page.

The mappings in the sock2path.d directory are configured by the soconfig utility at boot. The mappings are available to applications once the service management facility, [smf\(5\)](#), reaches the single-user milestone:

```
svc:/milestone/single-user:default
```

**Examples** **EXAMPLE 1** An Example of a Mapping File

The following is an example of a mapping file:

| # | Family | Type | Protocol | Module | Path           |
|---|--------|------|----------|--------|----------------|
|   | 2      | 2    | 0        | tcp    |                |
|   | 2      | 2    | 6        | tcp    |                |
|   | 26     | 2    | 0        | tcp    |                |
|   | 26     | 2    | 6        | tcp    |                |
|   | 2      | 1    | 0        | udp    |                |
|   | 2      | 1    | 17       | udp    |                |
|   | 26     | 1    | 0        | udp    |                |
|   | 26     | 1    | 17       | udp    |                |
|   | 1      | 2    | 0        |        | /dev/ticotsord |
|   | 1      | 6    | 0        |        | /dev/ticotsord |
|   | 1      | 1    | 0        |        | /dev/ticlts    |
|   | 2      | 4    | 0        | icmp   |                |
|   | 26     | 4    | 0        | icmp   |                |
|   | 24     | 4    | 0        | rts    |                |
|   | 27     | 4    | 2        |        | /dev/keysock   |

**See Also** [init\(1M\)](#), [soconfig\(1M\)](#), [socket\(3SOCKET\)](#), [smf\(5\)](#)

**Name** space – disk space requirement file

**Description** space is an ASCII file that gives information about disk space requirements for the target environment. The space file defines space needed beyond what is used by objects defined in the [prototype\(4\)](#) file; for example, files which will be installed with the [installf\(1M\)](#) command. The space file should define the maximum amount of additional space that a package will require.

The generic format of a line in this file is:

*pathname blocks inodes*

Definitions for the fields are as follows:

|                 |                                                                                                                                                                 |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>pathname</i> | Specify a directory name which may or may not be the mount point for a filesystem. Names that do not begin with a slash ('/') indicate relocatable directories. |
| <i>blocks</i>   | Define the number of disk blocks required for installation of the files and directory entries contained in the pathname (using a 512-byte block size).          |
| <i>inodes</i>   | Define the number of inodes required for installation of the files and directory entries contained in the pathname.                                             |

**Examples** EXAMPLE 1 A sample file.

```
extra space required by config data which is
dynamically loaded onto the system
data 500 1
```

**See Also** [installf\(1M\)](#), [prototype\(4\)](#)

*Application Packaging Developer's Guide*

**Name** ssh\_config – ssh configuration file

**Synopsis** /etc/ssh/ssh\_config

\$HOME/.ssh/config

**Description** The first ssh\_config path, above, provides the system-wide defaults for ssh(1). The second version is user-specific defaults for ssh.

ssh obtains configuration data from the following sources, in this order: command line options, user's configuration file (\$HOME/.ssh/config), and system-wide configuration file (/etc/ssh/ssh\_config). For each parameter, the first obtained value is used. The configuration files contain sections bracketed by Host specifications, and that section is applied only for hosts that match one of the patterns given in the specification. The matched host name is the one given on the command line.

Since the first obtained value for each parameter is used, host-specific declarations should be given near the beginning of the file, and general defaults at the end.

The configuration file has the following format and syntax:

- Empty lines and lines starting with # are comments.
- Non-commented lines are of the form:  
*keyword arguments*
- Configuration options can be separated by white space or optional whitespace and exactly one equal sign. The latter format allows you to avoid the need to quote white space when specifying configuration options using the -o option to ssh, scp, and sftp.

The possible keywords and their meanings are listed in the following list. Keywords are case-insensitive and arguments are case-sensitive.

|             |                                                                                                                                                                                                                              |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BatchMode   | The argument must be yes or no. If set to yes, passphrase/password querying is disabled. This option is useful in scripts and other batch jobs where you have no user to supply the password.                                |
| BindAddress | Specify the interface to transmit from on machines with multiple interfaces or aliased addresses. This option does not work if UsePrivilegedPort is set to yes.                                                              |
| CheckHostIP | If this flag is set to yes, ssh additionally checks the host IP address in the known_hosts file. This allows ssh to detect if a host key changed due to DNS spoofing. If the option is set to no, the check is not executed. |

|                                                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cipher                                                                                                                                      | Specifies the cipher to use for encrypting the session in protocol version 1. Only a single cipher can be specified. Currently, <code>blowfish</code> , <code>3des</code> , and <code>des</code> are supported. <code>3des</code> (triple-des) is an encrypt-decrypt-encrypt triple with three different keys. It is believed to be secure. <code>blowfish</code> is a fast block cipher. It appears very secure and is much faster than <code>3des</code> . <code>des</code> is only supported in the <code>ssh</code> client for interoperability with legacy protocol 1 implementations that do not support the <code>3des</code> cipher. Its use is strongly discouraged due to cryptographic weaknesses. The default is <code>3des</code> . |
| Ciphers                                                                                                                                     | Specifies the ciphers allowed for protocol version 2 in order of preference. Multiple ciphers must be comma separated.<br><br>The default cipher list contains all supported ciphers in this order:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <pre> aes128-ctr, aes192-ctr, aes256-ctr, arcfour128, arcfour256, arcfour, aes128-cbc, aes192-cbc, aes256-cbc, 3des-cbc,blowfish-cbc </pre> | While CBC modes are not considered as secure as other modes in connection with the SSH protocol 2 they are present at the back of the default client cipher list for backward compatibility with SSH servers that do not support other cipher modes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| ClearAllForwardings                                                                                                                         | Specifies that all local, remote, and dynamic port forwardings specified in the configuration files or on the command line be cleared. This option is primarily useful when used from the <code>ssh</code> command line to clear port forwardings set in configuration files and is automatically set by <code>scp(1)</code> and <code>sftp(1)</code> . The argument must be <code>yes</code> or <code>no</code> . The default is <code>no</code> .                                                                                                                                                                                                                                                                                              |
| Compression                                                                                                                                 | Specifies whether to use compression. The argument must be <code>yes</code> or <code>no</code> . Defaults to <code>no</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| CompressionLevel                                                                                                                            | Specifies the compression level to use if compression is enabled. The argument must be an integer from 1 (fast) to 9 (slow, best). The default level is 6, which is good for most applications. This option applies to protocol version 1 only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ConnectionAttempts | Specifies the number of tries (one per second) to make before falling back to <code>rsh</code> or exiting. The argument must be an integer. This can be useful in scripts if the connection sometimes fails. The default is 1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| ConnectTimeout     | Specifies the timeout (in seconds) used when connecting to the <code>ssh</code> server, instead of using the default system TCP timeout. This value is used only when the target is down or truly unreachable, not when it refuses the connection.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| DisableBanner      | <p>If set to <code>yes</code>, disables the display of the banner message. If set to <code>in-exec-mode</code>, disables the display of banner message when in remote command mode only.</p> <p>The default value is <code>no</code>, which means that the banner is displayed unless the log level is <code>QUIET</code>, <code>FATAL</code>, or <code>ERROR</code>. See also the <code>Banner</code> option in <a href="#">sshd_config(4)</a>. This option applies to protocol version 2 only.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| DynamicForward     | <p>Specifies that a TCP/IP port on the local machine be forwarded over the secure channel. The application protocol is then used to determine where to connect to from the remote machine.</p> <p>The argument must be <code>[bind_address:]port</code>. IPv6 addresses can be specified by enclosing addresses in square brackets or by using an alternative syntax: <code>[bind_address/]port</code>. By default, the local port is bound in accordance with the <code>GatewayPorts</code> setting. However, an explicit <code>bind_address</code> can be used to bind the connection to a specific address. The <code>bind_address</code> of <code>localhost</code> indicates that the listening port be bound for local use only, while an empty address or <code>*</code> indicates that the port should be available from all interfaces.</p> <p>Currently the <code>SOCKS4</code> and <code>SOCKS5</code> protocols are supported, and <code>ssh</code> acts as a <code>SOCKS</code> server. Multiple forwardings can be specified and additional</p> |



---

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | forwardings can be specified on the command line. Only a user with enough privileges can forward privileged ports.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| EscapeChar    | Sets the escape character. The default is tilde (~). The escape character can also be set on the command line. The argument should be a single character, ^, followed by a letter, or none to disable the escape character entirely (making the connection transparent for binary data).                                                                                                                                                                                                                                                                                      |
| FallBackToRsh | Specifies that if connecting with ssh fails due to a connection refused error (there is no <code>sshd(1M)</code> listening on the remote host), <code>rsh(1)</code> should automatically be used instead (after a suitable warning about the session being unencrypted). The argument must be yes or no.                                                                                                                                                                                                                                                                      |
| ForwardAgent  | Specifies whether the connection to the authentication agent (if any) is forwarded to the remote machine. The argument must be yes or no. The default is no.<br><br>Agent forwarding should be enabled with caution. Users with the ability to bypass file permissions on the remote host (for the agent's Unix-domain socket) can access the local agent through the forwarded connection. An attacker cannot obtain key material from the agent, however he can perform operations on the keys that enable him to authenticate using the identities loaded into the agent.  |
| ForwardX11    | Specifies whether X11 connections are automatically redirected over the secure channel and DISPLAY set. The argument must be yes or no. The default is no.<br><br>X11 forwarding should be enabled with caution. Users with the ability to bypass file permissions on the remote host (for the user's X authorization database) can access the local X11 display through the forwarded connection. An attacker might then be able to perform activities such as keystroke monitoring. See the <code>ForwardX11Trusted</code> option for more information how to prevent this. |

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ForwardX11Trusted         | <p>If this option is set to yes, remote X11 clients have full access to the original X11 display. This option is set to yes by default.</p> <p>If this option is set to no, remote X11 clients are considered untrusted and prevented from stealing or tampering with data belonging to trusted X11 clients. Furthermore, the xauth(1) token used for the session is set to expire after 20 minutes. Remote clients are refused access after this time.</p> <p>See the X11 SECURITY extension specification for full details on the restrictions imposed on untrusted clients.</p> |
| GatewayPorts              | <p>Specifies whether remote hosts are allowed to connect to local forwarded ports. By default, ssh binds local port forwardings to the loopback address. This prevents other remote hosts from connecting to forwarded ports. GatewayPorts can be used to specify that ssh should bind local port forwardings to the wildcard address, thus allowing remote hosts to connect to forwarded ports. The argument must be yes or no. The default is no.</p>                                                                                                                            |
| GlobalKnownHostsFile      | <p>Specifies a file to use instead of /etc/ssh/ssh_known_hosts.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| GSSAPIAuthentication      | <p>Enables/disables GSS-API user authentication. The default is yes.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| GSSAPIDelegateCredentials | <p>Enables/disables GSS-API credential forwarding. The default is no.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| GSSAPIKeyExchange         | <p>Enables/disables GSS-API-authenticated key exchanges. The default is yes.</p> <p>This option is intended primarily to allow users to disable the use of GSS-API key exchange for SSHv2 when it would otherwise be selected and then fail (due to server misconfiguration, for example). SSHv2 key exchange failure always results in disconnection.</p> <p>This option also enables the use of the GSS-API to authenticate the user to the server after the key exchange. GSS-API key exchange can succeed but</p>                                                              |

---

|                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                         | <p>the subsequent authentication using the GSS-API fail if the server does not authorize the user's GSS principal name to the target user account.</p>                                                                                                                                                                                                                                                                                                                                                       |
| HashKnownHosts          | <p>Indicates that <code>ssh(1)</code>, should hash host names and addresses when they are added to <code>~/.ssh/known_hosts</code>. These hashed names can be used normally by <code>ssh(1)</code> and <code>sshd(1M)</code>, but they do not reveal identifying information should the file's contents be disclosed. The default is <code>no</code>. Existing names and addresses in known hosts files are not be converted automatically, but can be manually hashed using <code>ssh-keygen(1)</code>.</p> |
| Host                    | <p>Restricts the following declarations (up to the next Host keyword) to be only for those hosts that match one of the patterns given after the keyword. An asterisk (*) and a question mark (?) can be used as wildcards in the patterns. A single asterisk as a pattern can be used to provide global defaults for all hosts. The host is the host name argument given on the command line (that is, the name is not converted to a canonicalized host name before matching).</p>                          |
| HostbasedAuthentication | <p>Specifies whether to try <code>rhosts</code>-based authentication with public key authentication. The argument must be <code>yes</code> or <code>no</code>. The default is <code>no</code>. This option applies to protocol version 2 only and is similar to <code>RhostsRSAAuthentication</code>.</p>                                                                                                                                                                                                    |
| HostKeyAlgorithms       | <p>Specifies the protocol version 2 host key algorithms that the client wants to use in order of preference. The default for this option is: <code>ssh-rsa, ssh-dss</code>.</p>                                                                                                                                                                                                                                                                                                                              |
| HostKeyAlias            | <p>Specifies an alias that should be used instead of the real host name when looking up or saving the host key in the host key database files. This option is useful for tunneling <code>ssh</code> connections or for multiple servers running on a single host.</p>                                                                                                                                                                                                                                        |
| HostName                | <p>Specifies the real host name to log into. This can be used to specify nicknames or abbreviations for hosts. Default is the name given on the command</p>                                                                                                                                                                                                                                                                                                                                                  |

|                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                | <p>line. Numeric IP addresses are also permitted (both on the command line and in <code>HostName</code> specifications).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>IdentityFile</code>      | <p>Specifies a file from which the user's RSA or DSA authentication identity is read. The default is <code>\$HOME/.ssh/identity</code> for protocol version 1 and <code>\$HOME/.ssh/id_rsa</code> and <code>\$HOME/.ssh/id_dsa</code> for protocol version 2. Additionally, any identities represented by the authentication agent is used for authentication. The file name can use the tilde syntax to refer to a user's home directory. It is possible to have multiple identity files specified in configuration files; all these identities is tried in sequence.</p>                                                               |
| <code>IgnoreIfUnknown</code>   | <p>Specifies a comma-separated list of <code>ssh_config</code> parameters, which, if unknown to <code>ssh(1)</code>, are to be ignored by <code>ssh</code>.</p> <p>This parameter is primarily intended to be used in the per-user <code>ssh_config</code>, <code>~/.ssh/config</code>. While this parameter can also be used in the system wide <code>/etc/ssh/ssh_config</code> file, it is generally useless as the capabilities of the <code>ssh(1)</code> client on that host should match that file.</p>                                                                                                                           |
| <code>KeepAlive</code>         | <p>Specifies whether the system should send TCP keepalive messages to the other side. If they are sent, death of the connection or crash of one of the machines is properly noticed. However, this means that connections die if the route is down temporarily, which can be a source of annoyance.</p> <p>The default is <code>yes</code> (to send keepalives), which means the client notices if the network goes down or the remote host dies. This is important in scripts, and many users want it too. To disable keepalives, the value should be set to <code>no</code> in both the server and the client configuration files.</p> |
| <code>KMFPolicyDatabase</code> | <p>A filename for the KMF policy database. If not set, KMF defaults to <code>/etc/security/kmfpolicy.xml</code>. See the <code>Using X.509 Certificates</code> section in the <code>sshd(1M)</code> man page.</p>                                                                                                                                                                                                                                                                                                                                                                                                                        |

---

|                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| KMFPolicyName                    | A name of the KMF policy to be used. If not set, default is used. See the Using X.509 Certificates section in the <code>sshd(1M)</code> man page.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| LocalForward                     | Specifies that a TCP/IP port on the local machine be forwarded over the secure channel to a given <i>host:port</i> from the remote machine. The first argument must be <i>[bind_address:]port</i> and the second must be <i>host:port</i> . IPv6 addresses can be specified by enclosing addresses in square brackets or by using an alternative syntax: <i>[bind_address/]port</i> and <i>host/port</i> . Multiple forwardings can be specified and additional forwardings can be given on the command line. Only a user with enough privileges can forward privileged ports. By default, the local port is bound in accordance with the GatewayPorts setting. However, an explicit <i>bind_address</i> can be used to bind the connection to a specific address. The <i>bind_address</i> of <i>localhost</i> indicates that the listening port be bound for local use only, while an empty address or <i>*</i> indicates that the port should be available from all interfaces. |
| LogLevel                         | Gives the verbosity level that is used when logging messages from <code>ssh</code> . The possible values are: FATAL, ERROR, QUIET, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. The default is INFO. DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify higher levels of verbose output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| MACs                             | Specifies the MAC (message authentication code) algorithms in order of preference. The MAC algorithm is used in protocol version 2 for data integrity protection. Multiple algorithms must be comma-separated. The default is <code>hmac-md5,hmac-sha1,hmac-sha1-96,hmac-md5-96</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| NoHostAuthenticationForLocalhost | This option can be used if the home directory is shared across machines. In this case <code>localhost</code> refers to a different machine on each of the machines and the user gets many warnings about changed host keys. However, this option disables host authentication for <code>localhost</code> . The argument                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

|                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                       | to this keyword must be yes or no. The default is to check the host key for <code>localhost</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>NumberOfPasswordPrompts</code>  | Specifies the number of attempts before giving up for password and keyboard-interactive methods. Attempts for each method are counted separately. The argument to this keyword must be an integer. The default is 3.                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>PasswordAuthentication</code>   | Specifies whether to use password authentication. The argument to this keyword must be yes or no. This option applies to both protocol versions 1 and 2. The default is yes.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>Port</code>                     | Specifies the port number to connect on the remote host. The default is 22.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>PreferredAuthentications</code> | Specifies the order in which the client should try protocol 2 authentication methods. This allows a client to prefer one method (for example, <code>keyboard-interactive</code> ) over another method (for example, <code>password</code> ). The default for this option is:<br><code>hostbased,publickey,keyboard-interactive,password</code> .                                                                                                                                                                                                                                                                    |
| <code>Protocol</code>                 | Specifies the protocol versions <code>ssh</code> should support in order of preference. The possible values are 1 and 2. Multiple versions must be comma-separated. The default is 2, 1. This means that <code>ssh</code> tries version 2 and falls back to version 1 if version 2 is not available.<br><br>Support for the protocol version 1 might be dropped in a future release. See the <a href="#">ssh(1)</a> manual page for more information.                                                                                                                                                               |
| <code>ProxyCommand</code>             | Specifies the command to use to connect to the server. The command string extends to the end of the line, and is executed with <code>/bin/sh</code> . In the command string, <code>%h</code> is substituted by the host name to connect and <code>%p</code> by the port. The string can be any valid command, and should read from its standard input and write to its standard output. It should eventually connect an <a href="#">sshd(1M)</a> server running on some machine, or execute <code>sshd -i</code> somewhere. Host key management is done using the <code>HostName</code> of the host being connected |

---

|                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | <p>(defaulting to the name typed by the user).<br/>CheckHostIP is not available for connects with a proxy command.</p> <p>A special value of none can be used to indicate that for this Host section no proxy connect command should be used.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| PubkeyAuthentication | <p>Specifies whether to try public key authentication. The argument to this keyword must be yes or no. The default is yes. This option applies to protocol version 2 only.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| RekeyLimit           | <p>Specifies the maximum amount of data that can be transmitted before the session key is renegotiated. The argument is the number of bytes, with an optional suffix of K, M, or G to indicate Kilobytes, Megabytes, or Gigabytes, respectively. The default is between 1G and 4G, depending on the cipher. This option applies to protocol version 2 only.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| RemoteForward        | <p>Specifies that a TCP/IP port on the remote machine be forwarded over the secure channel to a given <i>host:port</i> from the local machine. The first argument must be [<i>bind_address</i>:]<i>port</i> and the second argument must be <i>host:port</i>. IPv6 addresses can be specified by enclosing addresses in square brackets or by using an alternative syntax: [<i>bind_address</i>/<i>port</i> and <i>host/port</i>. You can specify multiple forwardings and give additional forwardings on the command line. Only a user with enough privileges can forward privileged ports.</p> <p>If the <i>bind_address</i> is not specified, the default is to only bind to loopback addresses. If the <i>bind_address</i> is * or an empty string, then the forwarding is requested to listen on all interfaces. Specifying a remote <i>bind_address</i> only succeeds if the server's GatewayPorts option is enabled. See <a href="#">sshd_config(4)</a>.</p> |
| RhostsAuthentication | <p>Specifies whether to try rhosts-based authentication. This declaration affects only the client side and has no effect whatsoever on security. Disabling rhosts authentication can</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

|                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                      | reduce authentication time on slow connections when <code>rhhosts</code> authentication is not used. Most servers do not permit <code>RhostsAuthentication</code> because it is not secure (see <code>RhostsRSAAuthentication</code> ). The argument to this keyword must be <code>yes</code> or <code>no</code> . This option applies only to the protocol version 1 and requires that <code>ssh</code> be <code>setuid root</code> and that <code>UsePrivilegedPort</code> be set to <code>yes</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>RhostsRSAAuthentication</code> | Specifies whether to try <code>rhhosts</code> -based authentication with RSA host authentication. This is the primary authentication method for most sites. The argument must be <code>yes</code> or <code>no</code> . This option applies only to the protocol version 1 and requires that <code>ssh</code> be <code>setuid root</code> and that <code>UsePrivilegedPort</code> be set to <code>yes</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>ServerAliveCountMax</code>     | Sets the number of server alive messages which can be sent without <code>ssh(1)</code> receiving messages back from the server. If this threshold is reached while server alive messages are being sent, <code>ssh</code> disconnects from the server, terminating the session. The use of server alive messages differs from <code>TCPKeepAlive</code> . Server alive messages are sent through the encrypted channel and are not spoofable. The TCP keep alive option enabled by <code>TCPKeepAlive</code> is spoofable. The server alive mechanism is valuable when the client or server depend on knowing when a connection has become inactive.<br><br>The default value is 3. If, for example, <code>ServerAliveInterval</code> is set to 15 and <code>ServerAliveCountMax</code> is left at the default, <code>ssh</code> disconnects in 45-60 seconds if the server becomes unresponsive. This option applies to protocol version 2 only. |
| <code>ServerAliveInterval</code>     | Sets a timeout interval in seconds after which if no data has been received from the server, <code>ssh(1)</code> sends a message through the encrypted channel to request a response from the server. The default is                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |



---

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StrictHostKeyChecking | <p>0, indicating that these messages are not sent to the server. This option applies to protocol version 2 only.</p> <p>If this flag is set to yes, ssh never automatically adds host keys to the <code>\$HOME/.ssh/known_hosts</code> file, and refuses to connect hosts whose host key has changed. This provides maximum protection against trojan horse attacks. However, it can be a source of inconvenience if you do not have good <code>/etc/ssh/ssh_known_hosts</code> files installed and frequently connect new hosts. This option forces the user to manually add any new hosts. Normally this option is disabled, and new hosts are automatically added to the known host files. The host keys of known hosts are verified automatically in either case. The argument must be yes or no or ask. The default is ask.</p> |
| TrustedAnchorKeystore | <p>Specifies a directory where certificates of trusted anchors are located. Those certificates are used to validate host certificates if used as host keys.</p> <p>Currently only one level certificate chains are supported. This means that certificates must be signed by a private key that corresponds to a certificate located in the directory set by this option. Host key certificates can be self-signed as well. See the <a href="#">Using X.509 Certificates</a> section in the <a href="#">sshd(1M)</a> man page.</p>                                                                                                                                                                                                                                                                                                   |
| UseOpenSSL            | <p>Specifies whether ssh should use the OpenSSL PKCS#11 engine for offloading cryptographic operations to the Cryptographic Framework. Cryptographic operations are accelerated according to the available installed plug-ins. When no suitable plug-ins are present this option does not have an effect. The default is yes.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| UsePrivilegedPort     | <p>Specifies whether to use a privileged port for outgoing connections. The argument must be yes or no. The default is yes. Setting this option to no turns off <code>RhostsAuthentication</code> and <code>RhostsRSAAuthentication</code>. If set to yes ssh must be setuid root. Defaults to no.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

|                    |                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| User               | Specifies the user to log in as. This can be useful if you have different user names on different machines. This saves you the trouble of having to remember to enter the user name on the command line.                                                                                                                                                                 |
| UserKnownHostsFile | Specifies a file to use instead of <code>\$HOME/.ssh/known_hosts</code> .                                                                                                                                                                                                                                                                                                |
| UseRsh             | Specifies that <code>rlogin</code> or <code>rsh</code> should be used for this host. It is possible that the host does not support the <code>ssh</code> protocol. This causes <code>ssh</code> to immediately execute <a href="#">rsh(1)</a> . All other options (except <code>HostName</code> ) are ignored if this has been specified. The argument must be yes or no. |
| XAuthLocation      | Specifies the location of the <code>xauth(1)</code> program. The default is <code>/usr/openwin/bin/xauth</code> .                                                                                                                                                                                                                                                        |

**See Also** [rsh\(1\)](#), [ssh\(1\)](#), [ssh-http-proxy-connect\(1\)](#), [ssh-keygen\(1\)](#), [ssh-socks5-proxy-connect\(1\)](#), [sshd\(1M\)](#), [sshd\\_config\(4\)](#), [kerberos\(5\)](#)

*RFC 4252*

---

|                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>                     | sshd_config – sshd configuration file                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Synopsis</b>                 | /etc/ssh/sshd_config                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Description</b>              | <p>The <code>sshd(1M)</code> daemon reads configuration data from <code>/etc/ssh/sshd_config</code> (or the file specified with <code>sshd -f</code> on the command line). The file contains keyword-value pairs, one per line. A line starting with a hash mark (<code>#</code>) and empty lines are interpreted as comments.</p> <p>The <code>sshd_config</code> file supports the following keywords. Unless otherwise noted, keywords and their arguments are case-insensitive.</p>                                                                                                                         |
| <code>AllowGroups</code>        | <p>This keyword can be followed by a number of group names, separated by spaces. If specified, login is allowed only for users whose primary group or supplementary group list matches one of the patterns. Asterisk (<code>*</code>) and question mark (<code>?</code>) can be used as wildcards in the patterns. Only group names are valid; a numerical group ID is not recognized. By default, login is allowed regardless of the primary group.</p>                                                                                                                                                        |
| <code>AllowTcpForwarding</code> | <p>Specifies whether TCP forwarding is permitted. The default is yes. Disabling TCP forwarding does not improve security unless users are also denied shell access, as they can always install their own forwarders.</p>                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>AllowUsers</code>         | <p>This keyword can be followed by a number of user names, separated by spaces. If specified, login is allowed only for user names that match one of the patterns. Asterisk (<code>*</code>) and question mark (<code>?</code>) can be used as wildcards in the patterns. Only user names are valid; a numerical user ID is not recognized. By default login is allowed regardless of the user name.</p> <p>If a specified pattern takes the form <code>user@host</code> then <code>user</code> and <code>host</code> are checked separately, restricting logins to particular users from particular hosts.</p> |
| <code>AuthorizedKeysFile</code> | <p>Specifies the file that contains the public keys that can be used for user authentication.</p> <p><code>AuthorizedKeysFile</code> can contain tokens of the form <code>%T</code>, which are substituted during connection set-up. The following tokens are defined: <code>%%</code> is replaced by a literal <code>%</code>, <code>%h</code> is replaced by the home directory of the user being authenticated and <code>%u</code> is replaced by the username of that user. After</p>                                                                                                                       |

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                 | <p>expansion, <code>AuthorizedKeysFile</code> is taken to be an absolute path or one relative to the user's home directory. The default is <code>.ssh/authorized_keys</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Banner          | <p>In some jurisdictions, sending a warning message before authentication can be relevant for getting legal protection. The contents of the specified file are sent to the remote user before authentication is allowed. This option is only available for protocol version 2. By default, no banner is displayed.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| ChrootDirectory | <p>Specifies a path to <code>chroot(2)</code> to after authentication. This path, and all its components, must be root-owned directories that are not writable by any other user or group.</p> <p>The server always tries to change to the user's home directory locally under the chrooted environment but a failure to do so is not considered an error. In addition, the path might contain the following tokens that are expanded at runtime once the connecting user has been authenticated: <code>%%</code> is replaced by a literal <code>%</code>, <code>%h</code> is replaced by the home directory of the user being authenticated, and <code>%u</code> is replaced by the username of that user.</p> <p>The <code>ChrootDirectory</code> must contain the necessary files and directories to support the user's session. For an interactive SSH session this requires at least a user's shell, shared libraries needed by the shell, dynamic linker, and possibly basic <code>/dev</code> nodes such as <code>null</code>, <code>zero</code>, <code>stdin</code>, <code>stdout</code>, <code>stderr</code>, <code>random</code>, and <code>tty</code>. Additionally, terminal databases are needed for screen oriented applications. For file transfer sessions using <code>sftp</code> with the SSH protocol version 2, no additional configuration of the environment is necessary if the in-process <code>sftp</code> server is used. See <code>Subsystem</code> for details.</p> <p>The default is not to <code>chroot(2)</code>.</p> |
| Ciphers         | <p>Specifies the ciphers allowed for protocol version 2. Cipher ordering on the server side is not relevant. Multiple ciphers must be comma separated.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

Valid ciphers are: aes128-ctr, aes192-ctr, aes256-ctr, aes128-cbc, aes192-cbc, aes256-cbc, arcfour, arcfour128, arcfour256, 3des-cbc, and blowfish-cbc.

The default cipher list is:

```
aes128-ctr,aes192-ctr,aes256-ctr,arcfour128,
arcfour256,arcfour
```

Using CBC modes on the server side is not recommended due to potential security issues in connection with the SSH protocol version 2.

`ClientAliveCountMax`

Sets the number of client alive messages, (see `ClientAliveInterval`), that can be sent without `sshd` receiving any messages back from the client. If this threshold is reached while client alive messages are being sent, `sshd` disconnects the client, terminating the session. The use of client alive messages is very different from `KeepAlive`. The client alive messages are sent through the encrypted channel and therefore are not spoofable. The TCP `keepalive` option enabled by `KeepAlive` is spoofable. The client alive mechanism is valuable when a client or server depend on knowing when a connection has become inactive.

The default value is 3. If `ClientAliveInterval` is set to 15, and `ClientAliveCountMax` is left at the default, unresponsive `ssh` clients are disconnected after approximately 45 seconds.

`ClientAliveInterval`

Sets a timeout interval in seconds after which, if no data has been received from the client, `sshd` sends a message through the encrypted channel to request a response from the client. The default is 0, indicating that these messages are not sent to the client. This option applies only to protocol version 2.

`Compression`

Controls whether the server allows the client to negotiate the use of compression. The default is yes.

`DenyGroups`

Can be followed by a number of group names, separated by spaces. Users whose primary or supplementary group matches one of the patterns

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | <p>are not allowed to log in. Asterisk (*) and question mark (?) can be used as wildcards in the patterns. Only group names are valid; a numerical group ID is not recognized. By default, login is allowed regardless of the primary group.</p>                                                                                                                                                                                                                                                                                                                                                                                                                      |
| DenyUsers    | <p>Can be followed by a number of user names, separated by spaces. Login is disallowed for user names that match one of the patterns. Asterisk (*) and question mark (?) can be used as wildcards in the patterns. Only user names are valid; a numerical user ID is not recognized. By default, login is allowed regardless of the user name.</p> <p>If a specified pattern takes the form <i>user@host</i> then <i>user</i> and <i>host</i> are checked separately, disallowing logins to particular users from particular hosts.</p>                                                                                                                               |
| ForceCommand | <p>Forces the execution of the command specified by ForceCommand, ignoring any command supplied by the client, and, if present, <code>~/.ssh/rc</code>. The command is invoked by using the user's login shell with the <code>-c</code> option. This applies to shell, command, or subsystem execution. It is most useful inside a Match block. The command originally supplied by the client is available in the <code>SSH_ORIGINAL_COMMAND</code> environment variable. Specifying a command of <code>internal-sftp</code> forces the use of an in-process sftp server that requires no support files when used with <code>ChrootDirectory</code>.</p>              |
| GatewayPorts | <p>Specifies whether remote hosts are allowed to connect to ports forwarded for the client. By default, sshd binds remote port forwardings to the loopback address. This prevents other remote hosts from connecting to forwarded ports. GatewayPorts can be used to specify that sshd should bind remote port forwardings to the wildcard address, thus allowing remote hosts to connect to forwarded ports.</p> <p>The argument can be <code>no</code> to force remote port forwardings to be available to the local host only, <code>yes</code> to force remote port forwardings to bind to the wildcard address, or <code>clientspecified</code> to allow the</p> |

---

|                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                              | <p>client to select the address to which the forwarding is bound. The default is no. See also <code>RemoteForward</code> in <code>sshd_config(4)</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>GSSAPIAuthentication</code>            | <p>Enables/disables GSS-API user authentication. The default is yes.</p> <p>Currently <code>sshd</code> authorizes client user principals to user accounts as follows: if the principal name matches the requested user account, then the principal is authorized. Otherwise, GSS-API authentication fails.</p>                                                                                                                                                                                                                                                                                                                |
| <code>GSSAPIKeyExchange</code>               | <p>Enables/disables GSS-API-authenticated key exchanges. The default is yes.</p> <p>This option also enables the use of the GSS-API to authenticate the user to server after the key exchange. GSS-API key exchange can succeed but the subsequent authentication using the GSS-API fail if the server does not authorize the user's GSS principal name to the target user account.</p> <p>Currently <code>sshd</code> authorizes client user principals to user accounts as follows: if the principal name matches the requested user account, then the principal is authorized. Otherwise, GSS-API authentication fails.</p> |
| <code>GSSAPIStoreDelegatedCredentials</code> | <p>Enables/disables the use of delegated GSS-API credentials on the server-side. The default is yes.</p> <p>Specifically, this option, when enabled, causes the server to store delegated GSS-API credentials in the user's default GSS-API credential store (which for the Kerberos V mechanism means <code>/tmp/krb5cc_&lt;uid&gt;</code>).</p> <p><code>sshd</code> does not take any steps to explicitly destroy stored delegated GSS-API credentials upon logout. It is the responsibility of PAM modules to destroy credentials associated with a session.</p>                                                           |
| <code>HostbasedAuthentication</code>         | <p>Specifies whether to try <code>hosts</code>-based authentication with public key authentication. The argument must be yes or no. The default is no. This option applies to protocol version 2 only and is</p>                                                                                                                                                                                                                                                                                                                                                                                                               |

|                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                              | similar to <code>RhostsRSAAuthentication</code> . See <a href="#">ssh(1M)</a> for guidelines on setting up host-based authentication.                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>HostbasedUsesNameFromPacketOnly</code> | Controls which hostname is searched for in the files <code>~/.shosts</code> , <code>/etc/shosts.equiv</code> , and <code>/etc/hosts.equiv</code> . If this parameter is set to <code>yes</code> , the server uses the name the client claimed for itself and signed with that host's key. If set to <code>no</code> , the default, the server uses the name to which the client's IP address resolves.<br><br>Setting this parameter to <code>no</code> disables host-based authentication when using NAT or when the client gets to the server indirectly through a port-forwarding firewall. |
| <code>HostKey</code>                         | Specifies the file containing the private host key used by SSH. The default is <code>/etc/ssh/ssh_host_rsa_key</code> and <code>/etc/ssh/ssh_host_dsa_key</code> for protocol version 2. <code>ssh</code> refuses to use a file if it is group/world-accessible. It is possible to have multiple host key files. <code>dsa</code> or <code>rsa</code> keys are used for version 2 of the SSH protocol.                                                                                                                                                                                         |
| <code>IgnoreRhosts</code>                    | Specifies that <code>.rhosts</code> and <code>.shosts</code> files are not used in authentication. <code>/etc/hosts.equiv</code> and <code>/etc/shosts.equiv</code> are still used. The default is <code>yes</code> .                                                                                                                                                                                                                                                                                                                                                                          |
| <code>IgnoreUserKnownHosts</code>            | Specifies whether <code>ssh</code> should ignore the user's <code>\$HOME/.ssh/known_hosts</code> during <code>RhostsRSAAuthentication</code> . The default is <code>no</code> .                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>KbdInteractiveAuthentication</code>    | Specifies whether authentication by means of the keyboard-interactive authentication method is allowed. Defaults to <code>yes</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>KeepAlive</code>                       | Specifies whether the system should send <code>keepalive</code> messages to the other side. If they are sent, death of the connection or crash of one of the machines is properly noticed. However, this means that connections die if the route is down temporarily, which can be an annoyance. On the other hand, if <code>keepalives</code> are not sent, sessions can                                                                                                                                                                                                                      |



---

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   | <p>hang indefinitely on the server, leaving ghost users and consuming server resources.</p> <p>The default is yes (to send keepalives), and the server notices if the network goes down or the client host reboots. This avoids infinitely hanging sessions.</p> <p>To disable keepalives, the value should be set to no in both the server and the client configuration files.</p>                                                                                                                                                                                                                                                                                                   |
| KMFPolicyDatabase | <p>A filename for the KMF policy database. If not set, KMF defaults to <code>/etc/security/kmfpolicy.xml</code>. See the <a href="#">Using X.509 Certificates</a> section in the <a href="#">sshd(1M)</a> man page.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| KMFPolicyName     | <p>A name of the KMF policy to be used. If not set, default is used. See the <a href="#">Using X.509 Certificates</a> section in the <a href="#">sshd(1M)</a> man page.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| ListenAddress     | <p>Specifies what local address sshd should listen on. The following forms can be used:</p> <pre>ListenAddress host IPv4_addr IPv6_addr ListenAddress host IPv4_addr:port ListenAddress [host IPv6_addr]:port</pre> <p>If <i>port</i> is not specified, sshd listens on the address and all prior Port options specified. The default is to listen on all local addresses. Multiple ListenAddress options are permitted. Additionally, any Port options must precede this option for non-port qualified addresses.</p> <p>The default is to listen on all local addresses. Multiple options of this type are permitted. Additionally, the Ports options must precede this option.</p> |
| LoginGraceTime    | <p>The server disconnects after this time (in seconds) if the user has not successfully logged in. If the value is 0, there is no time limit. The default is 120 (seconds).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| LogLevel          | <p>Gives the verbosity level that is used when logging messages from sshd. The possible values are: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. The default is INFO. DEBUG2</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

---

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       | and DEBUG3 each specify higher levels of debugging output. Logging with level DEBUG violates the privacy of users and is not recommended.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| LookupClientHostnames | Specifies whether or not to reverse lookup the names of client's addresses. Setting this parameter to no can be useful where name resolution might be broken and thus cause sshd to spend a lot of time trying to resolve the client's IP address to a name. Defaults to yes. See VerifyReverseMapping.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| MACs                  | Specifies the available MAC (message authentication code) algorithms. The MAC algorithm is used in protocol version 2 for data integrity protection. Multiple algorithms must be comma-separated. The default is hmac-md5,hmac-sha1,hmac-sha1-96,hmac-md5-96.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Match                 | <p>Introduces a conditional block. If all of the criteria on the Match line are satisfied, the keywords on the following lines override those set in the global section of the config file, until either another Match line or the end of the file. Match blocks must be located at the end of the file, after all the global settings.</p> <p>The arguments to Match are one or more criteria-pattern pairs. The available criteria are User, Group, Host, and Address. The match patterns can consist of single entries or comma-separated lists and can use the wildcard (Asterisk * and question mark ?) and negation (!) operators.</p> <p>The patterns in a Host criteria should be hostname. The patterns in an Address criteria should be an IP address, which can additionally contain addresses to match in CIDR address/masklen format, for example, 192.0.2.0/24 or 2001:DB8::/32. The mask length provided must be consistent with the address - it is an error to specify a mask length that is too long for the address or one with bits set in this host portion of the address. For example, 192.0.2.0/33 and 192.0.2.0/8 respectively.</p> |

Only a subset of keywords can be used on the lines following a Match keyword. Available keywords are AllowTcpForwarding, AuthorizedKeysFile, Banner, ChrootDirectory, ForceCommand, GatewayPorts, GSSAPIAuthentication, HostbasedAuthentication, HostbasedUsesNameFromPacketOnly, KbdInteractiveAuth, MaxAuthTries, PasswordAuthentication, PermitEmptyPasswords, PermitRootLogin, PubkeyAuthentication, RhostsRSAAuthentication, RSAAuthentication, X11DisplayOffset, X11Forwarding, and X11UseLocalhost.

The following are four examples of using Match:

1. Disallowing user testuser to use TCP forwarding:

```
Match User testuser
 AllowTcpForwarding no
```

2. Displaying a special banner for users not in the staff group:

```
Match Group *,!staff
 Banner /etc/banner.text
```

3. Allowing root login from host rootallowed.example.com:

```
Match Host rootallowed.example.com
 PermitRootLogin yes
```

4. Allowing anyone to use GatewayPorts from the local net:

```
Match Address 192.168.0.0/24
 GatewayPorts yes
```

### MaxStartups

Specifies the maximum number of concurrent unauthenticated connections to the sshd daemon. Additional connections are dropped until authentication succeeds or the LoginGraceTime expires for a connection. The default is 10.

Alternatively, random early drop can be enabled by specifying the three colon-separated values *start:rate:full* (for example, 10:30:60). Referring

|                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                     | <p>to this example, <code>ssh</code> refuse connection attempts with a probability of <math>rate/100</math> (30% in our example) if there are currently 10 (from the <code>start</code> field) unauthenticated connections. The probability increases linearly and all connection attempts are refused if the number of unauthenticated connections reaches <i>full</i> (60 in our example).</p>                                                                                                                                                         |
| <code>PAMServiceName</code>         | <p>Specifies the PAM service name for the PAM session. The <code>PAMServiceName</code> and <code>PAMServicePrefix</code> options are mutually exclusive and if both set, <code>ssh</code> does not start. If this option is set the service name is the same for all user authentication methods. The option has no default value. See <code>PAMServicePrefix</code> for more information.</p>                                                                                                                                                           |
| <code>PAMServicePrefix</code>       | <p>Specifies the PAM service name prefix for service names used for individual user authentication methods. The default is <code>ssh</code>. The <code>PAMServiceName</code> and <code>PAMServicePrefix</code> options are mutually exclusive and if both set, <code>ssh</code> does not start.</p> <p>For example, if this option is set to <code>admincli</code>, the service name for the keyboard-interactive authentication method is <code>admincli-kbdint</code> instead of the default <code>ssh-kbdint</code>.</p>                              |
| <code>PasswordAuthentication</code> | <p>Specifies whether password authentication is allowed. The default is <code>yes</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>PermitEmptyPasswords</code>   | <p>When password or keyboard-interactive authentication is allowed, it specifies whether the server allows login to accounts with empty password strings.</p> <p>If not set then the <code>/etc/default/login</code> <code>PASSREQ</code> value is used instead.</p> <p><code>PASSREQ=no</code> is equivalent to <code>PermitEmptyPasswords yes</code>. <code>PASSREQ=yes</code> is equivalent to <code>PermitEmptyPasswords no</code>. If neither <code>PermitEmptyPasswords</code> or <code>PASSREQ</code> are set the default is <code>no</code>.</p> |
| <code>PermitRootLogin</code>        | <p>Specifies whether the root can log in using <code>ssh(1)</code>. The argument must be <code>yes</code>, <code>without-password</code>,</p>                                                                                                                                                                                                                                                                                                                                                                                                            |

forced-commands-only, or no. without-password means that root cannot be authenticated using the “password” or “keyboard-interactive” methods (see description of KbdInteractiveAuthentication). forced-commands-only means that authentication is allowed only for publickey (for SSHv2, or RSA, for SSHv1) and only if the matching authorized\_keys entry for root has a command=<cmd> option.

In Solaris, the default /etc/ssh/sshd\_config file is shipped with PermitRootLogin set to no. If unset by the administrator, then CONSOLE parameter from /etc/default/login supplies the default value as follows: if the CONSOLE parameter is not commented out (it can even be empty, that is, “CONSOLE=”), then without-password is used as default value. If CONSOLE is commented out, then the default for PermitRootLogin is yes.

The without-password and forced-commands-only settings are useful for, for example, performing remote administration and backups using trusted public keys for authentication of the remote client, without allowing access to the root account using passwords.

#### PermitUserEnvironment

Specifies whether a user's ~/.ssh/environment on the server side and environment options in the AuthorizedKeysFile file are processed by sshd. The default is no. Enabling environment processing can enable users to bypass access restrictions in some configurations using mechanisms such as LD\_PRELOAD.

Environment setting from a relevant entry in AuthorizedKeysFile file is processed only if the user was authenticated using the public key authentication method. Of the two files used, values of variables set in ~/.ssh/environment are of higher priority.

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PidFile         | Allows you to specify an alternative to <code>/var/run/sshd.pid</code> , the default file for storing the PID of the <code>sshd</code> listening for connections. See <a href="#">sshd(1M)</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Port            | Specifies the port number that <code>sshd</code> listens on. The default is 22. Multiple options of this type are permitted. See also <code>ListenAddress</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| PreUserauthHook | <p>Specifies an executable which is run prior to any of the processed authentication methods. The executable can be used to synchronize user information with a remote user-management facility using an arbitrary communication protocol.</p> <p>The executable is run before any user validation is conducted by <code>SSHD</code> so the user is not required to be existent before she tries to log in.</p> <p>The executable is invoked with two arguments in the following order: the name of the current authentication method and the username. The environment variable <code>SSH_CONNECTION</code> is also passed to the executable. If the executable returns a zero exit status, the current authentication method is processed as normal. See <a href="#">sshd(1M)</a>.</p> <p>If the exit status is 1, the current authentication method is ignored and can not be used to validate the user. The executable must be owned by root and have permissions of <code>0500</code>, otherwise it is treated as if it has exited with status 1.</p> <p>There is no default value for this property.</p> |
| PrintLastLog    | Specifies whether <code>sshd</code> should display the date and time when the user last logged in. The default is <code>yes</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| PrintMotd       | Specifies whether <code>sshd</code> should display the contents of <code>/etc/motd</code> when a user logs in interactively. (On some systems it is also displayed by the shell or a shell startup file, such as <code>/etc/profile</code> .) The default is <code>yes</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Protocol        | Specifies the protocol versions <code>sshd</code> should support in order of preference. The option is left for                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       | backwards compatibility and only value 2 is supported. The default is 2.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| PubkeyAuthentication  | Specifies whether public key authentication is allowed. The default is yes. This option applies to protocol version 2 only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| StrictModes           | Specifies whether sshd should check file modes and ownership of the user's files and home directory before accepting login. This is normally desirable because novices sometimes accidentally leave their directory or files world-writable. The default is yes.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Subsystem             | <p>Configures an external subsystem (for example, a file transfer daemon). Arguments should be a subsystem name and a command to execute upon subsystem request. The command <a href="#">sftp-server(1M)</a> implements the sftp file transfer subsystem.</p> <p>Alternately, the name <code>internal-sftp</code> implements an in-process sftp server. This can simplify configurations using <code>ChrootDirectory</code> to force a different filesystem root on clients.</p> <p>To specify an option to the internal sftp server, simply add the <code>sftp-server</code> option to the end of line. For example,</p> <pre>#sftp subsystem Subsystem sftp internal-sftp -u 002</pre> <p>By default, no subsystems are defined. This option applies to protocol version 2 only.</p> |
| TrustedAnchorKeystore | <p>Specifies a directory where certificates of trusted anchors are located. Those certificates are used to validate host certificates if used as host keys.</p> <p>Currently only one level certificate chains are supported. This means that certificates must be signed by a private key that corresponds to a certificate located in the directory set by this option. Host key certificates can be self-signed as well. See the <a href="#">Using X.509 Certificates</a> section in the <a href="#">sshd(1M)</a> man page.</p>                                                                                                                                                                                                                                                     |

|                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>SyslogFacility</code>       | <p>Gives the facility code that is used when logging messages from <code>sshd</code>. The possible values are: <code>DAEMON</code>, <code>USER</code>, <code>AUTH</code>, <code>LOCAL0</code>, <code>LOCAL1</code>, <code>LOCAL2</code>, <code>LOCAL3</code>, <code>LOCAL4</code>, <code>LOCAL5</code>, <code>LOCAL6</code>, and <code>LOCAL7</code>. The default is <code>AUTH</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>UseOpenSSLEngine</code>     | <p>Specifies whether <code>sshd</code> should use the OpenSSL PKCS#11 engine for offloading cryptographic operations to the Cryptographic Framework. Cryptographic operations are accelerated according to the available installed plug-ins. When no suitable plug-ins are present this option does not have an effect. The default is <code>yes</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>VerifyReverseMapping</code> | <p>Specifies whether <code>sshd</code> should try to verify the remote host name and check that the resolved host name for the remote IP address maps back to the very same IP address.</p> <p>A <code>yes</code> setting means <code>verify</code>. This feature is useful for Internet-facing servers. The default is <code>no</code>.</p> <p>The option is only usable if <code>LookupClientHostnames</code> is set to <code>yes</code>.</p>                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>X11DisplayOffset</code>     | <p>Specifies the first display number available for <code>sshd</code>'s X11 forwarding. This prevents <code>sshd</code> from interfering with real X11 servers. The default is 10.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>X11Forwarding</code>        | <p>Specifies whether X11 forwarding is permitted. The default is <code>yes</code>. Disabling X11 forwarding does not improve security in any way, as users can always install their own forwarders.</p> <p>When X11 forwarding is enabled, there can be additional exposure to the server and to client displays if the <code>sshd</code> proxy display is configured to listen on the wildcard address (see <code>X11UseLocalhost</code>). However, this is not the default. Additionally, the authentication spoofing and authentication data verification and substitution occur on the client side. The security risk of using X11 forwarding is that the client's X11 display server can be exposed to attack when the <code>ssh</code> client requests forwarding (see the warnings for <code>ForwardX11</code> in <code>ssh_config(4)</code>). A system</p> |



administrator who wants to protect clients that expose themselves to attack by unwittingly requesting X11 forwarding, should specify a no setting.

Disabling X11 forwarding does not prevent users from forwarding X11 traffic, as users can always install their own forwarders.

#### X11UseLocalhost

Specifies whether `sshd` should bind the X11 forwarding server to the loopback address or to the wildcard address. By default, `sshd` binds the forwarding server to the loopback address and sets the hostname part of the `DISPLAY` environment variable to `localhost`. This prevents remote hosts from connecting to the proxy display. However, some older X11 clients might not function with this configuration. `X11UseLocalhost` can be set to `no` to specify that the forwarding server should be bound to the wildcard address. The argument must be `yes` or `no`. The default is `yes`.

#### XAuthLocation

Specifies the location of the `xauth(1)` program. The default is `/usr/X11/bin/xauth` and `sshd` attempts to open it when X11 forwarding is enabled.

**Time Formats** `sshd` command-line arguments and configuration file options that specify time can be expressed using a sequence of the form: *time*[*qualifier*,] where *time* is a positive integer value and *qualifier* is one of the following:

|                                 |         |
|---------------------------------|---------|
| <code>&lt;none&gt;</code>       | seconds |
| <code>s</code>   <code>S</code> | seconds |
| <code>m</code>   <code>M</code> | minutes |
| <code>h</code>   <code>H</code> | hours   |
| <code>d</code>   <code>D</code> | days    |
| <code>w</code>                  | weeks   |

Each element of the sequence is added together to calculate the total time value. For example:

|                    |                                 |
|--------------------|---------------------------------|
| <code>600</code>   | 600 seconds (10 minutes)        |
| <code>10m</code>   | 10 minutes                      |
| <code>1h30m</code> | 1 hour, 30 minutes (90 minutes) |

**Files** /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.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTETYPE       | ATTRIBUTEVALUE |
|---------------------|----------------|
| Availability        | network/ssh    |
| Interface Stability | Uncommitted    |

**See Also** [login\(1\)](#), [sshd\(1M\)](#), [chroot\(2\)](#), [ssh\\_config\(4\)](#), [attributes\(5\)](#), [kerberos\(5\)](#)

**Authors** OpenSSH is a derivative of the original and free ssh 1.2.12 release by Tatu Ylonen. Aaron Campbell, Bob Beck, Markus Friedl, Niels Provos, Theo de Raadt, and Dug Song removed many bugs, re-added recent features, and created OpenSSH. Markus Friedl contributed the support for SSH protocol versions 1.5 and 2.0. Niels Provos and Markus Friedl contributed support for privilege separation.

**Name** su`log` – su command log file

**Synopsis** /var/adm/su`log`

**Description** The su`log` file is a record of all attempts by users on the system to execute the `su(1M)` command. Each time `su(1M)` is executed, an entry is added to the su`log` file.

Each entry in the su`log` file is a single line of the form:

```
SU date time
result port user-newuser
```

where

**date** The month and date `su(1M)` was executed. `date` is displayed in the form *mm/dd* where *mm* is the month number and *dd* is the day number in the month.

**time** The time `su(1M)` was executed. `time` is displayed in the form *HH/MM* where *HH* is the hour number (24 hour system) and *MM* is the minute number.

**result** The result of the `su(1M)` command. A '+' sign is displayed in this field if the su attempt was successful; otherwise a '-' sign is displayed.

**port** The name of the terminal device from which `su(1M)` was executed.

**user** The user id of the user executing the `su(1M)` command.

**newuser** The user id being switched to with `su(1M)`.

**Examples** EXAMPLE 1 A sample su`log` file.

Here is a sample su`log` file:

```
SU 02/25 09:29 + console root-sys
SU 02/25 09:32 + pts/3 user1-root
SU 03/02 08:03 + pts/5 user1-root
SU 03/03 08:19 + pts/5 user1-root
SU 03/09 14:24 - pts/5 guest3-root
SU 03/09 14:24 - pts/5 guest3-root
SU 03/14 08:31 + pts/4 user1-root
```

**Files** /var/adm/su`log` su log file  
 /etc/default/su contains the default location of su`log`

**See Also** `su(1M)`

**Name** sysbus, isa – device tree properties for ISA bus device drivers

**Description** Solaris for x86 supports the ISA bus as the system bus. Drivers for devices on this buse use the device tree built by the booting system to retrieve the necessary system resources used by the driver. These resources include device I/O port addresses, any interrupt capabilities that the device can have, any DMA channels it can require, and any memory-mapped addresses it can occupy.

Configuration files for ISA device drivers are only necessary to describe properties used by a particular driver that are not part of the standard properties found in the device tree. See [driver.conf\(4\)](#) for further details of configuration file syntax.

The ISA nexus drivers all belong to class sysbus. All bus drivers of class sysbus recognize the following properties:

- interrupts** An arbitrary-length array where each element of the array represents a hardware interrupt (IRQ) that is used by the device. In general, this array only has one entry unless a particular device uses more than one IRQ.
- Solaris defaults all ISA interrupts to IPL 5. This interrupt priority can be overridden by placing an `interrupt-priorities` property in a .conf file for the driver. Each entry in the array of integers for the `interrupt-priorities` property is matched one-to-one with the elements in the `interrupts` property to specify the IPL value that is used by the system for this interrupt in this driver. This is the priority that this device's interrupt handler receives relative to the interrupt handlers of other drivers. The priority is an integer from 1 to 16. Generally, disks are assigned a priority of 5, while mice and printers are lower, and serial communication devices are higher, typically 7. 10 is reserved by the system and must not be used. Priorities 11 and greater are high level priorities and are generally not recommended (see [ddi\\_intr\\_hilevel\(9F\)](#)).
- The driver can refer to the elements of this array by index using [ddi\\_add\\_intr\(9F\)](#). The index into the array is passed as the *inumber* argument of `ddi_add_intr()`.
- Only devices that generate interrupts have an `interrupts` property.
- reg** An arbitrary-length array where each element of the array consists of a 3-tuple of integers. Each array element describes a contiguous memory address range associated with the device on the bus.
- The first integer of the tuple specifies the memory type, 0 specifies a memory range and 1 specifies an I/O range. The second integer specifies the base address of the memory range. The third integer of each 3-tuple specifies the size, in bytes, of the mappable region.

The driver can refer to the elements of this array by index, and construct kernel mappings to these addresses using `ddi_map_regs(9F)`. The index into the array is passed as the `rnumber` argument of `ddi_map_regs()`.

All sysbus devices have `reg` properties. The first tuple of this property is used to construct the address part of the device name under `/devices`. In the case of `PLug` and `PLay` ISA devices, the first tuple is a special tuple that does not denote a memory range, but is used by the system only to create the address part of the device name. This special tuple can be recognized by determining if the top bit of the first integer is set to a one.

The order of the tuples in the `reg` property is determined by the boot system probe code and depends on the characteristics of each particular device. However, the `reg` property maintains the same order of entries from system boot to system boot. The recommended way to determine the `reg` property for a particular device is to use the `prtconf(1M)` command after installing the particular device. The output of the `prtconf` command can be examined to determine the `reg` property for any installed device.

You can use the `ddi_get*` and `ddi_put*` family of functions to access register space from a high-level interrupt context.

`dma-channels` A list of integers that specifies the DMA channels used by this device. Only devices that use DMA channels have a `dma-channels` property.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|----------------|-----------------|
| Architecture   | x86             |

**See Also** [prtconf\(1M\)](#), [driver.conf\(4\)](#), [scsi\(4\)](#), [attributes\(5\)](#), [ddi\\_add\\_intr\(9F\)](#), [ddi\\_intr\\_hilevel\(9F\)](#), [ddi\\_map\\_regs\(9F\)](#), [ddi\\_prop\\_op\(9F\)](#)

*Writing Device Drivers*

**Name** syslog.conf – configuration file for syslogd system log daemon

**Synopsis** /etc/syslog.conf

**Description** The file `/etc/syslog.conf` contains information used by the system log daemon, `syslogd(1M)`, to forward a system message to appropriate log files and/or users. `syslogd` preprocesses this file through `m4(1)` to obtain the correct information for certain log files, defining `LOGHOST` if the address of `loghost` is the same as one of the addresses of the host that is running `syslogd`.

A configuration entry is composed of two TAB-separated fields:

```
selector action
```

The *selector* field contains a semicolon-separated list of priority specifications of the form:

```
facility.level [; facility.level]
```

where *facility* is a system facility, or comma-separated list of facilities, and *level* is an indication of the severity of the condition being logged. Recognized values for *facility* include:

|                       |                                                                                                                                                                                                                  |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>user</code>     | Messages generated by user processes. This is the default priority for messages from programs or facilities not listed in this file.                                                                             |
| <code>kern</code>     | Messages generated by the kernel.                                                                                                                                                                                |
| <code>mail</code>     | The mail system.                                                                                                                                                                                                 |
| <code>daemon</code>   | System daemons.                                                                                                                                                                                                  |
| <code>auth</code>     | The authorization system: <code>login(1)</code> , <code>su(1M)</code> , <code>getty(1M)</code> , among others.                                                                                                   |
| <code>lpr</code>      | The line printer spooling system.                                                                                                                                                                                |
| <code>news</code>     | Designated for the USENET network news system.                                                                                                                                                                   |
| <code>uucp</code>     | Designated for the UUCP system; it does not currently use the <code>syslog</code> mechanism.                                                                                                                     |
| <code>cron</code>     | Designated for <code>cron/at</code> messages generated by systems that do logging through <code>syslog</code> . The current version of the Solaris Operating Environment does not use this facility for logging. |
| <code>audit</code>    | Designated for audit messages generated by systems that audit by means of <code>syslog</code> .                                                                                                                  |
| <code>local0-7</code> | Designated for local use.                                                                                                                                                                                        |
| <code>mark</code>     | For timestamp messages produced internally by <code>syslogd</code> .                                                                                                                                             |
| <code>*</code>        | An asterisk indicates all facilities except for the <code>mark</code> facility.                                                                                                                                  |

Recognized values for *level* are (in descending order of severity):

|                      |                                                                                                                                                                                                                                   |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>emerg</code>   | For panic conditions that would normally be broadcast to all users.                                                                                                                                                               |
| <code>alert</code>   | For conditions that should be corrected immediately, such as a corrupted system database.                                                                                                                                         |
| <code>crit</code>    | For warnings about critical conditions, such as hard device errors.                                                                                                                                                               |
| <code>err</code>     | For other errors.                                                                                                                                                                                                                 |
| <code>warning</code> | For warning messages.                                                                                                                                                                                                             |
| <code>notice</code>  | For conditions that are not error conditions, but may require special handling. A configuration entry with a <i>level</i> value of <code>notice</code> must appear on a separate line.                                            |
| <code>info</code>    | Informational messages.                                                                                                                                                                                                           |
| <code>debug</code>   | For messages that are normally used only when debugging a program.                                                                                                                                                                |
| <code>none</code>    | Do not send messages from the indicated <i>facility</i> to the selected file. For example, a <i>selector</i> of<br><br><code>*.debug;mail.none</code><br><br>sends all messages <i>except</i> mail messages to the selected file. |

For a given *facility* and *level*, `syslogd` matches all messages for that level and all higher levels. For example, an entry that specifies a level of `crit` also logs messages at the `alert` and `emerg` levels.

The *action* field indicates where to forward the message. Values for this field can have one of four forms:

- A filename, beginning with a leading slash, which indicates that messages specified by the *selector* are to be written to the specified file. The file is opened in append mode if it exists. If the file does not exist, logging silently fails for this action.
- The name of a remote host, prefixed with an @, as with: `@server`, which indicates that messages specified by the *selector* are to be forwarded to the `syslogd` on the named host. The hostname `loghost` is treated, in the default `syslog.conf`, as the hostname given to the machine that logs `syslogd` messages. Every machine is `loghost` by default, per the hosts database. It is also possible to specify one machine on a network to be `loghost` by, literally, naming the machine `loghost`. If the local machine is designated to be `loghost`, then `syslogd` messages are written to the appropriate files. Otherwise, they are sent to the machine `loghost` on the network.
- A comma-separated list of usernames, which indicates that messages specified by the *selector* are to be written to the named users if they are logged in.
- An asterisk, which indicates that messages specified by the *selector* are to be written to all logged-in users.

Blank lines are ignored. Lines for which the first nonwhite character is a '#' are treated as comments.

**Examples** EXAMPLE 1 A Sample Configuration File

With the following configuration file:

```
*.notice /var/log/notice
mail.info /var/log/notice
*.crit /var/log/critical
kern,mark.debug /dev/console
kern.err @server
*.emerg *
*.alert root,operator
*.alert;auth.warning /var/log/auth
```

`syslogd(1M)` logs all mail system messages except debug messages and all notice (or higher) messages into a file named `/var/log/notice`. It logs all critical messages into `/var/log/critical`, and all kernel messages and 20-minute marks onto the system console.

Kernel messages of `err` (error) severity or higher are forwarded to the machine named `server`. Emergency messages are forwarded to all users. The users `root` and `operator` are informed of any `alert` messages. All messages from the authorization system of `warning` level or higher are logged in the file `/var/log/auth`.

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Committed       |

**See Also** [at\(1\)](#), [crontab\(1\)](#), [logger\(1\)](#), [login\(1\)](#), [m4\(1\)](#), [cron\(1M\)](#), [getty\(1M\)](#), [su\(1M\)](#), [syslogd\(1M\)](#), [syslog\(3C\)](#), [hosts\(4\)](#), [attributes\(5\)](#)



**Name** system – system configuration information file

**Description** The `system` file is used for customizing the operation of the operating system kernel. The recommended procedure is to preserve the original `system` file before modifying it.

The `system` file contains commands which are read by the kernel during initialization and used to customize the operation of your system. These commands are useful for modifying the system's treatment of its loadable kernel modules.

The syntax of the `system` file consists of a list of keyword/value pairs which are recognized by the system as valid commands. Comment lines must begin with an asterisk (\*) or a hash mark (#) and end with a newline character. All commands are case-insensitive except where noted.

Commands that modify the system's operation with respect to loadable kernel modules require you to specify the module type by listing the module's namespace. The following namespaces are currently supported on all platforms:

`drv` Modules in this namespace are device drivers.

`exec` Modules in this namespace are execution format modules. The following `exec` modules are currently provided:

Only on SPARC system:

`aoutexec`

Only on x86 system:

`coffexec`

On SPARC and IA systems:

`elfexec`

`intpexec`

`javaexec`

`fs` These modules are filesystems.

`sched` These modules implement a process scheduling algorithm.

`strmod` These modules are STREAMS modules.

`sys` These modules implement loadable system-call modules.

`misc` These modules do not fit into any of the above categories, so are considered "miscellaneous" modules.

SPARC only:

`dacf` These modules provide rules and actions for device auto-configuration.

`tod` These modules provide support for the time of day hardware.

`cpu` These modules provide CPU-specific kernel routines.

A description of each of the supported commands follows:

|                                                                                    |                                                                                                                                                                                                                                                                                                              |
|------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>exclude: &lt;namespace&gt;/&lt;modulename&gt;</code>                         | Do not allow the listed loadable kernel module to be loaded. <code>exclude</code> commands are cumulative; the list of modules to <code>exclude</code> is created by combining every <code>exclude</code> entry in the <code>system</code> file.                                                             |
| <code>include: &lt;namespace&gt;/&lt;modulename&gt;</code>                         | Include the listed loadable kernel module. This is the system's default, so using <code>include</code> does not modify the system's operation. <code>include</code> commands are cumulative.                                                                                                                 |
| <code>forceload: &lt;namespace&gt;/&lt;modulename&gt;</code>                       | Force this kernel module to be loaded during kernel initialization. The default action is to automatically load the kernel module when its services are first accessed. <code>forceload</code> commands are cumulative.                                                                                      |
| <code>rootdev: &lt;device name&gt;</code>                                          | Set the root device to the listed value instead of using the default root device as supplied by the boot program.                                                                                                                                                                                            |
| <code>rootfs: &lt;root filesystem type&gt;</code>                                  | Set the root filesystem type to the listed value.                                                                                                                                                                                                                                                            |
| <code>moddir: &lt;first module path&gt;[[{; }&lt;second ...&gt;]...]</code>        | Set the search path for loadable kernel modules. This command operates very much like the <code>PATH</code> shell variable. Multiple directories to search can be listed together, delimited either by blank spaces or colons.                                                                               |
| <code>set [&lt;module&gt;:]&lt;symbol&gt; {=,  , &amp;} [~][-]&lt;value&gt;</code> | Set an integer or character pointer in the kernel or in the selected kernel module to a new value. This command is used to change kernel and module parameters and thus modify the operation of your system. Assignment operations are not cumulative, whereas bitwise AND and OR operations are cumulative. |

Operations that are supported for modifying integer variables are: simple assignment, inclusive bitwise OR, bitwise AND, one's complement, and negation. Variables in a specific loadable module can be targeted for modification by specifying the variable name prefixed with the kernel module name and a colon (:) separator. Values can be specified as hexadecimal (0x10), Octal (046), or Decimal (5).

The only operation supported for modifying character pointers is simple assignment. Static string data such as character arrays cannot be modified using the set command. Use care and ensure that the variable you are modifying is in fact a character pointer. The set command is very powerful, and will likely cause problems if used carelessly. The following escape sequences are supported within the quoted string:

```
\n (newline)
\t (tab)
\b (backspace)
```

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

The following is a sample system file.

```
* Force the ELF exec kernel module to be loaded during kernel
* initialization. Execution type modules are in the exec namespace.
forceload: exec/elfexec
* Change the root device to /sbus@1,f8000000/esp@0,800000/sd@3,0:a.
* You can derive root device names from /devices.
* Root device names must be the fully expanded Open Boot Prom
* device name. This command is platform and configuration specific.
* This example uses the first partition (a) of the SCSI disk at
* SCSI target 3 on the esp host adapter in slot 0 (on board)
* of the SBus of the machine.
* Adapter unit-address 3,0 at sbus unit-address 0,800000.
rootdev: /sbus@1,f8000000/esp@0,800000/sd@3,0:a
* Set the filesystem type of the root to ufs. Note that
* the equal sign can be used instead of the colon.
```

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

```
rootfs:ufs
* Set the search path for kernel modules to look first in
* /usr/phil/mod_test for modules, then in /kernel/modules (the
* default) if not found. Useful for testing new modules.
* Note that you can delimit your module pathnames using
* colons instead of spaces: moddir:/newmodules:/kernel/modules
moddir:/usr/phil/mod_test /kernel/modules.
* Set the configuration option {_POSIX_CHOWN_RESTRICTED} :
* This configuration option is enabled by default.
set rstchown = 1
* Disable the configuration option {_POSIX_CHOWN_RESTRICTED} :
set rstchown = 0
* Turn on debugging messages in the modules mydriver. This is useful
* during driver development.
set mydriver:debug = 1
* Bitwise AND the kernel variable "moddebug" with the
* one's complement of the hex value 0x880, and set
* "moddebug" to this new value.
set moddebug & ~0x880
* Demonstrate the cumulative effect of the SET
* bitwise AND/OR operations by further modifying "moddebug"
* by ORing it with 0x40.
set moddebug | 0x40
```

**See Also** [boot\(1M\)](#), [init\(1M\)](#), [kernel\(1M\)](#)

**Warnings** Use care when modifying the system file; it modifies the operation of the kernel. If you preserved the original system file, you can boot using `boot -a`, which will ask you to specify the path to the saved file. This should allow the system to boot correctly. If you cannot locate a system file that will work, you may specify `/dev/null`. This acts as an empty system file, and the system will attempt to boot using its default settings.

**Notes** The `/etc/system` file is read only once, at boot time.

**Name** telnetrc – file for telnet default options

**Description** The `.telnetrc` file contains commands that are executed when a connection is established on a per-host basis. Each line in the file contains a host name, one or more spaces or tabs, and a [telnet\(1\)](#) command. The host name, `DEFAULT`, matches all hosts. Lines beginning with the pound sign (`#`) are interpreted as comments and therefore ignored. [telnet\(1\)](#) commands are case-insensitive to the contents of the `.telnetrc` file.

The `.telnetrc` file is retrieved from each user's HOME directory.

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

In the following example, a `.telnetrc` file executes the [telnet\(1\)](#) command, `toggle`:

```
weirdhost toggle crmod
Always export $PRINTER
DEFAULT environ export PRINTER
```

The lines in this file indicate that the `toggle` argument `crmod`, whose default value is “off” (or `FALSE`), should be enabled when connecting to the system `weirdhost`. In addition, the value of the environment variable `PRINTER` should be exported to all systems. In this case, the `DEFAULT` keyword is used in place of the host name.

**Files** `$HOME/.telnetrc` file for telnet default options

**See Also** [telnet\(1\)](#), [in.telnetd\(1M\)](#), [environ\(5\)](#)

**Name** term – format of compiled term file

**Synopsis** /usr/share/lib/terminfo/?/\*

**Description** The term file is compiled from [terminfo\(4\)](#) source files using [tic\(1M\)](#). Compiled files are organized in a directory hierarchy under the first letter of each terminal name. For example, the vt100 file would have the pathname /usr/lib/terminfo/v/vt100. The default directory is /usr/share/lib/terminfo. Synonyms for the same terminal are implemented by multiple links to the same compiled file.

The format has been chosen so that it is the same on all hardware. An 8-bit byte is assumed, but no assumptions about byte ordering or sign extension are made. Thus, these binary terminfo files can be transported to other hardware with 8-bit bytes.

Short integers are stored in two 8-bit bytes. The first byte contains the least significant 8 bits of the value, and the second byte contains the most significant 8 bits. (Thus, the value represented is  $256 * \text{second} + \text{first}$ .) The value  $-1$  is represented by `0377, 0377`, and the value  $-2$  is represented by `0376, 0377`; other negative values are illegal. The  $-1$  generally means that a capability is missing from this terminal. The  $-2$  means that the capability has been cancelled in the terminfo source and also is to be considered missing.

The compiled file is created from the source file descriptions of the terminals (see the `-I` option of `infocmp`) by using the terminfo compiler, `tic`, and read by the routine `setupterm` (see [curses\(3CURSES\)](#)). The file is divided into six parts in the following order: the header, terminal names, boolean flags, numbers, strings, and string table.

The header section begins the file six short integers in the format described below. These integers are:

1. the magic number (octal `0432`);
2. the size, in bytes, of the names section;
3. the number of bytes in the boolean section
4. the number of short integers in the numbers section;
5. the number of offsets (short integers) in the strings section;
6. the size, in bytes, of the string table.

The terminal name section comes next. It contains the first line of the terminfo description, listing the various names for the terminal, separated by the bar (|) character (see [term\(5\)](#)). The section is terminated with an ASCII NUL character.

The terminal name section is followed by the Boolean section, number section, string section, and string table.

The boolean flags section consists of one byte for each flag. This byte is either `0` or `1` as the flag is present or absent. The value of `2` means that the flag has been cancelled. The capabilities are in the same order as the file `<term.h>`.

Between the boolean flags section and the number section, a null byte is inserted, if necessary, to ensure that the number section begins on an even byte offset. All short integers are aligned on a short word boundary.

The numbers section is similar to the boolean flags section. Each capability takes up two bytes, and is stored as a short integer. If the value represented is  $-1$  or  $-2$ , the capability is taken to be missing.

The strings section is also similar. Each capability is stored as a short integer, in the format above. A value of  $-1$  or  $-2$  means the capability is missing. Otherwise, the value is taken as an offset from the beginning of the string table. Special characters in  $\^X$  or  $\c$  notation are stored in their interpreted form, not the printing representation. Padding information ( $\$<nn>$ ) and parameter information ( $\%x$ ) are stored intact in uninterpreted form.

The final section is the string table. It contains all the values of string capabilities referenced in the string section. Each string is null terminated.

Note that it is possible for `setupterm` to expect a different set of capabilities than are actually present in the file. Either the database may have been updated since `setupterm` has been recompiled (resulting in extra unrecognized entries in the file) or the program may have been recompiled more recently than the database was updated (resulting in missing entries). The routine `setupterm` must be prepared for both possibilities—this is why the numbers and sizes are included. Also, new capabilities must always be added at the end of the lists of boolean, number, and string capabilities.

As an example, here is terminal information on the AT&T Model 37 KSR terminal as output by the `infocmp -I tty37` command:

```
37|tty37|AT&T model 37 teletype,
 hc, os, xon,
 bel=^G, cr=\r, cub1=\b, cud1=\n, cuu1=\E7, hd=\E9,
 hu=\E8, ind=\n,
```

The following is an octal dump of the corresponding `term` file, produced by the `od -c /usr/share/lib/terminfo/t/tty37` command:

```
0000000 032 001 \0 032 \0 013 \0 021 001 3 \0 3 7 | t
0000020 t y 3 7 | A T & T m o d e l
0000040 3 7 t e l e t y p e \0 \0 \0 \0 \0
0000060 \0 \0 \0 001 \0 \0 \0 \0 \0 \0 \0 001 \0 \0 \0 \0
0000100 001 \0 \0 \0 \0 \0 377 377 377 377 377 377 377 377 377
0000120 377 377 377 377 377 377 377 377 377 377 377 377 377 377
0000140 \0 377 377 377 377 377 377 377 377 377 377 377 377 377
0000160 377 377 " \0 377 377 377 377 (\0 377 377 377 377 377
0000200 377 377 0 \0 377 377 377 377 377 377 377 377 - \0 377 377
0000220 377 377 377 377 377 377 377 377 377 377 377 377 377 377
*
```

```
0000520 377 $ \0
0000540 377 * \0
0000560 377 *
*
0001160 377 3 7
0001200 | t t y 3 7 | A T & T m o d e
0001220 l 3 7 t e l e t y p e \0 \r \0
0001240 \n \0 \n \0 007 \0 \b \0 033 8 \0 033 9 \0 033 7
0001260 \0 \0
0001261
```

Some limitations: total compiled entries cannot exceed 4096 bytes; all entries in the name field cannot exceed 128 bytes.

**Files** /usr/share/lib/terminfo/?/\* compiled terminal description database  
/usr/include/term.h terminfo header  
/usr/xpg4/include/term.h X/Open Curses terminfo header

**See Also** [infocmp\(1M\)](#), [curses\(3CURSES\)](#), [curses\(3XCURSES\)](#), [terminfo\(4\)](#), [term\(5\)](#)



**Name** terminfo – terminal and printer capability database

**Synopsis** /usr/share/lib/terminfo/?/\*

**Description** The terminfo database describes the capabilities of devices such as terminals and printers. Devices are described in terminfo source files by specifying a set of capabilities, by quantifying certain aspects of the device, and by specifying character sequences that affect particular results. This database is often used by screen oriented applications such as vi and curses-based programs, as well as by some system commands such as ls and more. This usage allows them to work with a variety of devices without changes to the programs.

terminfo descriptions are located in the directory pointed to by the environment variable TERMINFO or in /usr/share/lib/terminfo. terminfo descriptions are generated by [tic\(1M\)](#).

terminfo source files consist of one or more device descriptions. Each description consists of a header (beginning in column 1) and one or more lines that list the features for that particular device. Every line in a terminfo source file must end in a comma (,). Every line in a terminfo source file except the header must be indented with one or more white spaces (either spaces or tabs).

Entries in terminfo source files consist of a number of comma-separated fields. White space after each comma is ignored. Embedded commas must be escaped by using a backslash. Each device entry has the following format:

```
alias1 | alias2 | . . . | aliasn | fullname,
 capability1, capability2,
 .
 .
 .
 capabilityn,
```

The first line, commonly referred to as the header line, must begin in column one and must contain at least two aliases separated by vertical bars. The last field in the header line must be the long name of the device and it may contain any string. Alias names must be unique in the terminfo database and they must conform to system file naming conventions. See [tic\(1M\)](#). They cannot, for example, contain white space or slashes.

Every device must be assigned a name, such as “vt100”. Device names (except the long name) should be chosen using the following conventions. The name should not contain hyphens because hyphens are reserved for use when adding suffixes that indicate special modes.

These special modes may be modes that the hardware can be in, or user preferences. To assign a special mode to a particular device, append a suffix consisting of a hyphen and an indicator of the mode to the device name. For example, the -w suffix means “wide mode”. When specified, it allows for a width of 132 columns instead of the standard 80 columns. Therefore, if you want to use a “vt100” device set to wide mode, name the device “vt100-w”. Use the following suffixes where possible.

| Suffix | Meaning                              | Example   |
|--------|--------------------------------------|-----------|
| -w     | Wide mode (more than 80 columns)     | 5410-w    |
| -am    | With auto. margins (usually default) | vt100-am  |
| -nam   | Without automatic margins            | vt100-nam |
| -n     | Number of lines on the screen        | 2300-40   |
| -na    | No arrow keys (leave them in local)  | c100-na   |
| -np    | Number of pages of memory            | c100-4p   |
| -rv    | Reverse video                        | 4415-rv   |

The terminfo reference manual page is organized in two sections:

- PART 1: DEVICE CAPABILITIES
- PART 2: PRINTER CAPABILITIES

#### PART 1: DEVICE CAPABILITIES

Capabilities in terminfo are of three types: Boolean capabilities (which show that a device has or does not have a particular feature), numeric capabilities (which quantify particular features of a device), and string capabilities (which provide sequences that can be used to perform particular operations on devices).

In the following table, a `Variable` is the name by which a C programmer accesses a capability (at the terminfo level). A `Capname` is the short name for a capability specified in the terminfo source file. It is used by a person updating the source file and by the `tput` command. A `Termcap Code` is a two-letter sequence that corresponds to the termcap capability name. (Note that termcap is no longer supported.)

Capability names have no real length limit, but an informal limit of five characters has been adopted to keep them short. Whenever possible, capability names are chosen to be the same as or similar to those specified by the ANSI X3.64-1979 standard. Semantics are also intended to match those of the ANSI standard.

All string capabilities listed below may have padding specified, with the exception of those used for input. Input capabilities, listed under the `Strings` section in the following tables, have names beginning with `key_`. The `#i` symbol in the description field of the following tables refers to the *i*th parameter.

#### Booleans

| Variable                      | Cap-name        | Termcap Code    | Description                              |
|-------------------------------|-----------------|-----------------|------------------------------------------|
| <code>auto_left_margin</code> | <code>bw</code> | <code>bw</code> | <code>cub1</code> wraps from column 0 to |

|                          |       |    |                                                   |
|--------------------------|-------|----|---------------------------------------------------|
|                          |       |    | last column                                       |
| auto_right_margin        | am    | am | Terminal has automatic margins                    |
| back_color_erase         | bce   | be | Screen erased with background color               |
| can_change               | ccc   | cc | Terminal can re-define existing color             |
| ceol_standout_glitch     | xhp   | xs | Standout not erased by overwriting (hp)           |
| col_addr_glitch          | xhpa  | YA | Only positive motion for hpa/mhpa caps            |
| cpi_changes_res          | cpix  | YF | Changing character pitch changes resolution       |
| cr_cancels_micro_mode    | crxm  | YB | Using cr turns off micro mode                     |
| dest_tabs_magic_smo      | xt    | xt | Destructive tabs, magic smo char (t1061)          |
| eat_newline_glitch       | xenl  | xn | Newline ignored after 80 columns (Concept)        |
| erase_overstrike         | eo    | eo | Can erase overstrikes with a blank                |
| generic_type             | gn    | gn | Generic line type (for example, dialup, switch)   |
| hard_copy                | hc    | hc | Hardcopy terminal                                 |
| hard_cursor              | chts  | HC | Cursor is hard to see                             |
| has_meta_key             | km    | km | Has a meta key (shift, sets parity bit)           |
| has_print_wheel          | daisy | YC | Printer needs operator to change character set    |
| has_status_line          | hs    | hs | Has extra "status line"                           |
| hue_lightness_saturation | hls   | hl | Terminal uses only HLS color notation (Tektronix) |
| insert_null_glitch       | in    | in | Insert mode distinguishes nulls                   |
| lpi_changes_res          | lpix  | YG | Changing line pitch changes resolution            |
| memory_above             | da    | da | Display may be retained above the screen          |
| memory_below             | db    | db | Display may be retained below the screen          |
| move_insert_mode         | mir   | mi | Safe to move while in insert mode                 |
| move_standout_mode       | msgr  | ms | Safe to move in standout modes                    |
| needs_xon_xoff           | nxon  | nx | Padding won't work, xon/xoff required             |
| no_esc_ctlc              | xsb   | xb | Beehive (f1=escape, f2=ctrl C)                    |
| no_pad_char              | npc   | NP | Pad character doesn't exist                       |
| non_dest_scroll_region   | ndscr | ND | Scrolling region is nondestructive                |
| non_rev_rmcup            | nrrmc | NR | smcup does not reverse rmcup                      |
| over_strike              | os    | os | Terminal overstrikes                              |

|                        |       |    |                                           |
|------------------------|-------|----|-------------------------------------------|
|                        |       |    | on hard-copy terminal                     |
| prtr_silent            | mc5i  | 5i | Printer won't echo on screen              |
| row_addr_glitch        | xvpa  | YD | Only positive motion<br>for vpa/mvpa caps |
| semi_auto_right_margin | sam   | YE | Printing in last column causes<br>cr      |
| status_line_esc_ok     | eslok | es | Escape can be used on<br>the status line  |
| tilde_glitch           | hz    | hz | Hazeltine; can't print tilde (~)          |
| transparent_underline  | ul    | ul | Underline character overstrikes           |
| xon_xoff               | xon   | xo | Terminal uses xon/xoff<br>handshaking     |

## Numbers

| Variable            | Cap-name | Termcap Code | Description                                               |
|---------------------|----------|--------------|-----------------------------------------------------------|
| bit_image_entwining | bitwin   | Yo           | Number of passes for each<br>bit-map row                  |
| bit_image_type      | bitype   | Yp           | Type of bit image device                                  |
| buffer_capacity     | bufsz    | Ya           | Number of bytes buffered<br>before printing               |
| buttons             | btms     | BT           | Number of buttons on the mouse                            |
| columns             | cols     | co           | Number of columns in a line                               |
| dot_horz_spacing    | spinh    | Yc           | Spacing of dots horizontally<br>in dots per inch          |
| dot_vert_spacing    | spinv    | Yb           | Spacing of pins vertically<br>in pins per inch            |
| init_tabs           | it       | it           | Tabs initially every # spaces                             |
| label_height        | lh       | lh           | Number of rows in each label                              |
| label_width         | lw       | lw           | Number of columns in each label                           |
| lines               | lines    | li           | Number of lines on a screen or<br>a page                  |
| lines_of_memory     | lm       | lm           | Lines of memory if > lines;<br>0 means varies             |
| max_attributes      | ma       | ma           | Maximum combined video attributes<br>terminal can display |
| magic_cookie_glitch | xmc      | sg           | Number of blank characters<br>left by smso or rmso        |
| max_colors          | colors   | Co           | Maximum number of colors<br>on the screen                 |
| max_micro_address   | maddr    | Yd           | Maximum value in<br>micro..._address                      |
| max_micro_jump      | mjump    | Ye           | Maximum value in parm..._micro                            |
| max_pairs           | pairs    | pa           | Maximum number of<br>color-pairs on the screen            |

|                      |       |    |                                                             |
|----------------------|-------|----|-------------------------------------------------------------|
| maximum_windows      | Wnum  | MW | Maximum number of definable windows                         |
| micro_char_size      | mcs   | Yf | Character step size when<br>in micro mode                   |
| micro_line_size      | mls   | Yg | Line step size when in micro mode                           |
| no_color_video       | ncv   | NC | Video attributes that<br>can't be used with colors          |
| num_labels           | nlab  | NL | Number of labels on screen                                  |
| number_of_pins       | npins | Yh | Number of pins in print-head                                |
| output_res_char      | orc   | Yi | Horizontal resolution in<br>units per character             |
| output_res_line      | orl   | Yj | Vertical resolution in units per<br>line                    |
| output_res_horz_inch | orhi  | Yk | Horizontal resolution in<br>units per inch                  |
| output_res_vert_inch | orvi  | Yl | Vertical resolution in<br>units per inch                    |
| padding_baud_rate    | pb    | pb | Lowest baud rate                                            |
| print_rate           | cps   | Ym | Print rate in characters per second<br>where padding needed |
| virtual_terminal     | vt    | vt | Virtual terminal number (system)                            |
| wide_char_size       | widcs | Yn | Character step size when<br>in double wide mode             |
| width_status_line    | wsl   | ws | Number of columns in status line                            |

### Strings

| Variable                  | Cap-<br>name | Termcap<br>Code | Description                                                          |
|---------------------------|--------------|-----------------|----------------------------------------------------------------------|
| acs_chars                 | acsc         | ac              | Graphic charset pairs aAbBcC                                         |
| alt_scancode_esc          | scesa        | S8              | Alternate escape for<br>scancode emulation<br>(default is for vt100) |
| back_tab                  | cbt          | bt              | Back tab                                                             |
| bell                      | bel          | bl              | Audible signal (bell)                                                |
| bit_image_carriage_return | bicr         | Yv              | Move to beginning of<br>same row (use tparm)                         |
| bit_image_newline         | binel        | Zz              | Move to next row of<br>the bit image (use tparm)                     |
| bit_image_repeat          | birep        | Zy              | Repeat bit-image cell<br>#1 #2 times (use tparm)                     |
| carriage_return           | cr           | cr              | Carriage return                                                      |
| change_char_pitch         | cpi          | ZA              | Change number of<br>characters per inch                              |
| change_line_pitch         | lpi          | ZB              | Change number of lines per inch                                      |
| change_res_horz           | chr          | ZC              | Change horizontal resolution                                         |
| change_res_vert           | cvr          | ZD              | Change vertical resolution                                           |

---

|                         |        |    |                                                  |
|-------------------------|--------|----|--------------------------------------------------|
| change_scroll_region    | csr    | cs | Change to lines #1 through #2 (vt100)            |
| char_padding            | rmp    | rP | Like ip but when in replace mode                 |
| char_set_names          | csnm   | Zy | List of character set names                      |
| clear_all_tabs          | tbc    | ct | Clear all tab stops                              |
| clear_margins           | mgc    | MC | Clear all margins (top, bottom, and sides)       |
| clear_screen            | clear  | cl | Clear screen and home cursor                     |
| clr_bol                 | ell    | cb | Clear to beginning of line, inclusive            |
| clr_eol                 | el     | ce | Clear to end of line                             |
| clr_eos                 | ed     | cd | Clear to end of display                          |
| code_set_init           | csin   | ci | Init sequence for multiple codesets              |
| color_names             | colorm | Yw | Give name for color #1                           |
| column_address          | hpa    | ch | Horizontal position                              |
| command_character       | cmdch  | CC | Terminal settable cmd character in prototype     |
| create_window           | cwin   | CW | Define win #1 to go from #2,#3to #4,#5           |
| cursor_address          | cup    | cm | Move to row #1 col #2                            |
| cursor_down             | cudl   | do | Down one line                                    |
| cursor_home             | home   | ho | Home cursor (if no cup)                          |
| cursor_invisible        | civis  | vi | Make cursor invisible                            |
| cursor_left             | cubl   | le | Move left one space.                             |
| cursor_mem_address      | mrcup  | CM | Memory relative cursor addressing                |
| cursor_normal           | cnorm  | ve | Make cursor appear normal (undo vs/vi)           |
| cursor_right            | cuf1   | nd | Non-destructive space (cursor or carriage right) |
| cursor_to_ll            | ll     | ll | Last line, first column (if no cup)              |
| cursor_up               | cuu1   | up | Upline (cursor up)                               |
| cursor_visible          | cvvis  | vs | Make cursor very visible                         |
| define_bit_image_region | defbi  | Yx | Define rectangular bit-image region (use tparm)  |
| define_char             | defc   | ZE | Define a character in a character set            |
| delete_character        | dch1   | dc | Delete character                                 |
| delete_line             | dl1    | dl | Delete line                                      |
| device_type             | devt   | dv | Indicate language/codeset support                |
| dial_phone              | dial   | DI | Dial phone number #1                             |
| dis_status_line         | dsl    | ds | Disable status line                              |
| display_clock           | dclk   | DK | Display time-of-day clock                        |
| display_pc_char         | dispc  | S1 | Display PC character                             |

|                           |       |    |                                           |
|---------------------------|-------|----|-------------------------------------------|
| down_half_line            | hd    | hd | Half-line down (forward 1/2 linefeed)     |
| ena_acs                   | enacs | eA | Enable alternate character set            |
| end_bit_image_region      | endbi | Yy | End a bit-image region (use tparm)        |
| enter_alt_charset_mode    | smacs | as | Start alternate character set             |
| enter_am_mode             | smam  | SA | Turn on automatic margins                 |
| enter_blink_mode          | blink | mb | Turn on blinking                          |
| enter_bold_mode           | bold  | md | Turn on bold (extra bright) mode          |
| enter_ca_mode             | smcup | ti | String to begin programs that use cup     |
| enter_delete_mode         | smdc  | dm | Delete mode (enter)                       |
| enter_dim_mode            | dim   | mh | Turn on half-bright mode                  |
| enter_doublewide_mode     | swidm | ZF | Enable double wide printing               |
| enter_draft_quality       | sdrfq | ZG | Set draft quality print mode              |
| enter_insert_mode         | smir  | im | Insert mode (enter)                       |
| enter_italics_mode        | sitm  | ZH | Enable italics                            |
| enter_leftward_mode       | slm   | ZI | Enable leftward carriage motion           |
| enter_micro_mode          | smicm | ZJ | Enable micro motion capabilities          |
| enter_near_letter_quality | snlq  | ZK | Set near-letter quality print             |
| enter_normal_quality      | snrmq | ZL | Set normal quality                        |
| enter_pc_charset_mode     | smpch | S2 | Enter PC character display mode           |
| enter_protected_mode      | prot  | mp | Turn on protected mode                    |
| enter_reverse_mode        | rev   | mr | Turn on reverse video mode                |
| enter_scancode_mode       | smsc  | S4 | Enter PC scancode mode                    |
| enter_scancode_mode       | smsc  | S4 | Enter PC scancode mode                    |
| enter_secure_mode         | invis | mk | Turn on blank mode (characters invisible) |
| enter_shadow_mode         | sshm  | ZM | Enable shadow printing                    |
| enter_standout_mode       | smso  | so | Begin standout mode                       |
| enter_subscript_mode      | ssubm | ZN | Enable subscript printing                 |
| enter_superscript_mode    | ssupm | ZO | Enable superscript printing               |
| enter_underline_mode      | smul  | us | Start underscore mode                     |
| enter_upward_mode         | sum   | ZP | Enable upward carriage motion mode        |
| enter_xon_mode            | smxon | SX | Turn on xon/xoff handshaking              |
| erase_chars               | ech   | ec | Erase #1 characters                       |
| exit_alt_charset_mode     | rmacs | ae | End alternate character set               |
| exit_am_mode              | rmam  | RA | Turn off automatic margins                |
| exit_attribute_mode       | sgr0  | me | Turn off all attributes                   |
| exit_ca_mode              | rmcup | te | String to end programs that use cup       |
| exit_delete_mode          | rmdc  | ed | End delete mode                           |
| exit_doublewide_mode      | rwidm | ZQ | Disable double wide printing              |
| exit_insert_mode          | rmir  | ei | End insert mode                           |

---

|                       |       |    |                                           |
|-----------------------|-------|----|-------------------------------------------|
| exit_italics_mode     | ritm  | ZR | Disable italics                           |
| exit_leftward_mode    | rlm   | ZS | Enable rightward (normal) carriage motion |
| exit_micro_mode       | rmicm | ZT | Disable micro motion capabilities         |
| exit_pc_charset_mode  | rmpch | S3 | Disable PC character display mode         |
| exit_scancode_mode    | rmsc  | S5 | Disable PC scancode mode                  |
| exit_shadow_mode      | rshm  | ZU | Disable shadow printing                   |
| exit_standout_mode    | rmsso | se | End standout mode                         |
| exit_subscript_mode   | rsubm | ZV | Disable subscript printing                |
| exit_superscript_mode | rsupm | ZW | Disable superscript printing              |
| exit_underline_mode   | rmul  | ue | End underscore mode                       |
| exit_upward_mode      | rum   | ZX | Enable downward (normal) carriage motion  |
| exit_xon_mode         | rmxon | RX | Turn off xon/xoff handshaking             |
| fixed_pause           | pause | PA | Pause for 2-3 seconds                     |
| flash_hook            | hook  | fh | Flash the switch hook                     |
| flash_screen          | flash | vb | Visible bell (may not move cursor)        |
| form_feed             | ff    | ff | Hardcopy terminal page eject              |
| from_status_line      | fsl   | fs | Return from status line                   |
| get_mouse             | getm  | Gm | Curses should get button events           |
| goto_window           | wingo | WG | Go to window #1                           |
| hangup                | hup   | HU | Hang-up phone                             |
| init_1string          | is1   | i1 | Terminal or printer initialization string |
| init_2string          | is2   | is | Terminal or printer initialization string |
| init_3string          | is3   | i3 | Terminal or printer initialization string |
| init_file             | if    | if | Name of initialization file               |
| init_prog             | ipro  | iP | Path name of program for initialization   |
| initialize_color      | initc | Ic | Initialize the definition of color        |
| initialize_pair       | initp | Ip | Initialize color-pair                     |
| insert_character      | ich1  | ic | Insert character                          |
| insert_line           | ill1  | al | Add new blank line                        |
| insert_padding        | ip    | ip | Insert pad after character inserted       |

### key\_Strings

The "key\_" strings are sent by specific keys. The "key\_" descriptions include the macro, defined in `<curses.h>`, for the code returned by the curses routine `getch` when the key is pressed (see [curs\\_getch\(3CURSES\)](#)).

---

Cap-      Termcap



| Variable      | name  | Code | Description                                     |
|---------------|-------|------|-------------------------------------------------|
| key_a1        | ka1   | K1   | KEY_A1, upper left of keypad                    |
| key_a3        | ka3   | K3   | KEY_A3, upper right of keypad                   |
| key_b2        | kb2   | K2   | KEY_B2, center of keypad                        |
| key_backspace | kbs   | kb   | KEY_BACKSPACE, sent by<br>backspace key         |
| key_beg       | kbeg  | @1   | KEY_BEG, sent by beg(inning) key                |
| key_btab      | kcbt  | kB   | KEY_BTAB, sent by back-tab key                  |
| key_c1        | kc1   | K4   | KEY_C1, lower left of keypad                    |
| key_c3        | kc3   | K5   | KEY_C3, lower right of keypad                   |
| key_cancel    | kcan  | @2   | KEY_CANCEL, sent by cancel key                  |
| key_catab     | ktbc  | ka   | KEY_CATAB, sent by<br>clear-all-tabs key        |
| key_clear     | kclr  | kC   | KEY_CLEAR, sent by<br>clear-screen or erase key |
| key_close     | kclo  | @3   | KEY_CLOSE, sent by close key                    |
| key_command   | kcmd  | @4   | KEY_COMMAND, sent by<br>cmd (command) key       |
| key_copy      | kcpy  | @5   | KEY_COPY, sent by copy key                      |
| key_create    | kcrt  | @6   | KEY_CREATE, sent by create key                  |
| key_ctab      | kctab | kt   | KEY_CTAB, sent by clear-tab key                 |
| key_dc        | kdch1 | kD   | KEY_DC, sent by delete-character<br>key         |
| key_dl        | kdll  | kL   | KEY_DL, sent by delete-line key                 |
| key_down      | kcud1 | kd   | KEY_DOWN, sent by terminal<br>down-arrow key    |
| key_eic       | krmir | kM   | KEY_EIC, sent by rmir or smir in<br>insert mode |
| key_end       | kend  | @7   | KEY_END, sent by end key                        |
| key_enter     | kent  | @8   | KEY_ENTER, sent by enter/send<br>key            |
| key_eol       | kel   | kE   | KEY_EOL, sent by<br>clear-to-end-of-line key    |
| key_eos       | ked   | kS   | KEY_EOS, sent by<br>clear-to-end-of-screen key  |
| key_exit      | kext  | @9   | KEY_EXIT, sent by exit key                      |
| key_f0        | kf0   | k0   | KEY_F(0), sent by function key f0               |
| key_f1        | kf1   | k1   | KEY_F(1), sent by function key f1               |
| key_f2        | kf2   | k2   | KEY_F(2), sent by function key f2               |
| key_f3        | kf3   | k3   | KEY_F(3), sent by function key f3               |
| key_fB        | kf4   | k4   | KEY_F(4), sent by function key fB               |
| key_f5        | kf5   | k5   | KEY_F(5), sent by function key f5               |
| key_f6        | kf6   | k6   | KEY_F(6), sent by function key f6               |
| key_f7        | kf7   | k7   | KEY_F(7), sent by function key f7               |
| key_f8        | kf8   | k8   | KEY_F(8), sent by function key f8               |
| key_f9        | kf9   | k9   | KEY_F(9), sent by function key f9               |

|         |      |    |                                     |
|---------|------|----|-------------------------------------|
| key_f10 | kf10 | k; | KEY_F(10), sent by function key f10 |
| key_f11 | kf11 | F1 | KEY_F(11), sent by function key f11 |
| key_f12 | kf12 | F2 | KEY_F(12), sent by function key f12 |
| key_f13 | kf13 | F3 | KEY_F(13), sent by function key f13 |
| key_f14 | kf14 | F4 | KEY_F(14), sent by function key f14 |
| key_f15 | kf15 | F5 | KEY_F(15), sent by function key f15 |
| key_f16 | kf16 | F6 | KEY_F(16), sent by function key f16 |
| key_f17 | kf17 | F7 | KEY_F(17), sent by function key f17 |
| key_f18 | kf18 | F8 | KEY_F(18), sent by function key f18 |
| key_f19 | kf19 | F9 | KEY_F(19), sent by function key f19 |
| key_f20 | kf20 | FA | KEY_F(20), sent by function key f20 |
| key_f21 | kf21 | FB | KEY_F(21), sent by function key f21 |
| key_f22 | kf22 | FC | KEY_F(22), sent by function key f22 |
| key_f23 | kf23 | FD | KEY_F(23), sent by function key f23 |
| key_f24 | kf24 | FE | KEY_F(24), sent by function key f24 |
| key_f25 | kf25 | FF | KEY_F(25), sent by function key f25 |
| key_f26 | kf26 | FG | KEY_F(26), sent by function key f26 |
| key_f27 | kf27 | FH | KEY_F(27), sent by function key f27 |
| key_f28 | kf28 | FI | KEY_F(28), sent by function key f28 |
| key_f29 | kf29 | FJ | KEY_F(29), sent by function key f29 |
| key_f30 | kf30 | FK | KEY_F(30), sent by function key f30 |
| key_f31 | kf31 | FL | KEY_F(31), sent by function key f31 |
| key_f32 | kf32 | FM | KEY_F(32), sent by function key f32 |
| key_f33 | kf33 | FN | KEY_F(13), sent by function key     |

|         |      |    |                                     |
|---------|------|----|-------------------------------------|
|         |      |    | f13                                 |
| key_f34 | kf34 | F0 | KEY_F(34), sent by function key f34 |
| key_f35 | kf35 | FP | KEY_F(35), sent by function key f35 |
| key_f36 | kf36 | FQ | KEY_F(36), sent by function key f36 |
| key_f37 | kf37 | FR | KEY_F(37), sent by function key f37 |
| key_f38 | kf38 | FS | KEY_F(38), sent by function key f38 |
| key_f39 | kf39 | FT | KEY_F(39), sent by function key f39 |
| key_fB0 | kf40 | FU | KEY_F(40), sent by function key fB0 |
| key_fB1 | kf41 | FV | KEY_F(41), sent by function key fB1 |
| key_fB2 | kf42 | FW | KEY_F(42), sent by function key fB2 |
| key_fB3 | kf43 | FX | KEY_F(43), sent by function key fB3 |
| key_fB4 | kf44 | FY | KEY_F(44), sent by function key fB4 |
| key_fB5 | kf45 | FZ | KEY_F(45), sent by function key fB5 |
| key_fB6 | kf46 | Fa | KEY_F(46), sent by function key fB6 |
| key_fB7 | kf47 | Fb | KEY_F(47), sent by function key fB7 |
| key_fB8 | kf48 | Fc | KEY_F(48), sent by function key fB8 |
| key_fB9 | kf49 | Fd | KEY_F(49), sent by function key fB9 |
| key_f50 | kf50 | Fe | KEY_F(50), sent by function key f50 |
| key_f51 | kf51 | Ff | KEY_F(51), sent by function key f51 |
| key_f52 | kf52 | Fg | KEY_F(52), sent by function key f52 |
| key_f53 | kf53 | Fh | KEY_F(53), sent by function key f53 |
| key_f54 | kf54 | Fi | KEY_F(54), sent by function key f54 |
| key_f55 | kf55 | Fj | KEY_F(55), sent by function key f55 |
| key_f56 | kf56 | Fk | KEY_F(56), sent by function key f56 |
| key_f57 | kf57 | FL | KEY_F(57), sent by function key     |

|               |       |    |                                             |
|---------------|-------|----|---------------------------------------------|
|               |       |    | f57                                         |
| key_f58       | kf58  | Fm | KEY_F(58), sent by function key f58         |
| key_f59       | kf59  | Fn | KEY_F(59), sent by function key f59         |
| key_f60       | kf60  | Fo | KEY_F(60), sent by function key f60         |
| key_f61       | kf61  | Fp | KEY_F(61), sent by function key f61         |
| key_f62       | kf62  | Fq | KEY_F(62), sent by function key f62         |
| key_f63       | kf63  | Fr | KEY_F(63), sent by function key f63         |
| key_find      | kfnd  | @0 | KEY_FIND, sent by find key                  |
| key_help      | khlp  | %1 | KEY_HELP, sent by help key                  |
| key_home      | khome | kh | KEY_HOME, sent by home key                  |
| key_ic        | kich1 | kI | KEY_IC, sent by ins-char/enter ins-mode key |
| key_il        | kill  | kA | KEY_IL, sent by insert-line key             |
| key_left      | kcub1 | kl | KEY_LEFT, sent by terminal left-arrow key   |
| key_ll        | kll   | kH | KEY_LL, sent by home-down key               |
| key_mark      | kmrk  | %2 | KEY_MARK, sent by                           |
| key_message   | kmsg  | %3 | KEY_MESSAGE, sent by message key            |
| key_mouse     | kmous | Km | 0631, Mouse event has occurred              |
| key_move      | kmov  | %4 | KEY_MOVE, sent by move key                  |
| key_next      | knxt  | %5 | KEY_NEXT, sent by next-object key           |
| key_npage     | knp   | kN | KEY_NPAGE, sent by next-page key            |
| key_open      | kopn  | %6 | KEY_OPEN, sent by open key                  |
| key_options   | kopt  | %7 | KEY_OPTIONS, sent by options key            |
| key_ppage     | kpp   | kP | KEY_PPAGE, sent by previous-page key        |
| key_previous  | kprv  | %8 | KEY_PREVIOUS, sent by previous-object key   |
| key_print     | kprt  | %9 | KEY_PRINT, sent by print or copy key        |
| key_redo      | krdo  | %0 | KEY_REDO, sent by redo key                  |
| key_reference | kref  | &1 | KEY_REFERENCE, sent by reference key        |
| key_refresh   | krfr  | &2 | KEY_REFRESH, sent by refresh key            |
| key_replace   | krpl  | &3 | KEY_REPLACE, sent by replace key            |
| key_restart   | krst  | &4 | KEY_RESTART, sent by restart key            |

---

|               |       |    |                                                |
|---------------|-------|----|------------------------------------------------|
| key_resume    | kres  | &5 | KEY_RESUME, sent by resume key                 |
| key_right     | kcufl | kr | KEY_RIGHT, sent by terminal<br>right-arrow key |
| key_save      | ksav  | &6 | KEY_SAVE, sent by save key                     |
| key_sbeg      | kBEG  | &9 | KEY_SBEG, sent by<br>shifted beginning key     |
| key_scancel   | kCAN  | &0 | KEY_SCANCEL, sent by<br>shifted cancel key     |
| key_scommand  | kCMD  | *1 | KEY_SCOMMAND, sent by<br>shifted command key   |
| key_scopy     | kCPY  | *2 | KEY_SCOPY, sent by<br>shifted copy key         |
| key_screate   | kCRT  | *3 | KEY_SCREATE, sent by<br>shifted create key     |
| key_sdc       | kDC   | *4 | KEY_SDC, sent by<br>shifted delete-char key    |
| key_sdl       | kDL   | *5 | KEY_SDL, sent by<br>shifted delete-line key    |
| key_select    | kslt  | *6 | KEY_SELECT, sent by<br>select key              |
| key_send      | kEND  | *7 | KEY_SEND, sent by<br>shifted end key           |
| key_seol      | KEOL  | *8 | KEY_SEOL, sent by<br>shifted clear-line key    |
| key_sexit     | kEXT  | *9 | KEY_SEXIT, sent by<br>shifted exit key         |
| key_sf        | kind  | kF | KEY_SF, sent by<br>scroll-forward/down key     |
| key_sfind     | kFND  | *0 | KEY_SFIND, sent by<br>shifted find key         |
| key_shelp     | kHLP  | #1 | KEY_SHELP, sent by<br>shifted help key         |
| key_shome     | KHOM  | #2 | KEY_SHOME, sent by<br>shifted home key         |
| key_sic       | kIC   | #3 | KEY_SIC, sent by<br>shifted input key          |
| key_sleft     | kLFT  | #4 | KEY_SLEFT, sent by<br>shifted left-arrow key   |
| key_smessage  | kMSG  | %a | KEY_SMESSAGE, sent by<br>shifted message key   |
| key_smove     | kMOV  | %b | KEY_SMOVE, sent by<br>shifted move key         |
| key_snext     | KNXT  | %c | KEY_SNEXT, sent by<br>shifted next key         |
| key_soptions  | KOPT  | %d | KEY_SOPTIONS, sent by<br>shifted options key   |
| key_sprevious | kPRV  | %e | KEY_SPREVIOUS, sent by<br>shifted prev key     |

---

|              |       |    |                                             |
|--------------|-------|----|---------------------------------------------|
| key_sprint   | kPRT  | %f | KEY_SPRINT, sent by shifted print key       |
| key_sr       | kri   | kR | KEY_SR, sent by scroll-backward/up key      |
| key_sredo    | krDO  | %g | KEY_SREDO, sent by shifted redo key         |
| key_sreplace | krPL  | %h | KEY_SREPLACE, sent by shifted replace key   |
| key_sright   | krIT  | %i | KEY_SRIGHT, sent by shifted right-arrow key |
| key_sresume  | kRES  | %j | KEY_SRSUME, sent by shifted resume key      |
| key_ssave    | kSAV  | !1 | KEY_SSAVE, sent by shifted save key         |
| key_ssuspend | kSPD  | !2 | KEY_SSUSPEND, sent by shifted suspend key   |
| key_stab     | khts  | kT | KEY_STAB, sent by set-tab key               |
| key_sundo    | kUND  | !3 | KEY_SUNDO, sent by shifted undo key         |
| key_suspend  | kspd  | &7 | KEY_SUSPEND, sent by suspend key            |
| key_undo     | kund  | &8 | KEY_UNDO, sent by undo key                  |
| key_up       | kcuu1 | ku | KEY_UP, sent by terminal up-arrow key       |
| keypad_local | rmkx  | ke | Out of "keypad-transmit" mode               |
| keypad_xmit  | smkx  | ks | Put terminal in "keypad-transmit" mode      |
| lab_f0       | lf0   | l0 | Labels on function key f0 if not f0         |
| lab_f1       | lf1   | l1 | Labels on function key f1 if not f1         |
| lab_f2       | lf2   | l2 | Labels on function key f2 if not f2         |
| lab_f3       | lf3   | l3 | Labels on function key f3 if not f3         |
| lab_fB       | lfB   | l4 | Labels on function key fB if not fB         |
| lab_f5       | lf5   | l5 | Labels on function key f5 if not f5         |
| lab_f6       | lf6   | l6 | Labels on function key f6 if not f6         |
| lab_f7       | lf7   | l7 | Labels on function key f7 if not f7         |
| lab_f8       | lf8   | l8 | Labels on function key f8 if not f8         |
| lab_f9       | lf9   | l9 | Labels on function key                      |

|                      |        |    |                                               |
|----------------------|--------|----|-----------------------------------------------|
|                      |        |    | f9 if not f9                                  |
| lab_f10              | lf10   | la | Labels on function key<br>f10 if not f10      |
| label_format         | fln    | Lf | Label format                                  |
| label_off            | rmln   | LF | Turn off soft labels                          |
| label_on             | smln   | LO | Turn on soft labels                           |
| meta_off             | rmm    | mo | Turn off "meta mode"                          |
| meta_on              | smm    | mm | Turn on "meta mode" (8th bit)                 |
| micro_column_address | mhpa   | ZY | Like column_address<br>for micro adjustment   |
| micro_down           | mcud1  | ZZ | Like cursor_down<br>for micro adjustment      |
| micro_left           | mcub1  | Za | Like cursor_left<br>for micro adjustment      |
| micro_right          | mcuf1  | Zb | Like cursor_right<br>for micro adjustment     |
| micro_row_address    | mvpa   | Zc | Like row_address<br>for micro adjustment      |
| micro_up             | mcuu1  | Zd | Like cursor_up<br>for micro adjustment        |
| mouse_info           | minfo  | Mi | Mouse status information                      |
| newline              | nel    | nw | Newline (behaves like<br>cr followed by lf)   |
| order_of_pins        | porder | Ze | Matches software bits<br>to print-head pins   |
| orig_colors          | oc     | oc | Set all color(-pair)s<br>to the original ones |
| orig_pair            | op     | op | Set default color-pair<br>to the original one |
| pad_char             | pad    | pc | Pad character (rather than null)              |
| parm_dch             | dch    | DC | Delete #1 chars                               |
| parm_delete_line     | dl     | DL | Delete #1 lines                               |
| parm_down_cursor     | cud    | DO | Move down #1 lines                            |
| parm_down_micro      | mcud   | Zf | Like parm_down_cursor<br>for micro adjust     |
| parm_ich             | ich    | IC | Insert #1 blank chars                         |
| parm_index           | indn   | SF | Scroll forward #1 lines                       |
| parm_insert_line     | il     | AL | Add #1 new blank lines                        |
| parm_left_cursor     | cub    | LE | Move cursor left #1 spaces                    |
| parm_left_micro      | mcub   | Zg | Like parm_left_cursor<br>for micro adjust     |
| parm_right_cursor    | cuf    | RI | Move right #1 spaces                          |
| parm_right_micro     | mcuf   | Zh | Like parm_right_cursor<br>for micro adjust    |
| parm_rindex          | rin    | SR | Scroll backward #1 lines                      |
| parm_up_cursor       | cuu    | UP | Move cursor up #1 lines                       |
| parm_up_micro        | mcuu   | Zi | Like parm_up_cursor<br>for micro adjust       |

---

|                  |       |    |                                                     |
|------------------|-------|----|-----------------------------------------------------|
| pc_term_options  | pctrm | S6 | PC terminal options                                 |
| pkey_key         | pfkey | pk | Prog funct key #1 to<br>type string #2              |
| pkey_local       | pfloc | pl | Prog funct key #1 to<br>execute string #2           |
| pkey_plab        | pfxl  | xl | Prog key #1 to xmit<br>string #2 and show string #3 |
| pkey_xmit        | px    | px | Prog funct key #1 to<br>xmit string #2              |
| plab_norm        | pln   | pn | Prog label #1 to show<br>string #2                  |
| print_screen     | mc0   | ps | Print contents of the screen                        |
| prtr_non         | mc5p  | p0 | Turn on the printer for #1 bytes                    |
| prtr_off         | mc4   | pf | Turn off the printer                                |
| prtr_on          | mc5   | po | Turn on the printer                                 |
| pulse            | pulse | PU | Select pulse dialing                                |
| quick_dial       | qdial | QD | Dial phone number #1, without<br>progress detection |
| remove_clock     | rmclk | RC | Remove time-of-day clock                            |
| repeat_char      | rep   | rp | Repeat char #1 #2 times                             |
| req_for_input    | rfi   | RF | Send next input char (for ptys)                     |
| req_mouse_pos    | reqmp | RQ | Request mouse position report                       |
| reset_1string    | rs1   | r1 | Reset terminal completely to<br>sane modes          |
| reset_2string    | rs2   | r2 | Reset terminal completely to<br>sane modes          |
| reset_3string    | rs3   | r3 | Reset terminal completely to<br>sane modes          |
| reset_file       | rf    | rf | Name of file containing<br>reset string             |
| restore_cursor   | rc    | rc | Restore cursor to<br>position of last sc            |
| row_address      | vpa   | cv | Vertical position absolute                          |
| save_cursor      | sc    | sc | Save cursor position                                |
| scancode_escape  | scesc | S7 | Escape for scancode emulation                       |
| scroll_forward   | ind   | sf | Scroll text up                                      |
| scroll_reverse   | ri    | sr | Scroll text down                                    |
| select_char_set  | scs   | Zj | Select character set                                |
| set0_des_seq     | s0ds  | s0 | Shift into codeset 0<br>(EUC set 0, ASCII)          |
| set1_des_seq     | s1ds  | s1 | Shift into codeset 1                                |
| set2_des_seq     | s2ds  | s2 | Shift into codeset 2                                |
| set3_des_seq     | s3ds  | s3 | Shift into codeset 3                                |
| set_a_background | setab | AB | Set background color<br>using ANSI escape           |
| set_a_foreground | setaf | AF | Set foreground color<br>using ANSI escape           |



|                        |          |    |                                                      |
|------------------------|----------|----|------------------------------------------------------|
| set_attributes         | sgr      | sa | Define the video attributes #1-#9                    |
| set_background         | setb     | Sb | Set current background color                         |
| set_bottom_margin      | smgb     | Zk | Set bottom margin at current line                    |
| set_bottom_margin_parm | smgbp    | Zl | Set bottom margin at line #1 or #2 lines from bottom |
| set_clock              | sclk     | SC | Set time-of-day clock                                |
| set_color_band         | setcolor |    | YzChange to ribbon color #1                          |
| set_color_pair         | scp      | sp | Set current color-pair                               |
| set_foreground         | setf     | Sf | Set current foreground color1                        |
| set_left_margin        | smgl     | ML | Set left margin at current line                      |
| set_left_margin_parm   | smglp    | Zm | Set left (right) margin at column #1 (#2)            |
| set_lr_margin          | smglr    | ML | Sets both left and right margins                     |
| set_page_length        | slines   | YZ | Set page length to #1 lines (use tparm) of an inch   |
| set_right_margin       | smgr     | MR | Set right margin at current column                   |
| set_right_margin_parm  | smgrp    | Zn | Set right margin at column #1                        |
| set_tab                | hts      | st | Set a tab in all rows, current column                |
| set_tb_margin          | smgtb    | MT | Sets both top and bottom margins                     |
| set_top_margin         | smgt     | Zo | Set top margin at current line                       |
| set_top_margin_parm    | smgtp    | Zp | Set top (bottom) margin at line #1 (#2)              |
| set_window             | wind     | wi | Current window is lines #1-#2 cols #3-#4             |
| start_bit_image        | sbim     | Zq | Start printing bit image graphics                    |
| start_char_set_def     | scsd     | Zr | Start definition of a character set                  |
| stop_bit_image         | rbim     | Zs | End printing bit image graphics                      |
| stop_char_set_def      | rcsd     | Zt | End definition of a character set                    |
| subscript_characters   | subcs    | Zu | List of "subscript-able" characters                  |
| superscript_characters | supcs    | Zv | List of "superscript-able" characters                |
| tab                    | ht       | ta | Tab to next 8-space hardware tab stop                |
| these_cause_cr         | docr     | Zw | Printing any of these chars causes cr                |
| to_status_line         | tsl      | ts | Go to status line, col #1                            |
| tone                   | tone     | TO | Select touch tone dialing                            |
| user0                  | u0       | u0 | User string 0                                        |
| user1                  | u1       | u1 | User string 1                                        |
| user2                  | u2       | u2 | User string 2                                        |
| user3                  | u3       | u3 | User string 3                                        |

|                |       |    |                                           |
|----------------|-------|----|-------------------------------------------|
| user4          | u4    | u4 | User string 4                             |
| user5          | u5    | u5 | User string 5                             |
| user6          | u6    | u6 | User string 6                             |
| user7          | u7    | u7 | User string 7                             |
| user8          | u8    | u8 | User string 8                             |
| user9          | u9    | u9 | User string 9                             |
| underline_char | uc    | uc | Underscore one char<br>and move past it   |
| up_half_line   | hu    | hu | Half-line up (reverse<br>1/2 linefeed)    |
| wait_tone      | wait  | WA | Wait for dial tone                        |
| xoff_character | xoffc | XF | X-off character                           |
| xon_character  | xonc  | XN | X-on character                            |
| zero_motion    | zerom | Zx | No motion for the<br>subsequent character |

Sample Entry The following entry, which describes the AT&T 610 terminal, is among the more complex entries in the terminfo file as of this writing.

```
610|610bct|ATT610|att610|AT&T610;80column;98key keyboard
am, eslok, hs, mir, msgr, xenl, xon,
cols#80, it#8, lh#2, lines#24, lw#8, nlab#8, wsl#80,
acsc="aaffggjjkkllmmnnooppqqrrssttuvwxyz{|}|}~",
bel=^G, blink=\E[5m, bold=\E[1m, cbt=\E[Z,
civis=\E[?25l, clear=\E[H\E[J, cnorm=\E[?25h\E[?12l,
cr=\r, csr=\E[%i%p1%d;%p2%dr, cub=\E[%p1%dD, cub1=\b,
cud=\E[%p1%dB, cud1=\E[B, cuf=\E[%p1%DC, cuf1=\E[C,
cup=\E[%i%p1%d;%p2%dH, cuu=\E[%p1%DA, cuu1=\E[A,
cvvis=\E[?12;25h, dch=\E[%p1%DP, dch1=\E[P, dim=\E[2m,
dl=\E[%p1%DM, dll=\E[M, ed=\E[J, el=\E[K, ell=\E[1K,
flash=\E[?5h$<200>\E[?5l, fsl=\E8, home=\E[H, ht=\t,
ich=\E[%p1%d@, il=\E[%p1%DL, ill=\E[L, ind=\ED, .ind=\ED$<9>,
invis=\E[8m,
is1=\E[8;0 | \E[?3;4;5;13;15l\E[13;20l\E[?7h\E[12h\E(B\E)0,
is2=\E[0m^O, is3=\E(B\E)0, kLFT=\E[\s@, kRIT=\E[\sA,
kbs=^H, kcbt=\E[Z, kclr=\E[2J, kcub1=\E[D, kcud1=\E[B,
kcufl=\E[C, kcuu1=\E[A, kf1=\E0c, kf10=\ENp,
kf11=\ENq, kf12=\ENr, kf13=\ENS, kf14=\ENT, kf2=\E0d,
kf3=\E0e, kf4=\E0f, kf5=\E0g, kf6=\E0h, kf7=\E0i,
kf8=\E0j, kf9=\E0n, khome=\E[H, kind=\E[S, kri=\E[T,
ll=\E[24H, mc4=\E[?4i, mc5=\E[?5i, nel=\EE,
pfxl=\E[%p1%d;%p2%l%02dq%?%p1{9}%<t\s\s\F%p1%d\s\s\s\s\s
\s\s\s\s\s\s%?%p2%$s,
pln=\E[%p1%d;0;0;0q%?%p2%:-16.16s, rc=\E8, rev=\E[7m,
ri=\EM, rmacs=^O, rmir=\E[4l, rmln=\E[2p, rmso=\E[m,
rmul=\E[m, rs2=\Ec\E[?3l, sc=\E7,
sgr=\E[0%?%p6%t;1%?%p5%t;2%?%p2%t;4%?%p4%t;5%
%?%p3%?%p1 | %t;7%?%p7%t;8%?%p9%t^N%e^0%,
sgr0=\E[m^O, smacs=^N, smir=\E[4h, smln=\E[p,
```

```
smso=\E[7m, smul=\E[4m, tsl=\E7\E[25;%i%p1%dx,
```

#### Types of Capabilities in the Sample Entry

The sample entry shows the formats for the three types of `terminfo` capabilities listed: Boolean, numeric, and string. All capabilities specified in the `terminfo` source file must be followed by commas, including the last capability in the source file. In `terminfo` source files, capabilities are referenced by their capability names (as shown in the previous tables).

Boolean capabilities are specified simply by their comma separated cap names.

Numeric capabilities are followed by the character '#' and then a positive integer value. Thus, in the sample, `cols` (which shows the number of columns available on a device) is assigned the value 80 for the AT&T 610. (Values for numeric capabilities may be specified in decimal, octal, or hexadecimal, using normal C programming language conventions.)

Finally, string-valued capabilities such as `el` (clear to end of line sequence) are listed by a two-to five-character capname, an '=', and a string ended by the next occurrence of a comma. A delay in milliseconds may appear anywhere in such a capability, preceded by '\$' and enclosed in angle brackets, as in `el=\EK$<3>`. Padding characters are supplied by `tput`. The delay can be any of the following: a number, a number followed by an asterisk, such as `5*`, a number followed by a slash, such as `5/`, or a number followed by both, such as `5*/`. A '\*' shows that the padding required is proportional to the number of lines affected by the operation, and the amount given is the per-affected-unit padding required. (In the case of insert characters, the factor is still the number of lines affected. This is always 1 unless the device has `in` and the software uses it.) When a '\*' is specified, it is sometimes useful to give a delay of the form `3.5` to specify a delay per unit to tenths of milliseconds. (Only one decimal place is allowed.)

A '/' indicates that the padding is mandatory. If a device has `xon` defined, the padding information is advisory and will only be used for cost estimates or when the device is in raw mode. Mandatory padding will be transmitted regardless of the setting of `xon`. If padding (whether advisory or mandatory) is specified for `bel` or `flash`, however, it will always be used, regardless of whether `xon` is specified.

`terminfo` offers notation for encoding special characters. Both `\E` and `\e` map to an ESCAPE character, `^x` maps to a control `x` for any appropriate `x`, and the sequences `\n`, `\l`, `\r`, `\t`, `\b`, `\f`, and `\s` give a newline, linefeed, return, tab, backspace, formfeed, and space, respectively. Other escapes include: `\^` for caret (^); `\` for backslash (\); `\,` for comma (,); `\:` for colon (:); and `\0` for null. (`\0` will actually produce `\200`, which does not terminate a string but behaves as a null character on most devices, providing CS7 is specified. (See [stty\(1\)](#))). Finally, characters may be given as three octal digits after a backslash (for example, `\123`).

Sometimes individual capabilities must be commented out. To do this, put a period before the capability name. For example, see the second `ind` in the example above. Note that capabilities are defined in a left-to-right order and, therefore, a prior definition will override a later definition.

**Preparing Descriptions** The most effective way to prepare a device description is by imitating the description of a similar device in `terminfo` and building up a description gradually, using partial descriptions with `vi` to check that they are correct. Be aware that a very unusual device may expose deficiencies in the ability of the `terminfo` file to describe it or the inability of `vi` to work with that device. To test a new device description, set the environment variable `TERMINFO` to the pathname of a directory containing the compiled description you are working on and programs will look there rather than in `/usr/share/lib/terminfo`. To get the padding for insert-line correct (if the device manufacturer did not document it) a severe test is to comment out `xon`, edit a large file at 9600 baud with `vi`, delete 16 or so lines from the middle of the screen, and then press the `u` key several times quickly. If the display is corrupted, more padding is usually needed. A similar test can be used for insert-character.

**Section 1-1: Basic Capabilities** The number of columns on each line for the device is given by the `cols` numeric capability. If the device has a screen, then the number of lines on the screen is given by the `lines` capability. If the device wraps around to the beginning of the next line when it reaches the right margin, then it should have the `am` capability. If the terminal can clear its screen, leaving the cursor in the home position, then this is given by the `clear` string capability. If the terminal overstrikes (rather than clearing a position when a character is struck over) then it should have the `os` capability. If the device is a printing terminal, with no soft copy unit, specify both `hc` and `os`. If there is a way to move the cursor to the left edge of the current row, specify this as `cr`. (Normally this will be carriage return, control M.) If there is a way to produce an audible signal (such as a bell or a beep), specify it as `bel`. If, like most devices, the device uses the `xon-xoff` flow-control protocol, specify `xon`.

If there is a way to move the cursor one position to the left (such as backspace), that capability should be given as `cub1`. Similarly, sequences to move to the right, up, and down should be given as `cuf1`, `cuu1`, and `cud1`, respectively. These local cursor motions must not alter the text they pass over; for example, you would not normally use "`cuf1=\s`" because the space would erase the character moved over.

A very important point here is that the local cursor motions encoded in `terminfo` are undefined at the left and top edges of a screen terminal. Programs should never attempt to backspace around the left edge, unless `bw` is specified, and should never attempt to go up locally off the top. To scroll text up, a program goes to the bottom left corner of the screen and sends the `ind` (index) string.

To scroll text down, a program goes to the top left corner of the screen and sends the `ri` (reverse index) string. The strings `ind` and `ri` are undefined when not on their respective corners of the screen.

Parameterized versions of the scrolling sequences are `indn` and `rin`. These versions have the same semantics as `ind` and `ri`, except that they take one parameter and scroll the number of lines specified by that parameter. They are also undefined except at the appropriate edge of the screen.

The `am` capability tells whether the cursor sticks at the right edge of the screen when text is output, but this does not necessarily apply to a `cu f1` from the last column. Backward motion from the left edge of the screen is possible only when `bw` is specified. In this case, `cub1` will move to the right edge of the previous row. If `bw` is not given, the effect is undefined. This is useful for drawing a box around the edge of the screen, for example. If the device has switch selectable automatic margins, `am` should be specified in the `terminfo` source file. In this case, initialization strings should turn on this option, if possible. If the device has a command that moves to the first column of the next line, that command can be given as `ne l` (newline). It does not matter if the command clears the remainder of the current line, so if the device has no `cr` and `lf` it may still be possible to craft a working `ne l` out of one or both of them.

These capabilities suffice to describe hardcopy and screen terminals. Thus the AT&T 5320 hardcopy terminal is described as follows:

```
5320|att5320|AT&T 5320 hardcopy terminal,
 am, hc, os,
 cols#132,
 bel=^G, cr=^r, cub1=^b, cnd1=^n,
 dch1=^E[P, dll=^E[M,
 ind=^n,
```

while the Lear Siegler ADM-3 is described as

```
adm3 | lsi adm3,
 am, bel=^G, clear=^Z, cols#80, cr=^M, cub1=^H,
 cud1=^J, ind=^J, lines#24,
```

## Section 1-2: Parameterized Strings

Cursor addressing and other strings requiring parameters are described by a parameterized string capability, with `printf`-like escapes (`%x`) in it. For example, to address the cursor, the `cup` capability is given, using two parameters: the row and column to address to. (Rows and columns are numbered from zero and refer to the physical screen visible to the user, not to any unseen memory.) If the terminal has memory relative cursor addressing, that can be indicated by `mr cup`.

The parameter mechanism uses a stack and special `%` codes to manipulate the stack in the manner of Reverse Polish Notation (postfix). Typically a sequence will push one of the parameters onto the stack and then print it in some format. Often more complex operations are necessary. Operations are in postfix form with the operands in the usual order. That is, to subtract 5 from the first parameter, one would use `%p1%{5}%-`.

The `%` encodings have the following meanings:

|                                                    |                                                                    |
|----------------------------------------------------|--------------------------------------------------------------------|
| <code>%%</code>                                    | outputs <code>'%'</code>                                           |
| <code>%[[:]flags][width[.precision]][doxXs]</code> | as in <code>printf</code> , flags are <code>[-+#]</code> and space |
| <code>%c</code>                                    | print pop gives <code>%c</code>                                    |
| <code>%p[1-9]</code>                               | push <i>i</i> th parm                                              |

|                                                |                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>%P[a-z]</code>                           | set dynamic variable [a-z] to pop                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>%g[a-z]</code>                           | get dynamic variable [a-z] and push it                                                                                                                                                                                                                                                                                                                                                                              |
| <code>%P[A-Z]</code>                           | set static variable [a-z] to pop                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>%g[A-Z]</code>                           | get static variable [a-z] and push it                                                                                                                                                                                                                                                                                                                                                                               |
| <code>%'c'</code>                              | push char constant <i>c</i>                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>%{nn}</code>                             | push decimal constant <i>nn</i>                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>%l</code>                                | push <code>strlen(pop)</code>                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>%+ %− %* %/ %m</code>                    | arithmetic (% <i>m</i> is mod): push(pop integer2 op pop integer1)                                                                                                                                                                                                                                                                                                                                                  |
| <code>%&amp; %  %^</code>                      | bit operations: push(pop integer2 op pop integer1)                                                                                                                                                                                                                                                                                                                                                                  |
| <code>%= %&gt; %&lt;</code>                    | logical operations: push(pop integer2 op pop integer1)                                                                                                                                                                                                                                                                                                                                                              |
| <code>%A %O</code>                             | logical operations: and, or                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>%! %~</code>                             | unary operations: push(op pop)                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>%i</code>                                | (for ANSI terminals) add 1 to first parm, if one parm present, or first two parms, if more than one parm present                                                                                                                                                                                                                                                                                                    |
| <code>%? expr %t thenpart %e elsepart %</code> | if-then-else, % <i>e</i> <i>elsepart</i> is optional; else-if's are possible ala Algol 68: %? <i>c</i> <sub>1</sub> %t <i>b</i> <sub>1</sub> %e <i>c</i> <sub>2</sub> %t <i>b</i> <sub>2</sub> %e <i>c</i> <sub>3</sub> %t <i>b</i> <sub>3</sub> %e <i>c</i> <sub>4</sub> %t <i>b</i> <sub>4</sub> %e <i>b</i> <sub>5</sub> % <i>c</i> <sub><i>i</i></sub> are conditions, <i>b</i> <sub><i>i</i></sub> are bodies. |

If the “−” flag is used with “%[doxXs]”, then a colon (:) must be placed between the “%” and the “−” to differentiate the flag from the binary “%−” operator, for example “%:−16.16s”.

Consider the Hewlett-Packard 2645, which, to get to row 3 and column 12, needs to be sent `\E&a12c03Y` padded for 6 milliseconds. Note that the order of the rows and columns is inverted here, and that the row and column are zero-padded as two digits. Thus its cup capability is: `cup=\E&a%p2%2.2dc%p1%2.2dY$<6>`

The Micro-Term ACT-IV needs the current row and column sent preceded by a `^T`, with the row and column simply encoded in binary, “`cup=^T%p1%c%p2%c`”. Devices that use “%c” need to be able to backspace the cursor (`cub1`), and to move the cursor up one line on the screen (`cuu1`). This is necessary because it is not always safe to transmit `\n`, `^D`, and `\r`, as the system may change or discard them. (The library routines dealing with `terminfo` set tty modes so that tabs are never expanded, so `\t` is safe to send. This turns out to be essential for the Ann Arbor 4080.)

A final example is the LSI ADM-3a, which uses row and column offset by a blank character, thus "`cup=\E=%p1%\s'+%c%p2%\s'+%c`". After sending "`\E=`", this pushes the first parameter, pushes the ASCII value for a space (32), adds them (pushing the sum on the stack in place of the two previous values), and outputs that value as a character. Then the same is done for the second parameter. More complex arithmetic is possible using the stack.

### Section 1-3: Cursor Motions

If the terminal has a fast way to home the cursor (to very upper left corner of screen) then this can be given as `home`; similarly a fast way of getting to the lower left-hand corner can be given as `ll`; this may involve going up with `cuu1` from the home position, but a program should never do this itself (unless `ll` does) because it can make no assumption about the effect of moving up from the home position. Note that the home position is the same as addressing to (0,0): to the top left corner of the screen, not of memory. (Thus, the `\EH` sequence on Hewlett-Packard terminals cannot be used for home without losing some of the other features on the terminal.)

If the device has row or column absolute-cursor addressing, these can be given as single parameter capabilities `hpa` (horizontal position absolute) and `vpa` (vertical position absolute). Sometimes these are shorter than the more general two-parameter sequence (as with the Hewlett-Packard 2645) and can be used in preference to `cup`. If there are parameterized local motions (for example, move *n* spaces to the right) these can be given as `cud`, `cub`, `cuf`, and `cuu` with a single parameter indicating how many spaces to move. These are primarily useful if the device does not have `cup`, such as the Tektronix 4025.

If the device needs to be in a special mode when running a program that uses these capabilities, the codes to enter and exit this mode can be given as `smcup` and `rmcup`. This arises, for example, from terminals, such as the Concept, with more than one page of memory. If the device has only memory relative cursor addressing and not screen relative cursor addressing, a one screen-sized window must be fixed into the device for cursor addressing to work properly. This is also used for the Tektronix 4025, where `smcup` sets the command character to be the one used by `terminfo`. If the `smcup` sequence will not restore the screen after an `rmcup` sequence is output (to the state prior to outputting `rmcup`), specify `nr rmc`.

### Section 1-4: Area Clears

If the terminal can clear from the current position to the end of the line, leaving the cursor where it is, this should be given as `e1`. If the terminal can clear from the beginning of the line to the current position inclusive, leaving the cursor where it is, this should be given as `e11`. If the terminal can clear from the current position to the end of the display, then this should be given as `ed`. `ed` is only defined from the first column of a line. (Thus, it can be simulated by a request to delete a large number of lines, if a true `ed` is not available.)

### Section 1-5: Insert/Delete Line

If the terminal can open a new blank line before the line where the cursor is, this should be given as `i11`; this is done only from the first position of a line. The cursor must then appear on the newly blank line. If the terminal can delete the line which the cursor is on, then this should be given as `d11`; this is done only from the first position on the line to be deleted. Versions of `i11` and `d11` which take a single parameter and insert or delete that many lines can be given as `i1` and `d1`.

If the terminal has a settable destructive scrolling region (like the VT100) the command to set this can be described with the `csr` capability, which takes two parameters: the top and bottom lines of the scrolling region. The cursor position is, alas, undefined after using this command. It is possible to get the effect of insert or delete line using this command — the `sc` and `rc` (save and restore cursor) commands are also useful. Inserting lines at the top or bottom of the screen can also be done using `ri` or `ind` on many terminals without a true insert/delete line, and is often faster even on terminals with those features.

To determine whether a terminal has destructive scrolling regions or non-destructive scrolling regions, create a scrolling region in the middle of the screen, place data on the bottom line of the scrolling region, move the cursor to the top line of the scrolling region, and do a reverse index (`ri`) followed by a delete line (`d11`) or index (`ind`). If the data that was originally on the bottom line of the scrolling region was restored into the scrolling region by the `d11` or `ind`, then the terminal has non-destructive scrolling regions. Otherwise, it has destructive scrolling regions. Do not specify `csr` if the terminal has non-destructive scrolling regions, unless `ind`, `ri`, `indn`, `rin`, `d1`, and `d11` all simulate destructive scrolling.

If the terminal has the ability to define a window as part of memory, which all commands affect, it should be given as the parameterized string `wind`. The four parameters are the starting and ending lines in memory and the starting and ending columns in memory, in that order.

If the terminal can retain display memory above, then the `da` capability should be given; if display memory can be retained below, then `db` should be given. These indicate that deleting a line or scrolling a full screen may bring non-blank lines up from below or that scrolling back with `ri` may bring down non-blank lines.

Section 1-6:  
Insert/Delete Character

There are two basic kinds of intelligent terminals with respect to insert/delete character operations which can be described using `terminfo`. The most common insert/delete character operations affect only the characters on the current line and shift characters off the end of the line rigidly. Other terminals, such as the Concept 100 and the Perkin Elmer Owl, make a distinction between typed and untyped blanks on the screen, shifting upon an insert or delete only to an untyped blank on the screen which is either eliminated, or expanded to two untyped blanks. You can determine the kind of terminal you have by clearing the screen and then typing text separated by cursor motions. Type "abc def" using local cursor motions (not spaces) between the abc and the def. Then position the cursor before the abc and put the terminal in insert mode. If typing characters causes the rest of the line to shift rigidly and characters to fall off the end, then your terminal does not distinguish between blanks and untyped positions. If the abc shifts over to the def which then move together around the end of the current line and onto the next as you insert, you have the second type of terminal, and should give the capability `in`, which stands for "insert null." While these are two logically separate attributes (one line versus multiline insert mode, and special treatment of untyped spaces) we have seen no terminals whose insert mode cannot be described with the single attribute.



`terminfo` can describe both terminals that have an insert mode and terminals which send a simple sequence to open a blank position on the current line. Give as `smir` the sequence to get into insert mode. Give as `rmir` the sequence to leave insert mode. Now give as `ich1` any sequence needed to be sent just before sending the character to be inserted. Most terminals with a true insert mode will not give `ich1`; terminals that send a sequence to open a screen position should give it here. (If your terminal has both, insert mode is usually preferable to `ich1`. Do not give both unless the terminal actually requires both to be used in combination.) If post-insert padding is needed, give this as a number of milliseconds padding in `ip` (a string option). Any other sequence which may need to be sent after an insert of a single character may also be given in `ip`. If your terminal needs both to be placed into an 'insert mode' and a special code to precede each inserted character, then both `smir/rmir` and `ich1` can be given, and both will be used. The `ich` capability, with one parameter, `n`, will insert `n` blanks.

If padding is necessary between characters typed while not in insert mode, give this as a number of milliseconds padding in `rpm`.

It is occasionally necessary to move around while in insert mode to delete characters on the same line (for example, if there is a tab after the insertion position). If your terminal allows motion while in insert mode you can give the capability `mir` to speed up inserting in this case. Omitting `mir` will affect only speed. Some terminals (notably Datamedia's) must not have `mir` because of the way their insert mode works.

Finally, you can specify `dch1` to delete a single character, `dch` with one parameter, `n`, to delete `n` characters, and delete mode by giving `smdc` and `rmdc` to enter and exit delete mode (any mode the terminal needs to be placed in for `dch1` to work).

A command to erase `n` characters (equivalent to outputting `n` blanks without moving the cursor) can be given as `ech` with one parameter.

Section 1-7:  
Highlighting,  
Underlining, and  
Visible Bells

Your device may have one or more kinds of display attributes that allow you to highlight selected characters when they appear on the screen. The following display modes (shown with the names by which they are set) may be available: a blinking screen (`blink`), bold or extra-bright characters (`bold`), dim or half-bright characters (`dim`), blanking or invisible text (`invis`), protected text (`prot`), a reverse-video screen (`rev`), and an alternate character set (`smacs` to enter this mode and `rmacs` to exit it). (If a command is necessary before you can enter alternate character set mode, give the sequence in `enacs` or "enable alternate-character-set" mode.) Turning on any of these modes singly may or may not turn off other modes.

`sgro` should be used to turn off all video enhancement capabilities. It should always be specified because it represents the only way to turn off some capabilities, such as `dim` or `blink`.

You should choose one display method as *standout mode* and use it to highlight error messages and other kinds of text to which you want to draw attention. Choose a form of display that provides strong contrast but that is easy on the eyes. (We recommend

reverse-video plus half-bright or reverse-video alone.) The sequences to enter and exit standout mode are given as `sms0` and `rms0`, respectively. If the code to change into or out of standout mode leaves one or even two blank spaces on the screen, as the TVI 912 and Teleray 1061 do, then `xmc` should be given to tell how many spaces are left.

Sequences to begin underlining and end underlining can be specified as `smul` and `rmul`, respectively. If the device has a sequence to underline the current character and to move the cursor one space to the right (such as the Micro-Term MIME), this sequence can be specified as `uc`.

Terminals with the "magic cookie" glitch (`xmc`) deposit special "cookies" when they receive mode-setting sequences, which affect the display algorithm rather than having extra bits for each character. Some terminals, such as the Hewlett-Packard 2621, automatically leave standout mode when they move to a new line or the cursor is addressed. Programs using standout mode should exit standout mode before moving the cursor or sending a newline, unless the `msgR` capability, asserting that it is safe to move in standout mode, is present.

If the terminal has a way of flashing the screen to indicate an error quietly (a bell replacement), then this can be given as `flash`; it must not move the cursor. A good flash can be done by changing the screen into reverse video, `pad` for 200 ms, then return the screen to normal video.

If the cursor needs to be made more visible than normal when it is not on the bottom line (to make, for example, a non-blinking underline into an easier to find block or blinking underline) give this sequence as `cvvis`. The boolean `chts` should also be given. If there is a way to make the cursor completely invisible, give that as `civis`. The capability `cnorm` should be given which undoes the effects of either of these modes.

If your terminal generates underlined characters by using the underline character (with no special sequences needed) even though it does not otherwise overstrike characters, then you should specify the capability `ul`. For devices on which a character overstriking another leaves both characters on the screen, specify the capability `os`. If overstrikes are erasable with a blank, then this should be indicated by specifying `eo`.

If there is a sequence to set arbitrary combinations of modes, this should be given as `sgR` (set attributes), taking nine parameters. Each parameter is either `0` or non-zero, as the corresponding attribute is on or off. The nine parameters are, in order: standout, underline, reverse, blink, dim, bold, blank, protect, alternate character set. Not all modes need to be supported by `sgR`; only those for which corresponding separate attribute commands exist should be supported. For example, let's assume that the terminal in question needs the following escape sequences to turn on various modes.

| tparm     |            |                  |
|-----------|------------|------------------|
| parameter | attribute  | escape sequence  |
|           | none       | \E[0m            |
| p1        | standout   | \E[0;4;7m        |
| p2        | underline  | \E[0;3m          |
| p3        | reverse    | \E[0;4m          |
| p4        | blink      | \E[0;5m          |
| p5        | dim        | \E[0;7m          |
| p6        | bold       | \E[0;3;4m        |
| p7        | invis      | \E[0;8m          |
| p8        | protect    | not available    |
| p9        | altcharset | ^O (off) ^N (on) |

Note that each escape sequence requires a `0` to turn off other modes before turning on its own mode. Also note that, as suggested above, *standout* is set up to be the combination of *reverse* and *dim*. Also, because this terminal has no *bold* mode, *bold* is set up as the combination of *reverse* and *underline*. In addition, to allow combinations, such as *underline+blink*, the sequence to use would be `\E[0;3;5m`. The terminal doesn't have *protect* mode, either, but that cannot be simulated in any way, so p8 is ignored. The *altcharset* mode is different in that it is either `^O` or `^N`, depending on whether it is off or on. If all modes were to be turned on, the sequence would be `\E[0;3;4;5;7;8m^N`.

Now look at when different sequences are output. For example, `;3` is output when either p2 or p6 is true, that is, if either *underline* or *bold* modes are turned on. Writing out the above sequences, along with their dependencies, gives the following:

| sequence | when to output    | terminfo translation |
|----------|-------------------|----------------------|
| \E[0     | always            | \E[0                 |
| ;3       | if p2 or p6       | %%?%p2%p6% %;3%      |
| ;4       | if p1 or p3 or p6 | %%?%p1%p3% p6% %;4%  |
| ;5       | if p4             | %%?%p4%;5%           |
| ;7       | if p1 or p5       | %%?%p1%p5% %;7%      |
| ;8       | if p7             | %%?%p7%;8%           |
| m        | always            | m                    |

| sequence | when to output    | terminfo translation |
|----------|-------------------|----------------------|
| ^N or ^O | if p9 ^N, else ^O | %?%p9?t^N%e^O%       |

Putting this all together into the sgr sequence gives:

```
sgr=\E[0%?%p2%p6%|%t;3%?%p1%p3%|%p6%|%t;4%?%p5%t;5%?%p1%p5%
|%t;7%?%p7%t;8%?%p9?t^N%e^O%,
```

Remember that sgr and sgr0 must always be specified.

**Section 1-8: Keypad** If the device has a keypad that transmits sequences when the keys are pressed, this information can also be specified. Note that it is not possible to handle devices where the keypad only works in local (this applies, for example, to the unshifted Hewlett-Packard 2621 keys). If the keypad can be set to transmit or not transmit, specify these sequences as smkx and rmkx. Otherwise the keypad is assumed to always transmit.

The sequences sent by the left arrow, right arrow, up arrow, down arrow, and home keys can be given as kcub1, kcufl1, kcuu1, kcud1, and khome, respectively. If there are function keys such as f0, f1, ..., f63, the sequences they send can be specified as kf0, kf1, ..., kf63. If the first 11 keys have labels other than the default f0 through f10, the labels can be given as lf0, lf1, ..., lf10. The codes transmitted by certain other special keys can be given: kll (home down), kbs (backspace), ktbc (clear all tabs), kctab (clear the tab stop in this column), kclr (clear screen or erase key), kdch1 (delete character), kdll1 (delete line), krmlr (exit insert mode), kel (clear to end of line), ked (clear to end of screen), kich1 (insert character or enter insert mode), kil1 (insert line), knp (next page), kpp (previous page), kind (scroll forward/down), kri (scroll backward/up), khts (set a tab stop in this column). In addition, if the keypad has a 3 by 3 array of keys including the four arrow keys, the other five keys can be given as ka1, ka3, kb2, kc1, and kc3. These keys are useful when the effects of a 3 by 3 directional pad are needed. Further keys are defined above in the capabilities list.

Strings to program function keys can be specified as pfkey, pflc, and pfx. A string to program screen labels should be specified as pln. Each of these strings takes two parameters: a function key identifier and a string to program it with. pfkey causes pressing the given key to be the same as the user typing the given string; pflc causes the string to be executed by the terminal in local mode; and pfx causes the string to be transmitted to the computer. The capabilities nlab, lw and lh define the number of programmable screen labels and their width and height. If there are commands to turn the labels on and off, give them in smln and rmln. smln is normally output after one or more pln sequences to make sure that the change becomes visible.

**Section 1-9: Tabs and Initialization** If the device has hardware tabs, the command to advance to the next tab stop can be given as ht (usually control I). A "backtab" command that moves leftward to the next tab stop can be given as cbt. By convention, if tty modes show that tabs are being expanded by the computer rather than being sent to the device, programs should not use ht or cbt (even if they are present) because the user may not have the tab stops properly set. If the device has hardware

tabs that are initially set every *n* spaces when the device is powered up, the numeric parameter `it` is given, showing the number of spaces the tabs are set to. This is normally used by `tput init` (see `tput(1)`) to determine whether to set the mode for hardware tab expansion and whether to set the tab stops. If the device has tab stops that can be saved in nonvolatile memory, the `terminfo` description can assume that they are properly set. If there are commands to set and clear tab stops, they can be given as `tbc` (clear all tab stops) and `hts` (set a tab stop in the current column of every row).

Other capabilities include: `is1`, `is2`, and `is3`, initialization strings for the device; `ipro`, the path name of a program to be run to initialize the device; and `if`, the name of a file containing long initialization strings. These strings are expected to set the device into modes consistent with the rest of the `terminfo` description. They must be sent to the device each time the user logs in and be output in the following order: run the program `ipro`; output `is1`; output `is2`; set the margins using `mgc`, `smgl` and `smgr`; set the tabs using `tbc` and `hts`; print the file `if`; and finally output `is3`. This is usually done using the `init` option of `tput`.

Most initialization is done with `is2`. Special device modes can be set up without duplicating strings by putting the common sequences in `is2` and special cases in `is1` and `is3`. Sequences that do a reset from a totally unknown state can be given as `rs1`, `rs2`, `rf`, and `rs3`, analogous to `is1`, `is2`, `is3`, and `if`. (The method using files, `if` and `rf`, is used for a few terminals, from `/usr/share/lib/tabset/*`; however, the recommended method is to use the initialization and reset strings.) These strings are output by `tput reset`, which is used when the terminal gets into a wedged state. Commands are normally placed in `rs1`, `rs2`, `rs3`, and `rf` only if they produce annoying effects on the screen and are not necessary when logging in. For example, the command to set a terminal into 80-column mode would normally be part of `is2`, but on some terminals it causes an annoying glitch on the screen and is not normally needed because the terminal is usually already in 80-column mode.

If a more complex sequence is needed to set the tabs than can be described by using `tbc` and `hts`, the sequence can be placed in `is2` or `if`.

Any margin can be cleared with `mgc`. (For instructions on how to specify commands to set and clear margins, see “Margins” below under “PRINTER CAPABILITIES”.)

Section 1-10: Delays Certain capabilities control padding in the `tty` driver. These are primarily needed by hard-copy terminals, and are used by `tput init` to set `tty` modes appropriately. Delays embedded in the capabilities `cr`, `ind`, `cub1`, `ff`, and `tab` can be used to set the appropriate delay bits to be set in the `tty` driver. If `pb` (padding baud rate) is given, these values can be ignored at baud rates below the value of `pb`.

Section 1-11: Status Lines If the terminal has an extra “status line” that is not normally used by software, this fact can be indicated. If the status line is viewed as an extra line below the bottom line, into which one can cursor address normally (such as the Heathkit `h19`’s 25th line, or the 24th line of a `VT100` which is set to a 23-line scrolling region), the capability `hs` should be given. Special strings that go to a given column of the status line and return from the status line can be given as `tsl` and `fsl`. (`fsl` must leave the cursor position in the same place it was before `tsl`. If necessary, the

sc and rc strings can be included in tsl and fsl to get this effect.) The capability tsl takes one parameter, which is the column number of the status line the cursor is to be moved to.

If escape sequences and other special commands, such as tab, work while in the status line, the flag eslok can be given. A string which turns off the status line (or otherwise erases its contents) should be given as dsl. If the terminal has commands to save and restore the position of the cursor, give them as sc and rc. The status line is normally assumed to be the same width as the rest of the screen, for example, cols. If the status line is a different width (possibly because the terminal does not allow an entire line to be loaded) the width, in columns, can be indicated with the numeric parameter wsl.

Section 1-12: Line  
Graphics

If the device has a line drawing alternate character set, the mapping of glyph to character would be given in acsc. The definition of this string is based on the alternate character set used in the DEC VT100 terminal, extended slightly with some characters from the AT&T 4410v1 terminal.

| Glyph Name              | vt100+ Character |
|-------------------------|------------------|
| arrow pointing right    | +                |
| arrow pointing left     | ,                |
| arrow pointing down     | .                |
| solid square block      | 0                |
| lantern symbol          | I                |
| arrow pointing up       | -                |
| diamond                 | ‘                |
| checker board (stipple) | a                |
| degree symbol           | f                |
| plus/minus              | g                |
| board of squares        | h                |
| lower right corner      | j                |
| upper right corner      | k                |
| upper left corner       | l                |
| lower left corner       | m                |
| plus                    | n                |
| scan line 1             | o                |
| horizontal line         | q                |

| Glyph Name    | vt100+ Character |
|---------------|------------------|
| scan line 9   | s                |
| left tee      | t                |
| right tee     | u                |
| bottom tee    | v                |
| top tee       | w                |
| vertical line | x                |
| bullet        | ~                |

The best way to describe a new device's line graphics set is to add a third column to the above table with the characters for the new device that produce the appropriate glyph when the device is in the alternate character set mode. For example,

| Glyph Name         | vt100+ Char | Newtty Char |
|--------------------|-------------|-------------|
| upper left corner  | l           | R           |
| lower left corner  | m           | F           |
| upper right corner | k           | T           |
| lower right corner | j           | G           |
| horizontal line    | q           | ,           |
| vertical line      | x           | .           |

Now write down the characters left to right, as in “`acs c= lRmFkTjGq\ , x .`”.

In addition, `terminfo` allows you to define multiple character sets. See Section 2-5 for details.

### Section 1-13: Color Manipulation

Let us define two methods of color manipulation: the Tektronix method and the HP method. The Tektronix method uses a set of  $N$  predefined colors (usually 8) from which a user can select “current” foreground and background colors. Thus a terminal can support up to  $N$  colors mixed into  $N \times N$  color-pairs to be displayed on the screen at the same time. When using an HP method the user cannot define the foreground independently of the background, or vice-versa. Instead, the user must define an entire color-pair at once. Up to  $M$  color-pairs, made from  $2 \times M$  different colors, can be defined this way. Most existing color terminals belong to one of these two classes of terminals.

The numeric variables `colors` and `pairs` define the number of colors and color-pairs that can be displayed on the screen at the same time. If a terminal can change the definition of a color (for example, the Tektronix 4100 and 4200 series terminals), this should be specified with `ccc`

(can change color). To change the definition of a color (Tektronix 4200 method), use `initc` (initialize color). It requires four arguments: color number (ranging from 0 to `colors-1`) and three RGB (red, green, and blue) values or three HLS colors (Hue, Lightness, Saturation). Ranges of RGB and HLS values are terminal dependent.

Tektronix 4100 series terminals only use HLS color notation. For such terminals (or dual-mode terminals to be operated in HLS mode) one must define a boolean variable `hls`; that would instruct the curses `init_color` routine to convert its RGB arguments to HLS before sending them to the terminal. The last three arguments to the `initc` string would then be HLS values.

If a terminal can change the definitions of colors, but uses a color notation different from RGB and HLS, a mapping to either RGB or HLS must be developed.

To set current foreground or background to a given color, use `setaf` (set ANSI foreground) and `setab` (set ANSI background). They require one parameter: the number of the color. To initialize a color-pair (HP method), use `initp` (initialize pair). It requires seven parameters: the number of a color-pair (range=0 to `pairs-1`), and six RGB values: three for the foreground followed by three for the background. (Each of these groups of three should be in the order RGB.) When `initc` or `initp` are used, RGB or HLS arguments should be in the order “red, green, blue” or “hue, lightness, saturation”, respectively. To make a color-pair current, use `scp` (set color-pair). It takes one parameter, the number of a color-pair.

Some terminals (for example, most color terminal emulators for PCs) erase areas of the screen with current background color. In such cases, `bce` (background color erase) should be defined. The variable `op` (original pair) contains a sequence for setting the foreground and the background colors to what they were at the terminal start-up time. Similarly, `oc` (original colors) contains a control sequence for setting all colors (for the Tektronix method) or color-pairs (for the HP method) to the values they had at the terminal start-up time.

Some color terminals substitute color for video attributes. Such video attributes should not be combined with colors. Information about these video attributes should be packed into the `ncv` (no color video) variable. There is a one-to-one correspondence between the nine least significant bits of that variable and the video attributes. The following table depicts this correspondence.

| Attribute   | Bit Position | Decimal Value |
|-------------|--------------|---------------|
| A_STANDOUT  | 0            | 1             |
| A_UNDERLINE | 1            | 2             |
| A_REVERSE   | 2            | 4             |
| A_BLINK     | 3            | 8             |
| A_DIM       | 4            | 16            |



| Attribute    | Bit Position | Decimal Value |
|--------------|--------------|---------------|
| A_BOLD       | 5            | 32            |
| A_INVIS      | 6            | 64            |
| A_PROTECT    | 7            | 128           |
| A_ALTCHARSET | 8            | 256           |

When a particular video attribute should not be used with colors, the corresponding `ncv` bit should be set to 1; otherwise it should be set to zero. To determine the information to pack into the `ncv` variable, you must add together the decimal values corresponding to those attributes that cannot coexist with colors. For example, if the terminal uses colors to simulate reverse video (bit number 2 and decimal value 4) and bold (bit number 5 and decimal value 32), the resulting value for `ncv` will be 36 (4 + 32).

Section 1-14: Miscellaneous If the terminal requires other than a null (zero) character as a pad, then this can be given as `pad`. Only the first character of the pad string is used. If the terminal does not have a pad character, specify `npc`.

If the terminal can move up or down half a line, this can be indicated with `hu` (half-line up) and `hd` (half-line down). This is primarily useful for superscripts and subscripts on hardcopy terminals. If a hardcopy terminal can eject to the next page (form feed), give this as `ff` (usually control L).

If there is a command to repeat a given character a given number of times (to save time transmitting a large number of identical characters) this can be indicated with the parameterized string `rep`. The first parameter is the character to be repeated and the second is the number of times to repeat it. Thus, `tparam(repeat_char, 'x', 10)` is the same as `xxxxxxxxxx`.

If the terminal has a settable command character, such as the Tektronix 4025, this can be indicated with `cmdch`. A prototype command character is chosen which is used in all capabilities. This character is given in the `cmdch` capability to identify it. The following convention is supported on some systems: If the environment variable `CC` exists, all occurrences of the prototype character are replaced with the character in `CC`.

Terminal descriptions that do not represent a specific kind of known terminal, such as `switch`, `dialup`, `patch`, and `network`, should include the `gn` (generic) capability so that programs can complain that they do not know how to talk to the terminal. (This capability does not apply to *virtual* terminal descriptions for which the escape sequences are known.) If the terminal is one of those supported by the system virtual terminal protocol, the terminal number can be given as `vt`. A line-turn-around sequence to be transmitted before doing reads should be specified in `rfi`.

If the device uses xon/xoff handshaking for flow control, give xon. Padding information should still be included so that routines can make better decisions about costs, but actual pad characters will not be transmitted. Sequences to turn on and off xon/xoff handshaking may be given in smxon and rmxon. If the characters used for handshaking are not ^S and ^Q, they may be specified with xonc and xoffc.

If the terminal has a "meta key" which acts as a shift key, setting the 8th bit of any character transmitted, this fact can be indicated with km. Otherwise, software will assume that the 8th bit is parity and it will usually be cleared. If strings exist to turn this "meta mode" on and off, they can be given as smm and rmm.

If the terminal has more lines of memory than will fit on the screen at once, the number of lines of memory can be indicated with lm. A value of lm#0 indicates that the number of lines is not fixed, but that there is still more memory than fits on the screen.

Media copy strings which control an auxiliary printer connected to the terminal can be given as mc0: print the contents of the screen, mc4: turn off the printer, and mc5: turn on the printer. When the printer is on, all text sent to the terminal will be sent to the printer. A variation, mc5p, takes one parameter, and leaves the printer on for as many characters as the value of the parameter, then turns the printer off. The parameter should not exceed 255. If the text is not displayed on the terminal screen when the printer is on, specify mc5i (silent printer). All text, including mc4, is transparently passed to the printer while an mc5p is in effect.

#### Section 1-15: Special Cases

The working model used by terminfo fits most terminals reasonably well. However, some terminals do not completely match that model, requiring special support by terminfo. These are not meant to be construed as deficiencies in the terminals; they are just differences between the working model and the actual hardware. They may be unusual devices or, for some reason, do not have all the features of the terminfo model implemented.

Terminals that cannot display tilde (~) characters, such as certain Hazeltine terminals, should indicate hz.

Terminals that ignore a linefeed immediately after an am wrap, such as the Concept 100, should indicate xenl. Those terminals whose cursor remains on the right-most column until another character has been received, rather than wrapping immediately upon receiving the right-most character, such as the VT100, should also indicate xenl.

If el is required to get rid of standout (instead of writing normal text on top of it), xhp should be given.

Those Teleray terminals whose tabs turn all characters moved over to blanks, should indicate xt (destructive tabs). This capability is also taken to mean that it is not possible to position the cursor on top of a "magic cookie." Therefore, to erase standout mode, it is necessary, instead, to use delete and insert line.

Those Beehive Superbee terminals which do not transmit the escape or control-C characters, should specify xsb, indicating that the f1 key is to be used for escape and the f2 key for control C.

- Section 1-16: Similar Terminals If there are two very similar terminals, one can be defined as being just like the other with certain exceptions. The string capability `use` can be given with the name of the similar terminal. The capabilities given before `use` override those in the terminal type invoked by `use`. A capability can be canceled by placing `xx@` to the left of the capability definition, where `xx` is the capability. For example, the entry
- ```
att4424-2|Teletype4424 in display function group ii,
rev@, sgr@, smul@, use=att4424,
```
- defines an AT&T4424 terminal that does not have the `rev`, `sgr`, and `smul` capabilities, and hence cannot do highlighting. This is useful for different modes for a terminal, or for different user preferences. More than one `use` capability may be given.
- PART 2: PRINTER CAPABILITIES The `terminfo` database allows you to define capabilities of printers as well as terminals. To find out what capabilities are available for printers as well as for terminals, see the two lists under “DEVICE CAPABILITIES” that list capabilities by variable and by capability name.
- Section 2-1: Rounding Values Because parameterized string capabilities work only with integer values, we recommend that `terminfo` designers create strings that expect numeric values that have been rounded. Application designers should note this and should always round values to the nearest integer before using them with a parameterized string capability.
- Section 2-2: Printer Resolution A printer's resolution is defined to be the smallest spacing of characters it can achieve. In general printers have independent resolution horizontally and vertically. Thus the vertical resolution of a printer can be determined by measuring the smallest achievable distance between consecutive printing baselines, while the horizontal resolution can be determined by measuring the smallest achievable distance between the left-most edges of consecutive printed, identical, characters.
- All printers are assumed to be capable of printing with a uniform horizontal and vertical resolution. The view of printing that `terminfo` currently presents is one of printing inside a uniform matrix: All characters are printed at fixed positions relative to each “cell” in the matrix; furthermore, each cell has the same size given by the smallest horizontal and vertical step sizes dictated by the resolution. (The cell size can be changed as will be seen later.)
- Many printers are capable of “proportional printing,” where the horizontal spacing depends on the size of the character last printed. `terminfo` does not make use of this capability, although it does provide enough capability definitions to allow an application to simulate proportional printing.
- A printer must not only be able to print characters as close together as the horizontal and vertical resolutions suggest, but also of “moving” to a position an integral multiple of the smallest distance away from a previous position. Thus printed characters can be spaced apart a distance that is an integral multiple of the smallest distance, up to the length or width of a single page.

Some printers can have different resolutions depending on different "modes." In "normal mode," the existing terminfo capabilities are assumed to work on columns and lines, just like a video terminal. Thus the old `lines` capability would give the length of a page in lines, and the `cols` capability would give the width of a page in columns. In "micro mode," many terminfo capabilities work on increments of lines and columns. With some printers the micro mode may be concomitant with normal mode, so that all the capabilities work at the same time.

Section 2-3: Specifying Printer Resolution

The printing resolution of a printer is given in several ways. Each specifies the resolution as the number of smallest steps per distance:

Specification of Printer Resolution Characteristic Number of Smallest Steps

```
orhi  Steps per inch horizontally
orvi  Steps per inch vertically
orc   Steps per column
orl   Steps per line
```

When printing in normal mode, each character printed causes movement to the next column, except in special cases described later; the distance moved is the same as the per-column resolution. Some printers cause an automatic movement to the next line when a character is printed in the rightmost position; the distance moved vertically is the same as the per-line resolution. When printing in micro mode, these distances can be different, and may be zero for some printers.

Specification of Printer Resolution Automatic Motion after Printing

Normal Mode:

```
orc   Steps moved horizontally
orl   Steps moved vertically
```

Micro Mode:

```
mcs   Steps moved horizontally
mls   Steps moved vertically
```

Some printers are capable of printing wide characters. The distance moved when a wide character is printed in normal mode may be different from when a regular width character is printed. The distance moved when a wide character is printed in micro mode may also be different from when a regular character is printed in micro mode, but the differences are assumed to be related: If the distance moved for a regular character is the same whether in normal mode or micro mode (`mcs=orc`), then the distance moved for a wide character is also the same whether in normal mode or micro mode. This doesn't mean the normal character distance is necessarily the same as the wide character distance, just that the distances don't change with a change in normal to micro mode. However, if the distance moved for a regular

character is different in micro mode from the distance moved in normal mode (`mcs < orc`), the micro mode distance is assumed to be the same for a wide character printed in micro mode, as the table below shows.

Specification of Printer Resolution
Automatic Motion after Printing Wide Character

Normal Mode or Micro Mode (`mcs = orc`):
`sp`
`widcs` Steps moved horizontally

Micro Mode (`mcs < orc`):

`mcs` Steps moved horizontally

There may be control sequences to change the number of columns per inch (the character pitch) and to change the number of lines per inch (the line pitch). If these are used, the resolution of the printer changes, but the type of change depends on the printer:

Specification of Printer Resolution
Changing the Character/Line Pitches

`spi` Change character pitch
`cpix` If set, `spi` changes `orhi`, otherwise changes
`orc`
`lpi` Change line pitch
`lpix` If set, `lpi` changes `orvi`, otherwise changes
`orl`
`chr` Change steps per column
`cvr` Change steps per line

The `spi` and `lpi` string capabilities are each used with a single argument, the pitch in columns (or characters) and lines per inch, respectively. The `chr` and `cvr` string capabilities are each used with a single argument, the number of steps per column and line, respectively.

Using any of the control sequences in these strings will imply a change in some of the values of `orc`, `orhi`, `orl`, and `orvi`. Also, the distance moved when a wide character is printed, `widcs`, changes in relation to `orc`. The distance moved when a character is printed in micro mode, `mcs`, changes similarly, with one exception: if the distance is 0 or 1, then no change is assumed (see items marked with * in the following table).

Programs that use `spi`, `lpi`, `chr`, or `cvr` should recalculate the printer resolution (and should recalculate other values— see “Effect of Changing Printing Resolution” under “Dot-Mapped Graphics”).

Specification of Printer Resolution
Effects of Changing the Character/Line Pitches

Before After

Using cpi with cpix clear:

```
$bold orhi '$ orhi
$bold orc '$ $bold orc = bold orhi over V sub italic cpi$
```

Using cpi with cpix set:

```
$bold orhi '$ $bold orhi = bold orc cdot V sub italic cpi$
$bold orc '$ $bold orc$
```

Using lpi with lpix clear:

```
$bold orvi '$ $bold orvi$
$bold orl '$ $bold orl = bold orvi over V sub italic lpi$
```

Using lpi with lpix set:

```
$bold orvi '$ $bold orvi = bold orl cdot V sub italic lpi$
$bold orl '$ $bold orl$
```

Using chr:

```
$bold orhi '$ $bold orhi$
$bold orc '$ $V sub italic chr$
```

Using cvr:

```
$bold orvi '$ $bold orvi$
$bold orl '$ $V sub italic cvr$
```

Using cpi or chr:

```
$bold widcs '$ $bold widcs = bold {widcs ' } bold orc over { bold {orc ' } }$
$bold mcs '$ $bold mcs = bold {mcs ' } bold orc over { bold {orc ' } }$
```

$\$V$ sub italic cpi\$, $\$V$ sub italic lpi\$, $\$V$ sub italic chr\$, and $\$V$ sub italic cvr\$ are the arguments used with cpi, lpi, chr, and cvr, respectively. The prime marks (') indicate the old values.

Section 2-4: Capabilities that Cause Movement

In the following descriptions, “movement” refers to the motion of the “current position.” With video terminals this would be the cursor; with some printers this is the carriage position. Other printers have different equivalents. In general, the current position is where a character would be displayed if printed.

terminfo has string capabilities for control sequences that cause movement a number of full columns or lines. It also has equivalent string capabilities for control sequences that cause movement a number of smallest steps.

String Capabilities for Motion

```
mcub1 Move 1 step left
mcuf1 Move 1 step right
mcuu1 Move 1 step up
mcul1 Move 1 step down
```

```

mcub    Move N steps left
mcuf    Move N steps right
mcuu    Move N steps up
mcud    Move N steps down
mhpa    Move N steps from the left
mvpa    Move N steps from the top

```

The latter six strings are each used with a single argument, *N*.

Sometimes the motion is limited to less than the width or length of a page. Also, some printers don't accept absolute motion to the left of the current position. `terminfo` has capabilities for specifying these limits.

Limits to Motion

```

mjump   Limit on use of mcub1, mcuf1, mcuu1,  mcud1
maddr   Limit on use of mhpa, mvpa
xhpa    If set, hpa and mhpa can't move left
xvpa    If set, vpa and mvpa can't move up

```

If a printer needs to be in a "micro mode" for the motion capabilities described above to work, there are string capabilities defined to contain the control sequence to enter and exit this mode. A boolean is available for those printers where using a carriage return causes an automatic return to normal mode.

Entering/Exiting Micro Mode

```

smicm   Enter micro mode
rmicm   Exit micro mode
crxm    Using cr exits micro mode

```

The movement made when a character is printed in the rightmost position varies among printers. Some make no movement, some move to the beginning of the next line, others move to the beginning of the same line. `terminfo` has boolean capabilities for describing all three cases.

What Happens After Character Printed in Rightmost Position

```

sam     Automatic move to beginning of same line

```

Some printers can be put in a mode where the normal direction of motion is reversed. This mode can be especially useful when there are no capabilities for leftward or upward motion, because those capabilities can be built from the motion reversal capability and the rightward or downward motion capabilities. It is best to leave it up to an application to build the leftward or upward capabilities, though, and not enter them in the `terminfo` database. This allows several reverse motions to be strung together without intervening wasted steps that leave and reenter reverse mode.

Entering/Exiting Reverse Modes

slm	Reverse sense of horizontal motions
rlm	Restore sense of horizontal motions
sum	Reverse sense of vertical motions
rum	Restore sense of vertical motions

While sense of horizontal motions reversed:

mcub1	Move 1 step right
mcuf1	Move 1 step left
mcub	Move N steps right
mcuf	Move N steps left
cub1	Move 1 column right
cuf1	Move 1 column left
cub	Move N columns right
cuf	Move N columns left

While sense of vertical motions reversed:

mcuul	Move 1 step down
mcudl	Move 1 step up
mcuu	Move N steps down
mcud	Move N steps up
cuul	Move 1 line down
cudl	Move 1 line up
cuu	Move N lines down
cud	Move N lines up

The reverse motion modes should not affect the mvpa and mhpa absolute motion capabilities. The reverse vertical motion mode should, however, also reverse the action of the line "wrapping" that occurs when a character is printed in the right-most position. Thus printers that have the standard terminfo capability am defined should experience motion to the beginning of the previous line when a character is printed in the right-most position under reverse vertical motion mode.

The action when any other motion capabilities are used in reverse motion modes is not defined; thus, programs must exit reverse motion modes before using other motion capabilities.

Two miscellaneous capabilities complete the list of new motion capabilities. One of these is needed for printers that move the current position to the beginning of a line when certain control characters, such as "line-feed" or "form-feed," are used. The other is used for the capability of suspending the motion that normally occurs after printing a character.

Miscellaneous Motion Strings

docr	List of control characters causing cr
zerom	Prevent auto motion after printing next single character

Margins `terminfo` provides two strings for setting margins on terminals: one for the left and one for the right margin. Printers, however, have two additional margins, for the top and bottom margins of each page. Furthermore, some printers require not using motion strings to move the current position to a margin and then fixing the margin there, but require the specification of where a margin should be regardless of the current position. Therefore `terminfo` offers six additional strings for defining margins with printers.

Setting Margins

```

smgl   Set left margin at current column
smgr   Set right margin at current column
smgb   Set bottom margin at current line
smgt   Set top margin at current line
smgbp  Set bottom margin at line N
smglp  Set left margin at column N
smgrp  Set right margin at column N
smgtp  Set top margin at line N

```

The last four strings are used with one or more arguments that give the position of the margin or margins to set. If both of `smglp` and `smgrp` are set, each is used with a single argument, *N*, that gives the column number of the left and right margin, respectively. If both of `smgtp` and `smgbp` are set, each is used to set the top and bottom margin, respectively: `smgtp` is used with a single argument, *N*, the line number of the top margin; however, `smgbp` is used with two arguments, *N* and *M*, that give the line number of the bottom margin, the first counting from the top of the page and the second counting from the bottom. This accommodates the two styles of specifying the bottom margin in different manufacturers' printers. When coding a `terminfo` entry for a printer that has a settable bottom margin, only the first or second parameter should be used, depending on the printer. When writing an application that uses `smgbp` to set the bottom margin, both arguments must be given.

If only one of `smglp` and `smgrp` is set, then it is used with two arguments, the column number of the left and right margins, in that order. Likewise, if only one of `smgtp` and `smgbp` is set, then it is used with two arguments that give the top and bottom margins, in that order, counting from the top of the page. Thus when coding a `terminfo` entry for a printer that requires setting both left and right or top and bottom margins simultaneously, only one of `smglp` and `smgrp` or `smgtp` and `smgbp` should be defined; the other should be left blank. When writing an application that uses these string capabilities, the pairs should be first checked to see if each in the pair is set or only one is set, and should then be used accordingly.

In counting lines or columns, line zero is the top line and column zero is the left-most column. A zero value for the second argument with `smgbp` means the bottom line of the page.

All margins can be cleared with `mgc`.

Shadows, Italics, Wide Characters Five new sets of strings describe the capabilities printers have of enhancing printed text.
Enhanced Printing

```
sshm   Enter shadow-printing mode
rshm   Exit shadow-printing mode
sitm   Enter italicizing mode
ritm   Exit italicizing mode
swidm  Enter wide character mode
rwidm  Exit wide character mode
ssupm  Enter superscript mode
rsupd  m Exit superscript mode
supcs  List of characters available as superscripts
ssubm  Enter subscript mode
rsubm  Exit subscript mode
subcs  List of characters available as subscripts
```

If a printer requires the `sshm` control sequence before every character to be shadow-printed, the `rshm` string is left blank. Thus programs that find a control sequence in `sshm` but none in `rshm` should use the `sshm` control sequence before every character to be shadow-printed; otherwise, the `sshm` control sequence should be used once before the set of characters to be shadow-printed, followed by `rshm`. The same is also true of each of the `sitm/ritm`, `swidm/rwidm`, `ssupm/rsupm`, and `ssubm/rsubm` pairs.

Note that `terminfo` also has a capability for printing emboldened text (`bold`). While shadow printing and emboldened printing are similar in that they “darken” the text, many printers produce these two types of print in slightly different ways. Generally, emboldened printing is done by overstriking the same character one or more times. Shadow printing likewise usually involves overstriking, but with a slight movement up and/or to the side so that the character is “fatter.”

It is assumed that enhanced printing modes are independent modes, so that it would be possible, for instance, to shadow print italicized subscripts.

As mentioned earlier, the amount of motion automatically made after printing a wide character should be given in `widcs`.

If only a subset of the printable ASCII characters can be printed as superscripts or subscripts, they should be listed in `supcs` or `subcs` strings, respectively. If the `ssupm` or `ssubm` strings contain control sequences, but the corresponding `supcs` or `subcs` strings are empty, it is assumed that all printable ASCII characters are available as superscripts or subscripts.

Automatic motion made after printing a superscript or subscript is assumed to be the same as for regular characters. Thus, for example, printing any of the following three examples will result in equivalent motion:

```
Bi Bi Bi
```

Note that the existing `msg_r` boolean capability describes whether motion control sequences can be used while in “standout mode.” This capability is extended to cover the enhanced printing modes added here. `msg_r` should be set for those printers that accept any motion control sequences without affecting shadow, italicized, widened, superscript, or subscript printing. Conversely, if `msg_r` is not set, a program should end these modes before attempting any motion.

Section 2-5: Alternate Character Sets

In addition to allowing you to define line graphics (described in Section 1-12), `terminfo` lets you define alternate character sets. The following capabilities cover printers and terminals with multiple selectable or definable character sets.

Alternate Character Sets

```

scs      Select character set N
scsd     Start definition of character set N, M characters
defc     Define character A, B dots wide, descender D
rcsd     End definition of character set N
csnm     List of character set names
daisy    Printer has manually changed print-wheels

```

The `scs`, `rcsd`, and `csnm` strings are used with a single argument, *N*, a number from 0 to 63 that identifies the character set. The `scsd` string is also used with the argument *N* and another, *M*, that gives the number of characters in the set. The `defc` string is used with three arguments: *A* gives the ASCII code representation for the character, *B* gives the width of the character in dots, and *D* is zero or one depending on whether the character is a “descender” or not. The `defc` string is also followed by a string of “image-data” bytes that describe how the character looks (see below).

Character set 0 is the default character set present after the printer has been initialized. Not every printer has 64 character sets, of course; using `scs` with an argument that doesn't select an available character set should cause a null result from `tparam`.

If a character set has to be defined before it can be used, the `scsd` control sequence is to be used before defining the character set, and the `rcsd` is to be used after. They should also cause a null result from `tparam` when used with an argument *N* that doesn't apply. If a character set still has to be selected after being defined, the `scs` control sequence should follow the `rcsd` control sequence. By examining the results of using each of the `scs`, `scsd`, and `rcsd` strings with a character set number in a call to `tparam`, a program can determine which of the three are needed.

Between use of the `scsd` and `rcsd` strings, the `defc` string should be used to define each character. To print any character on printers covered by `terminfo`, the ASCII code is sent to the printer. This is true for characters in an alternate set as well as “normal” characters. Thus the definition of a character includes the ASCII code that represents it. In addition, the width of the character in dots is given, along with an indication of whether the character should descend below the print line (such as the lower case letter “g” in most character sets). The width of the character in dots also indicates the number of image-data bytes that will follow

the `defc` string. These image-data bytes indicate where in a dot-matrix pattern ink should be applied to “draw” the character; the number of these bytes and their form are defined below under “Dot-Mapped Graphics.”

It's easiest for the creator of `terminfo` entries to refer to each character set by number; however, these numbers will be meaningless to the application developer. The `csnm` string alleviates this problem by providing names for each number.

When used with a character set number in a call to `tparm`, the `csnm` string will produce the equivalent name. These names should be used as a reference only. No naming convention is implied, although anyone who creates a `terminfo` entry for a printer should use names consistent with the names found in user documents for the printer. Application developers should allow a user to specify a character set by number (leaving it up to the user to examine the `csnm` string to determine the correct number), or by name, where the application examines the `csnm` string to determine the corresponding character set number.

These capabilities are likely to be used only with dot-matrix printers. If they are not available, the strings should not be defined. For printers that have manually changed print-wheels or font cartridges, the boolean `daisy` is set.

Section 2-6: Dot-Matrix Graphics Dot-matrix printers typically have the capability of reproducing “raster-graphics” images. Three new numeric capabilities and three new string capabilities can help a program draw raster-graphics images independent of the type of dot-matrix printer or the number of pins or dots the printer can handle at one time.

Dot-Matrix Graphics

<code>npins</code>	Number of pins, <i>N</i> , in print-head
<code>spinv</code>	Spacing of pins vertically in pins per inch
<code>spinh</code>	Spacing of dots horizontally in dots per inch
<code>porder</code>	Matches software bits to print-head pins
<code>sbim</code>	Start printing bit image graphics, <i>B</i> bits wide
<code>rbim</code>	End printing bit image graphics

The `sbim` string is used with a single argument, *B*, the width of the image in dots.

The model of dot-matrix or raster-graphics that `terminfo` presents is similar to the technique used for most dot-matrix printers: each pass of the printer's print-head is assumed to produce a dot-matrix that is *N* dots high and *B* dots wide. This is typically a wide, squat, rectangle of dots. The height of this rectangle in dots will vary from one printer to the next; this is given in the `npins` numeric capability. The size of the rectangle in fractions of an inch will also vary; it can be deduced from the `spinv` and `spinh` numeric capabilities. With these three values an application can divide a complete raster-graphics image into several horizontal strips, perhaps interpolating to account for different dot spacing vertically and horizontally.

The `sbim` and `rbim` strings are used to start and end a dot-matrix image, respectively. The `sbim` string is used with a single argument that gives the width of the dot-matrix in dots. A sequence

of “image-data bytes” are sent to the printer after the `sbim` string and before the `rbim` string. The number of bytes is an integral multiple of the width of the dot-matrix; the multiple and the form of each byte is determined by the `porder` string as described below.

The `porder` string is a comma separated list of pin numbers optionally followed by an numerical offset. The offset, if given, is separated from the list with a semicolon. The position of each pin number in the list corresponds to a bit in an 8-bit data byte. The pins are numbered consecutively from 1 to `npins`, with 1 being the top pin. Note that the term “pin” is used loosely here; “ink-jet” dot-matrix printers don't have pins, but can be considered to have an equivalent method of applying a single dot of ink to paper. The bit positions in `porder` are in groups of 8, with the first position in each group the most significant bit and the last position the least significant bit. An application produces 8-bit bytes in the order of the groups in `porder`.

An application computes the “image-data bytes” from the internal image, mapping vertical dot positions in each print-head pass into 8-bit bytes, using a 1 bit where ink should be applied and 0 where no ink should be applied. This can be reversed (0 bit for ink, 1 bit for no ink) by giving a negative pin number. If a position is skipped in `porder`, a 0 bit is used. If a position has a lower case ‘x’ instead of a pin number, a 1 bit is used in the skipped position. For consistency, a lower case ‘o’ can be used to represent a 0 filled, skipped bit. There must be a multiple of 8 bit positions used or skipped in `porder`; if not, 0 bits are used to fill the last byte in the least significant bits. The offset, if given, is added to each data byte; the offset can be negative.

Some examples may help clarify the use of the `porder` string. The AT&T 470, AT&T 475 and C.Itoh 8510 printers provide eight pins for graphics. The pins are identified top to bottom by the 8 bits in a byte, from least significant to most. The `porder` strings for these printers would be 8,7,6,5,4,3,2,1. The AT&T 478 and AT&T 479 printers also provide eight pins for graphics. However, the pins are identified in the reverse order. The `porder` strings for these printers would be 1,2,3,4,5,6,7,8. The AT&T 5310, AT&T 5320, DEC LA100, and DEC LN03 printers provide six pins for graphics. The pins are identified top to bottom by the decimal values 1, 2, 4, 8, 16 and 32. These correspond to the low six bits in an 8-bit byte, although the decimal values are further offset by the value 63. The `porder` string for these printers would be , ,6,5,4,3,2,1;63, or alternately o,o,6,5,4,3,2,1;63.

Section 2-7: Effect of Changing Printing Resolution

If the control sequences to change the character pitch or the line pitch are used, the pin or dot spacing may change:

Dot-Matrix Graphics
Changing the Character/Line Pitches

```

cpi      Change character pitch
cpix     If set, cpi changes spinh
lpi      Change line pitch
lpix     If set, lpi changes spinv

```

Programs that use `cpi` or `lpi` should recalculate the dot spacing:

Dot-Matrix Graphics

Effects of Changing the Character/Line Pitches

Before	After
--------	-------

Using `cpi` with `cpix` clear:

<code>\$bold spinh '\$</code>	<code>\$bold spinh\$</code>
-------------------------------	-----------------------------

Using `cpi` with `cpix` set:

<code>\$bold spinh '\$</code>	<code>\$bold spinh = bold spinh ' cdot bold orhi over</code>
	<code>{ bold {orhi ' } }\$</code>

Using `lpi` with `lpix` clear:

<code>\$bold spinv '\$</code>	<code>\$bold spinv\$</code>
-------------------------------	-----------------------------

Using `lpi` with `lpix` set:

<code>\$bold spinv '\$</code>	<code>\$bold spinv = bold {spinv ' } cdot bold orhi over</code>
	<code>{ bold {orhi ' } }\$</code>

Using `chr`:

<code>\$bold spinh '\$</code>	<code>\$bold spinh\$</code>
-------------------------------	-----------------------------

Using `cvr`:

<code>\$bold spinv '\$</code>	<code>\$bold spinv\$</code>
-------------------------------	-----------------------------

`orhi'` and `orhi` are the values of the horizontal resolution in steps per inch, before using `cpi` and after using `cpi`, respectively. Likewise, `orvi'` and `orvi` are the values of the vertical resolution in steps per inch, before using `lpi` and after using `lpi`, respectively. Thus, the changes in the dots per inch for dot-matrix graphics follow the changes in steps per inch for printer resolution.

Section 2-8: Print Quality

Many dot-matrix printers can alter the dot spacing of printed text to produce near “letter quality” printing or “draft quality” printing. Usually it is important to be able to choose one or the other because the rate of printing generally falls off as the quality improves. There are three new strings used to describe these capabilities.

Print Quality

<code>snlq</code>	Set near-letter quality print
<code>snrmq</code>	Set normal quality print
<code>sdrfq</code>	Set draft quality print

The capabilities are listed in decreasing levels of quality. If a printer doesn't have all three levels, one or two of the strings should be left blank as appropriate.

Section 2-9: Printing Rate and Buffer Size

Because there is no standard protocol that can be used to keep a program synchronized with a printer, and because modern printers can buffer data before printing it, a program generally cannot determine at any time what has been printed. Two new numeric capabilities can help a program estimate what has been printed.

Print Rate/Buffer Size

`cps` Nominal print rate in characters per second
`bufsz` Buffer capacity in characters

`cps` is the nominal or average rate at which the printer prints characters; if this value is not given, the rate should be estimated at one-tenth the prevailing baud rate. `bufsz` is the maximum number of subsequent characters buffered before the guaranteed printing of an earlier character, assuming proper flow control has been used. If this value is not given it is assumed that the printer does not buffer characters, but prints them as they are received.

As an example, if a printer has a 1000-character buffer, then sending the letter “a” followed by 1000 additional characters is guaranteed to cause the letter “a” to print. If the same printer prints at the rate of 100 characters per second, then it should take 10 seconds to print all the characters in the buffer, less if the buffer is not full. By keeping track of the characters sent to a printer, and knowing the print rate and buffer size, a program can synchronize itself with the printer.

Note that most printer manufacturers advertise the maximum print rate, not the nominal print rate. A good way to get a value to put in for `cps` is to generate a few pages of text, count the number of printable characters, and then see how long it takes to print the text.

Applications that use these values should recognize the variability in the print rate. Straight text, in short lines, with no embedded control sequences will probably print at close to the advertised print rate and probably faster than the rate in `cps`. Graphics data with a lot of control sequences, or very long lines of text, will print at well below the advertised rate and below the rate in `cps`. If the application is using `cps` to decide how long it should take a printer to print a block of text, the application should pad the estimate. If the application is using `cps` to decide how much text has already been printed, it should shrink the estimate. The application will thus err in favor of the user, who wants, above all, to see all the output in its correct place.

Files	<code>/usr/share/lib/terminfo/??/*</code>	compiled terminal description database
	<code>/usr/share/lib/.COREterm/??/*</code>	subset of compiled terminal description database
	<code>/usr/share/lib/tabset/*</code>	tab settings for some terminals, in a format appropriate to be output to the terminal (escape sequences that set margins and tabs)

See Also [ls\(1\)](#), [pg\(1\)](#), [stty\(1\)](#), [tput\(1\)](#), [tty\(1\)](#), [vi\(1\)](#), [infocmp\(1M\)](#), [tic\(1M\)](#), [printf\(3C\)](#), [curses\(3CURSES\)](#), [curses\(3XCURSES\)](#)

Notes The most effective way to prepare a terminal description is by imitating the description of a similar terminal in `terminfo` and to build up a description gradually, using partial descriptions with a screen oriented editor, such as `vi`, to check that they are correct. To easily

test a new terminal description the environment variable `TERMINFO` can be set to the pathname of a directory containing the compiled description, and programs will look there rather than in `/usr/share/lib/terminfo`.

Name TIMEZONE – set default system time zone and locale

Synopsis /etc/TIMEZONE
/etc/default/init

Description This file sets the time zone environment variable TZ, and the locale-related environment variables LANG, LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, LC_NUMERIC, and LC_TIME.

/etc/TIMEZONE is a symbolic link to /etc/default/init.

The number of environment variables that can be set from /etc/default/init is limited to 20.

The format of the file is:

VAR=value

where *VAR* is a timezone environment variable and *value* is the value assigned to the variable. *value* can be enclosed in double quotes (") or single quotes ('). The double or single quotes cannot be part of the value.

See Also [init\(1M\)](#), [rtc\(1M\)](#), [ctime\(3C\)](#), [environ\(5\)](#)

Notes When changing the TZ setting on x86 systems, you must make a corresponding change to the /etc/rtc_config file to account for the new timezone setting. This can be accomplished by executing the following commands, followed by a reboot, to make the changes take effect:

```
# rtc -z zone-name
# rtc -c
```

where *zone-name* is the same name as the TZ variable setting.

See [rtc\(1M\)](#) for information on the rtc command.

Name timezone – default timezone data base

Synopsis /etc/timezone

Description The timezone file contains information regarding the default timezone for each host in a domain. Alternatively, a single default line for the entire domain may be specified. Each entry has the format:

Timezone-name *official-host-or-domain-name*

Items are separated by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. The timezone is a pathname relative to the directory /usr/share/lib/zoneinfo.

This file is not actually referenced by any system software; it is merely used as a source file to construct the NIS `timezone.byname` map. This map is read by installation software to initialize the timezone of the client system at installation time.

The `timezone` file does not set the timezone environment variable `TZ`. See [TIMEZONE\(4\)](#) for information to set the `TZ` environment variable.

Examples **EXAMPLE 1** Typical timezone line

Here is a typical line from the /etc/timezone file:

```
US/Eastern           East.Sun.COM #Sun East Coast
```

Files /etc/timezone

See Also [TIMEZONE\(4\)](#)

Installing Oracle Solaris 11 Systems

Name TrustedExtensionsPolicy – configuration file for Trusted Extensions X Server Extension

Synopsis /usr/lib/xorg/TrustedExtensionsPolicy

Description TrustedExtensionsPolicy is the configuration file for Trusted Extensions X Server Extension (SUN_TSOL). SUN_TSOL provides security policy enforcement. This enforcement is based on Mandatory Access Control (MAC) and Discretionary Access Control (DAC).

Blank lines and comments in the TrustedExtensionsPolicy file are ignored. Comments start with a pound sign (#). The format of the file is as follows:

keyword{space|tab}*value*

where *keyword* can be one of the following:

atom Label this atom ADMIN_LOW, so that XGetAtomName(3X11) succeeds.

property Instantiate this property once. The default is to polyinstantiate a property.

selection Polyinstantiate this selection. The default is to instantiate the selection once.

extension Disable this extension.

privilege Implicitly allow this window privilege on all clients.

For possible keyword values, see the /usr/lib/xorg/TrustedExtensionsPolicy file for the Xorg X server.

Examples The following entry in the TrustedExtensionsPolicy file polyinstantiates the Dtpad program:

```
selection Dtpad
```

If the entry is missing, or commented out, the Dtpad program is instantiated once.

Similarly, the following entry instantiates the WM_ICON_SIZE property once:

```
property WM_ICON_SIZE
```

If the entry is missing, or commented out, the WM_ICON_SIZE property is polyinstantiated.

Files /usr/lib/xorg/TrustedExtensionsPolicy Configuration file for Trusted Extensions X Server Extension

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	x11/trusted/libxtsol
Interface Stability	Committed

See Also [XGetAtomName\(3X11\)](#), [attributes\(5\)](#)

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

Name ts_dptbl – time-sharing dispatcher parameter table

Description The process scheduler (or dispatcher) is the portion of the kernel that controls allocation of the CPU to processes. The scheduler supports the notion of scheduling classes where each class defines a scheduling policy, used to schedule processes within that class. Associated with each scheduling class is a set of priority queues on which ready to run processes are linked. These priority queues are mapped by the system configuration into a set of global scheduling priorities which are available to processes within the class. (The dispatcher always selects for execution the process with the highest global scheduling priority in the system.) The priority queues associated with a given class are viewed by that class as a contiguous set of priority levels numbered from 0 (lowest priority) to n (highest priority—a configuration-dependent value). The set of global scheduling priorities that the queues for a given class are mapped into might not start at zero and might not be contiguous (depending on the configuration).

Processes in the time-sharing class which are running in user mode (or in kernel mode before going to sleep) are scheduled according to the parameters in a time-sharing dispatcher parameter table (ts_dptbl). Processes in the inter-active scheduling class are also scheduled according to the parameters in the time-sharing dispatcher parameter table. (Time-sharing processes and inter-active processes running in kernel mode after sleeping are run within a special range of priorities reserved for such processes and are not affected by the parameters in the ts_dptbl until they return to user mode.) The ts_dptbl consists of an array (config_ts_dptbl[]) of parameter structures (struct tsdpent_t), one for each of the n priority levels used by time-sharing processes and inter-active processes in user mode. The structures are accessed via a pointer, (ts_dptbl), to the array. The properties of a given priority level i are specified by the i th parameter structure in this array (ts_dptbl[i]).

A parameter structure consists of the following members. These are also described in the /usr/include/sys/ts.h header.

ts_globpri	The global scheduling priority associated with this priority level. The mapping between time-sharing priority levels and global scheduling priorities is determined at boot time by the system configuration. ts_globpri is the only member of the ts_dptbl which cannot be changed with <code>dispadm(1M)</code> .
ts_quantum	The length of the time quantum allocated to processes at this level in ticks (hz). In the high resolution clock mode (hi_res_tick set to 1), the value of hz is set to 1000. Increase quantum to maintain the same absolute time quantum.
ts_tqexp	Priority level of the new queue on which to place a process running at the current level if it exceeds its time quantum. Normally this field links to a lower priority time-sharing level that has a larger quantum.

<code>ts_slpret</code>	Priority level of the new queue on which to place a process, that was previously in user mode at this level, when it returns to user mode after sleeping. Normally this field links to a higher priority level that has a smaller quantum.
<code>ts_maxwait</code>	A per process counter, <code>ts_dispwait</code> is initialized to zero each time a time-sharing or inter-active process is placed back on the dispatcher queue after its time quantum has expired or when it is awakened (<code>ts_dispwait</code> is not reset to zero when a process is preempted by a higher priority process). This counter is incremented once per second for each process on a dispatcher or sleep queue. If a process' <code>ts_dispwait</code> value exceeds the <code>ts_maxwait</code> value for its level, the process' priority is changed to that indicated by <code>ts_lwait</code> . The purpose of this field is to prevent starvation.
<code>ts_lwait</code>	Move a process to this new priority level if <code>ts_dispwait</code> is greater than <code>ts_maxwait</code> .

An administrator can affect the behavior of the time-sharing portion of the scheduler by reconfiguring the `ts_dptbl`. Since processes in the time-sharing and inter-active scheduling classes share the same dispatch parameter table (`ts_dptbl`), changes to this table will affect both scheduling classes. There are two methods available for doing this: reconfigure with a loadable module at boot-time or by using `dispadmin(1M)` at run-time.

`ts_dptbl` Loadable Module The `ts_dptbl` can be reconfigured with a loadable module which contains a new time sharing dispatch table. The module containing the dispatch table is separate from the TS loadable module which contains the rest of the time-sharing and inter-active software. This is the only method that can be used to change the number of time-sharing priority levels or the set of global scheduling priorities used by the time-sharing and inter-active classes. The relevant procedure and source code is described in the REPLACING THE TS_DPTBL LOADABLE MODULE section.

`dispadmin` Configuration File With the exception of `ts_globpri` all of the members of the `ts_dptbl` can be examined and modified on a running system using the `dispadmin(1M)` command. Invoking `dispadmin` for the time-sharing or inter-active class allows the administrator to retrieve the current `ts_dptbl` configuration from the kernel's in-core table, or overwrite the in-core table with values from a configuration file. The configuration file used for input to `dispadmin` must conform to the specific format described below.

Blank lines are ignored and any part of a line to the right of a `#` symbol is treated as a comment. The first non-blank, non-comment line must indicate the resolution to be used for interpreting the `ts_quantum` time quantum values. The resolution is specified as

`RES=res`

where *res* is a positive integer between 1 and 1,000,000,000 inclusive and the resolution used is the reciprocal of *res* in seconds (for example, `RES=1000` specifies millisecond resolution).

Although very fine (nanosecond) resolution may be specified, the time quantum lengths are rounded up to the next integral multiple of the system clock's resolution.

The remaining lines in the file are used to specify the parameter values for each of the time-sharing priority levels. The first line specifies the parameters for time-sharing level 0, the second line specifies the parameters for time-sharing level 1, etc. There must be exactly one line for each configured time-sharing priority level.

Examples EXAMPLE 1 A Sample From a Configuration File

The following excerpt from a `dispadmin` configuration file illustrates the format. Note that for each line specifying a set of parameters there is a comment indicating the corresponding priority level. These level numbers indicate priority within the time-sharing and interactive classes, and the mapping between these time-sharing priorities and the corresponding global scheduling priorities is determined by the configuration specified in the `ts` master file. The level numbers are strictly for the convenience of the administrator reading the file and, as with any comment, they are ignored by `dispadmin`. `dispadmin` assumes that the lines in the file are ordered by consecutive, increasing priority level (from 0 to the maximum configured time-sharing priority). The level numbers in the comments should normally agree with this ordering; if for some reason they don't, however, `dispadmin` is unaffected.

```
# Time-Sharing Dispatcher Configuration File RES=1000

# ts_quantum  ts_tqexp  ts_slpret  ts_maxwait  ts_lwait  PRIORITY
#                                                    LEVEL
500           0         10         5           10        # 0
500           0         11         5           11        # 1
500           1         12         5           12        # 2
500           1         13         5           13        # 3
500           2         14         5           14        # 4
500           2         15         5           15        # 5
450           3         16         5           16        # 6
450           3         17         5           17        # 7
.             .           .           .           .         . .
.             .           .           .           .         . .
.             .           .           .           .         . .
50           48         59         5           59        # 58
50           49         59         5           59        # 59
```

EXAMPLE 2 Replacing The `ts_dptbl` Loadable Module

In order to change the size of the time sharing dispatch table, the loadable module which contains the dispatch table information will have to be built. It is recommended that you save the existing module before using the following procedure.

1. Place the dispatch table code shown below in a file called `ts_dptbl.c`. An example of this file follows.

EXAMPLE 2 Replacing The ts_dptbl Loadable Module *(Continued)*

2. Compile the code using the given compilation and link lines supplied.

```
cc -c -O -D_KERNEL
ts_dptbl.c
ld -r -o TS_DPTBL ts_dptbl.o
```

3. Copy the current dispatch table in /kernel/sched to TS_DPTBL.bak.
4. Replace the current TS_DPTBL in /kernel/sched.
5. You will have to make changes in the /etc/system file to reflect the changes to the sizes of the tables. See [system\(4\)](#). The two variables affected are ts_maxupri and ts_maxkmdpri. The syntax for setting these is as follows:

```
set TS:ts_maxupri=(value for max time-sharing user priority)
set TS:ts_maxkmdpri=(number of kernel mode priorities - 1)
```

6. Reboot the system to use the new dispatch table.

Great care should be used in replacing the dispatch table using this method. If you do not get it right, panics may result, thus making the system unusable.

The following is an example of a ts_dptbl.c file used for building the new ts_dptbl.

```
/* BEGIN ts_dptbl.c */
#include <sys/proc.h>
#include <sys/priocntl.h>
#include <sys/class.h>
#include <sys/disp.h>
#include <sys/ts.h>
#include <sys/rtpriocntl.h>
/*
 * This is the loadable module wrapper.
 */
#include <sys/modctl.h>
extern struct mod_ops mod_miscops;
/*
 * Module linkage information for the kernel.
 */
static struct modlmisc modlmisc = {
    &mod_miscops, "Time sharing dispatch table"
};
static struct modlinkage modlinkage = {
    MODREV_1, &modlmisc, 0
};
_init()
{
    return (mod_install(&modlinkage));
}
```


EXAMPLE 2 Replacing The ts_dptbl Loadable Module (Continued)

```

}
_info(modinfo)
    struct modinfo *modinfo;
{
    return (mod_info(&modlinkage, modinfo));
}
/*
 * array of global priorities used by ts procs sleeping or
 * running in kernel mode after sleep. Must have at least
 * 40 values.
 */
pri_t config_ts_kmdpris[] = {
    60,61,62,63,64,65,66,67,68,69,
    70,71,72,73,74,75,76,77,78,79,
    80,81,82,83,84,85,86,87,88,89,
    90,91,92,93,94,95,96,97,98,99,
};
tsdpent_t    config_ts_dptbl[] = {
/*  glbpri  qntm  tqexp  slprt  mxwt  lwt  */
    0,    100,  0,    10,    5,    10,
    1,    100,  0,    11,    5,    11,
    2,    100,  1,    12,    5,    12,
    3,    100,  1,    13,    5,    13,
    4,    100,  2,    14,    5,    14,
    5,    100,  2,    15,    5,    15,
    6,    100,  3,    16,    5,    16,
    7,    100,  3,    17,    5,    17,
    8,    100,  4,    18,    5,    18,
    9,    100,  4,    19,    5,    19,
    10,   80,   5,    20,    5,    20,
    11,   80,   5,    21,    5,    21,
    12,   80,   6,    22,    5,    22,
    13,   80,   6,    23,    5,    23,
    14,   80,   7,    24,    5,    24,
    15,   80,   7,    25,    5,    25,
    16,   80,   8,    26,    5,    26,
    17,   80,   8,    27,    5,    27,
    18,   80,   9,    28,    5,    28,
    19,   80,   9,    29,    5,    29,
    20,   60,  10,    30,    5,    30,
    21,   60,  11,    31,    5,    31,
    22,   60,  12,    32,    5,    33,
    24,   60,  14,    34,    5,    34,

```

EXAMPLE 2 Replacing The ts_dptbl Loadable Module *(Continued)*

```

25,    60,   15,   35,    5,   35,
26,    60,   16,   36,    5,   36,
27,    60,   17,   37,    5,   37,
28,    60,   18,   38,    5,   38,
29,    60,   19,   39,    5,   39,
30,    40,   20,   40,    5,   40,
31,    40,   21,   41,    5,   41,
32,    40,   22,   42,    5,   42,
33,    40,   23,   43,    5,   43,
34,    40,   24,   44,    5,   44,
35,    40,   25,   45,    5,   45,
36,    40,   26,   46,    5,   46,
37,    40,   27,   47,    5,   47,
38,    40,   28,   48,    5,   48,
39,    40,   29,   49,    5,   49,
40,    20,   30,   50,    5,   50,
41,    20,   31,   50,    5,   50,
42,    20,   32,   51,    5,   51,
43,    20,   33,   51,    5,   51,
44,    20,   34,   52,    5,   52,
45,    20,   35,   52,    5,   52,
46,    20,   36,   53,    5,   53,
47,    20,   37,   53,    5,   53,
48,    20,   38,   54,    5,   54,
49,    20,   39,   54,    5,   54,
50,    10,   40,   55,    5,   55,
51,    10,   41,   55,    5,   55,
52,    10,   42,   56,    5,   56,
53,    10,   43,   56,    5,   56,
54,    10,   44,   57,    5,   57,
55,    10,   45,   57,    5,   57,
56,    10,   46,   58,    5,   58,
57,    10,   47,   58,    5,   58,
58,    10,   48,   59,    5,   59,
59,    10,   49,   59,    5,   59,

};

short config_ts_maxumdpri = sizeof (config_ts_dptbl)/16 - 1;
/*
 * Return the address of config_ts_dptbl
 */
tsdpent_t *
ts_getdptbl()
{

```

EXAMPLE 2 Replacing The ts_dptbl Loadable Module (Continued)

```

        return (config_ts_dptbl);
    }

    /*
     * Return the address of config_ts_kmdpris
     */
    int *
    ts_getkmdpris()
    {
        return (config_ts_kmdpris);
    }

    /*
     * Return the address of ts_maxumdpri
     */
    short
    ts_getmaxumdpri()
    {
        return (config_ts_maxumdpri);
    }

    /* END ts_dptbl.c */

```

See Also [prioctl\(1\)](#), [dispadm\(1M\)](#), [prioctl\(2\)](#), [system\(4\)](#)

Oracle Solaris Administration: Common Tasks

Programming Interfaces Guide

Notes `dispadm` does some limited sanity checking on the values supplied in the configuration file. The sanity checking is intended to ensure that the new `ts_dptbl` values do not cause the system to panic. The sanity checking does not attempt to analyze the effect that the new values will have on the performance of the system. Unusual `ts_dptbl` configurations may have a dramatic negative impact on the performance of the system.

No sanity checking is done on the `ts_dptbl` values specified in the `TS_DPTBL` loadable module. Specifying an inconsistent or nonsensical `ts_dptbl` configuration through the `TS_DPTBL` loadable module could cause serious performance problems and/or cause the system to panic.

Name `ttydefs` – file contains terminal line settings information for `ttymon`

Description `/etc/ttydefs` is an administrative file that contains records divided into fields by colons (":"). This information used by `ttymon` to set up the speed and terminal settings for a TTY port.

The `ttydefs` file contains the following fields:

<i>ttylabel</i>	The string <code>ttymon</code> tries to match against the TTY port's <i>ttylabel</i> field in the port monitor administrative file. It often describes the speed at which the terminal is supposed to run, for example, <code>1200</code> .
<i>initial-flags</i>	Contains the initial <code>termio(7I)</code> settings to which the terminal is to be set. For example, the system administrator will be able to specify what the default erase and kill characters will be. <i>initial-flags</i> must be specified in the syntax recognized by the <code>stty</code> command.
<i>final-flags</i>	<i>final-flags</i> must be specified in the same format as <i>initial-flags</i> . <code>ttymon</code> sets these final settings after a connection request has been made and immediately prior to invoking a port's service.
<i>autobaud</i>	If the <i>autobaud</i> field contains the character 'A,' <i>autobaud</i> will be enabled. Otherwise, <i>autobaud</i> will be disabled. <code>ttymon</code> determines what line speed to set the TTY port to by analyzing the carriage returns entered. If <i>autobaud</i> has been disabled, the hunt sequence is used for baud rate determination.
<i>nextlabel</i>	If the user indicates that the current terminal setting is not appropriate by sending a BREAK, <code>ttymon</code> searches for a <code>ttydefs</code> entry whose <i>ttylabel</i> field matches the <i>nextlabel</i> field. If a match is found, <code>ttymon</code> uses that field as its <i>ttylabel</i> field. A series of speeds is often linked together in this way into a closed set called a hunt sequence. For example, <code>4800</code> may be linked to <code>1200</code> , which in turn is linked to <code>2400</code> , which is finally linked to <code>4800</code> .

See Also [sttydefs\(1M\)](#), [ttymon\(1M\)](#), [termio\(7I\)](#)

Oracle Solaris Administration: Common Tasks

Name ttysrch – directory search list for ttyname

Description `ttysrch` is an optional file that is used by the `ttyname` library routine. This file contains the names of directories in `/dev` that contain terminal and terminal-related device files. The purpose of this file is to improve the performance of `ttyname` by indicating which subdirectories in `/dev` contain terminal-related device files and should be searched first. These subdirectory names must appear on separate lines and must begin with `/dev`. Those path names that do not begin with `/dev` will be ignored and a warning will be sent to the console. Blank lines (lines containing only white space) and lines beginning with the comment character `"#"` will be ignored. For each file listed (except for the special entry `/dev`), `ttyname` will recursively search through subdirectories looking for a match. If `/dev` appears in the `ttysrch` file, the `/dev` directory itself will be searched but there will not be a recursive search through its subdirectories.

When `ttyname` searches through the device files, it tries to find a file whose major/minor device number, file system identifier, and inode number match that of the file descriptor it was given as an argument. If a match is not found, it will settle for a match of just major/minor device and file system identifier, if one can be found. However, if the file descriptor is associated with a cloned device, this algorithm does not work efficiently because the inode number of the device file associated with a clonable device will never match the inode number of the file descriptor that was returned by the `open` of that clonable device. To help with these situations, entries can be put into the `/etc/ttysrch` file to improve performance when cloned devices are used as terminals on a system (for example, for remote login). However, this is only useful if the minor devices related to a cloned device are put into a subdirectory. (It is important to note that device files need not exist for cloned devices and if that is the case, `ttyname` will eventually fail.) An optional second field is used in the `/etc/ttysrch` file to indicate the matching criteria. This field is separated by white space (any combination of blanks or tabs). The letter `M` means major/minor device number, `F` means file system identifier, and `I` means inode number. If this field is not specified for an entry, the default is `MFI` which means try to match on all three. For cloned devices the field should be `MF`, which indicates that it is not necessary to match on the inode number.

Without the `/etc/ttysrch` file, `ttyname` will search the `/dev` directory by first looking in the directories `/dev/term`, `/dev/pts`, and `/dev/xt`. If a system has terminal devices installed in directories other than these, it may help performance if the `ttysrch` file is created and contains that list of directories.

Examples **EXAMPLE 1** A sample display of `/etc/ttysrch` command.

A sample `/etc/ttysrch` file follows:

```
/dev/term    MFI
/dev/pts     MFI
/dev/xt      MFI
/dev/slan    MF
```

EXAMPLE 1 A sample display of `/etc/ttysrch` command. *(Continued)*

This file tells `ttyname` that it should first search through those directories listed and that when searching through the `/dev/slan` directory, if a file is encountered whose major/minor devices and file system identifier match that of the file descriptor argument to `ttyname`, this device name should be considered a match.

Files `/etc/ttysrch`

See Also [ttyname\(3C\)](#)

Name ufsdump, dumpdates – incremental dump format

Synopsis #include <sys/types.h>
 #include <sys/inode.h>
 #include <protocols/dumprestore.h>
 /etc/dumpdates

Description Tapes used by `ufsdump(1M)` and `ufsrestore(1M)` contain:

- a header record
- two groups of bit map records
- a group of records describing directories
- a group of records describing files

The format of the header record and the format of the first record of each description in the <protocols/dumprestore.h> include file are:

```
#define TP_BSIZE_MAX    65536
#define TP_BSIZE_MIN    1024
#define ESIZE_SHIFT_MAX 6

#ifdef SUPPORTS_MTB_TAPE_FORMAT
#define TP_BUFSIZE      TP_BSIZE_MAX
#else
#define TP_BSIZE        1024
#define TP_BUFSIZE      TP_BSIZE
#endif /* SUPPORTS_MTB_TAPE_FORMAT */

#define NTREC           10
#define HIGHDENSITYTREC 32
#define CARTRIDGETREC  63
#define TP_NINDIR       (TP_BSIZE_MIN/2)
#define TP_NINOS        (TP_NINDIR / sizeof (long))
#define LBSIZE          16
#define NAMELEN         64

#define OFS_MAGIC       (int)60011
#define NFS_MAGIC       (int)60012
#define MTB_MAGIC       (int)60013
#define CHECKSUM        (int)84446

union u_data {
    char    s_addr[TP_NINDIR];
    int32_t s_inos[TP_NINOS];
};

union u_shadow {
```

```
    struct s_nonsh {
        int32_t c_level;
        char    c_filesys[NAMELEN];
        char    c_dev[NAMELEN];
        char    c_host[NAMELEN];
    } c_nonsh;
    char    c_shadow[1];
};

union u_spcl {
    char dummy[TP_BUFSIZE];
    struct s_spcl {
        int32_t c_type;
        time32_t c_date;
        time32_t c_ddate;
        int32_t c_volume;
        daddr32_t c_tapea;
        ino32_t c_inumber;
        int32_t c_magic;
        int32_t c_checksum;
        struct dinode c_dinode;
        int32_t c_count;
        union u_data c_data;
        char    c_label[LBSIZE];
        union u_shadow c_shadow;
        int32_t c_flags;
        int32_t c_firstrec;
#ifdef SUPPORTS_MTB_TAPE_FORMAT
        int32_t c_tpbsize;
        int32_t c_spare[31];
#else
        int32_t c_spare[32];
#endif /* SUPPORTS_MTB_TAPE_FORMAT */
    } s_spcl;
} u_spcl;

int32_t          c_type;
time32_t         c_date;
time32_t         c_ddate;
int32_t          c_volume;
daddr32_t        c_tapea;
ino32_t          c_inumber;
int32_t          c_magic;
int32_t          c_checksum;
struct dinode    c_dinode;
int32_t          c_count;
union            u_data c_data;
char             c_label[LBSIZE];
```



```

union                u_shadow c_shadow;
int32_t             c_flags;
int32_t             c_firstrec;
#ifdef SUPPORTS_MTB_TAPE_FORMAT
int32_t             c_tpbsize;
int32_t             c_spare[31];
#else
int32_t             c_spare[32];
#endif
/*
   SUPPORTS_MTB_TAPE_FORMAT */

    } s_spcl;
} u_spcl;
#define spcl u_spcl.s_spcl
#define c_addr c_data.s_addrs
#define c_inos c_data.s_inos
#define c_level c_shadow.c_nonsh.c_level
#define c_filesys c_shadow.c_nonsh.c_filesys
#define c_dev c_shadow.c_nonsh.c_dev
#define c_host c_shadow.c_nonsh.c_host

#define TS_TAPE      1
#define TS_INODE     2
#define TS_ADDR      4
#define TS_BITS      3
#define TS_CLRI      6
#define TS_END       5
#define TS_EOM       7

#define DR_NEWHEADER 1
#define DR_INODEINFO 2
#define DR_REDUMP    4
#define DR_TRUEINC   8
#define DR_HASMETA   16

```

This header describes three formats for the `ufsdump/ufsrestore` interface:

- An old format, non-MTB, that supports dump sizes of less than 2 terabytes. This format is represented by `NFS_MAGIC`.
- A new format, MTB, that supports dump sizes of greater than 2 terabytes using a variable block size and 2 new constants: `TP_BSIZE_MIN` and `TP_BSIZE_MAX`. This format is represented by `MTB_MAGIC`.
- A much older format that might be found on existing backup tapes. The `ufsrestore` command can restore tapes of this format, but no longer generates tapes of this format. Backups in this format have the `OFS_MAGIC` magic number in their tape headers.

The constants are described as follows:

TP_BSIZE	Size of file blocks on the dump tapes for the old format. Note that TP_BSIZE must be a multiple of DEV_BSIZE. This is applicable for dumps of type NFS_MAGIC or OFS_MAGIC, but is not applicable for dumps of type MTB_MAGIC.
TP_BSIZE_MIN	Minimum size of file blocks on the dump tapes for the new MTB format (MTB_MAGIC) only.
TP_BSIZE_MAX	Maximum size of file blocks on the dump tapes for the new MTB format (MTB_MAGIC) only.
NTREC	Number of TP_BSIZE blocks that are written in each tape record.
HIGHDENSITYNTREC	Number of TP_BSIZE blocks that are written in each tape record on 6250 BPI or higher density tapes.
CARTRIDGETREC	Number of TP_BSIZE blocks that are written in each tape record on cartridge tapes.
TP_NINDIR	Number of indirect pointers in a TS_INODE or TS_ADDR record. It must be a power of 2.
TP_NINOS	The maximum number of volumes on a tape.
LBSIZE	The maximum size of a volume label.
NAMELEN	The maximum size of a host's name.
OFS_MAGIC	Magic number that is used for the very old format.
NFS_MAGIC	Magic number that is used for the non-MTB format.
MTB_MAGIC	Magic number that is used for the MTB format.
CHECKSUM	Header records checksum to this value.

The TS_ entries are used in the c_type field to indicate what sort of header this is. The types and their meanings are as follows:

TS_TAPE	Tape volume label.
TS_INODE	A file or directory follows. The c_dinode field is a copy of the disk inode and contains bits telling what sort of file this is.
TS_ADDR	A subrecord of a file description. See s_addrs below.
TS_BITS	A bit map follows. This bit map has a one bit for each inode that was dumped.
TS_CLRI	A bit map follows. This bit map contains a zero bit for all inodes that were empty on the file system when dumped.
TS_END	End of tape record.

The flags are described as follows:

DR_NEWHEADER	New format tape header.
DR_INFODEINFO	Header contains starting inode info.
DR_REDUMP	Dump contains recopies of active files.
DR_TRUEINC	Dump is a “true incremental”.
DR_HASMETA	The metadata in this header.
DUMPOUTFMT	Name, incon, and ctime (date) for printf.
DUMPINFMT	Inverse for scanf.

The fields of the header structure are as follows:

s_addr	An array of bytes describing the blocks of the dumped file. A byte is zero if the block associated with that byte was not present on the file system; otherwise, the byte is non-zero. If the block was not present on the file system, no block was dumped; the block will be stored as a hole in the file. If there is not sufficient space in this record to describe all the blocks in a file, TS_ADDR records will be scattered through the file, each one picking up where the last left off
s_inos	The starting inodes on tape.
c_type	The type of the record.
c_date	The date of the previous dump.
c_ddate	The date of this dump.
c_volume	The current volume number of the dump.
c_tapea	The logical block of this record.
c_inumber	The number of the inode being dumped if this is of type TS_INODE.
c_magic	This contains the value MAGIC above, truncated as needed.
c_checksum	This contains whatever value is needed to make the record sum to CHECKSUM.
c_dinode	This is a copy of the inode as it appears on the file system.
c_count	The count of bytes in s_addr.
u_data c_data	The union of either u_data c_data The union of either s_addr or s_inos.
c_label	Label for this dump.
c_level	Level of this dump.

c_filesys	Name of dumped file system.
c_dev	Name of dumped service.
c_host	Name of dumped host.
c_flags	Additional information.
c_firstrec	First record on volume.
c_spare	Reserved for future uses.
c_tpbsize	Tape block size for MTB format only.

Each volume except the last ends with a tapemark (read as an end of file). The last volume ends with a TS_END record and then the tapemark.

The dump history is kept in the file /etc/dumpdates. It is an ASCII file with three fields separated by white space:

- The name of the device on which the dumped file system resides.
- The level number of the dump tape; see [ufsdump\(1M\)](#).
- The date of the incremental dump in the format generated by [ctime\(3C\)](#).

DUMPOUTFMT is the format to use when using [printf\(3C\)](#) to write an entry to /etc/dumpdates; DUMPINFMT is the format to use when using [scanf\(3C\)](#) to read an entry from /etc/dumpdates.

Attributes See [attributes\(5\)](#) for a description of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Uncommitted

See Also [ufsdump\(1M\)](#), [ufsrestore\(1M\)](#), [ctime\(3C\)](#), [printf\(3C\)](#), [scanf\(3C\)](#), [types.h\(3HEAD\)](#), [attributes\(5\)](#),

Name updaters – configuration file for NIS updating

Synopsis /var/yp/updaters

Description The file /var/yp/updaters is a makefile (see [make\(1S\)](#)) which is used for updating the Network Information Service (NIS) databases. Databases can only be updated in a secure network, that is, one that has a [publickey\(4\)](#) database. Each entry in the file is a make target for a particular NIS database. For example, if there is an NIS database named passwd.byname that can be updated, there should be a make target named passwd.byname in the updaters file with the command to update the file.

The information necessary to make the update is passed to the update command through standard input. The information passed is described below (all items are followed by a NEWLINE except for 4 and 6):

1. Network name of client wishing to make the update (a string).
2. Kind of update (an integer).
3. Number of bytes in key (an integer).
4. Actual bytes of key.
5. Number of bytes in data (an integer).
6. Actual bytes of data.

After receiving this information through standard input, the command to update the particular database determines whether the user is allowed to make the change. If not, it exits with the status YPERR_ACCESS. If the user is allowed to make the change, the command makes the change and exits with a status of zero. If there are any errors that may prevent the updaters from making the change, it should exit with the status that matches a valid NIS error code described in <rpcsvc/ypclnt.h>.

Files /var/yp/updaters The makefile used for updating the NIS databases.

See Also [make\(1S\)](#), [rpc.yppupdated\(1M\)](#), [publickey\(4\)](#)

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 may not be used without permission.

Name user_attr – extended user attributes database

Synopsis /etc/user_attr

Description /etc/user_attr is a local source of extended attributes associated with users and roles. user_attr can be used with other user attribute sources, including the LDAP people container and the user_attr NIS map. Programs use the [getuserattr\(3C\)](#) routines to gain access to this information.

The search order for multiple user_attr sources is specified in the /etc/nsswitch.conf file, as described in the [nsswitch.conf\(4\)](#) man page. The search order follows that for [passwd\(4\)](#).

Each entry in the user_attr databases consists of a single line with five fields separated by colons (:). Line continuations using the backslash (\) character are permitted. Each entry has the form:

```
user:qualifier:res1:res2:attr
```

user

The name of the user as specified in the [passwd\(4\)](#) database.

qualifier

Reserved for future use.

res1

Reserved for future use.

res2

Reserved for future use.

attr

An optional list of semicolon-separated (;) key-value pairs that describe the security attributes to apply to the object upon execution. Zero or more keys can be specified. The following keys are currently interpreted by the system:

audit_flags

Specifies per-user audit preselection flags as colon-separated always-audit-flags and never-audit-flags. As in, `audit_flags=always-audit-flags:never-audit-flags`. See [audit_flags\(5\)](#).

auths

Specifies a comma-separated list of authorization names chosen from those names defined in the [auth_attr\(4\)](#) database. Authorization names can be specified using the asterisk (*) character as a wildcard. For example, `solaris.print.*` means all of Oracle Solaris' printer authorizations.

All of the authorizations from profiles are available to the user.

defaultpriv

The default set of privileges assigned to a user's inheritable set upon login. See [Privileges Keywords](#).

limitpriv

The maximum set of privileges a user or any process started by the user, whether through [su\(1M\)](#) or any other means, can obtain. See [Privileges Keywords](#).

lock_after_retries

Specifies whether an account is locked after the count of failed logins for a user equals or exceeds the allowed number of retries as defined by [RETRIES](#) in [/etc/default/login](#). Possible values are `yes` or `no`. The default is `no`. Account locking is applicable only to local accounts and accounts in the ldap name service repository. LDAP account must be configured with an [enableShadowUpdate](#) of `true` as specified in [ldapclient\(1M\)](#).

profiles

Contains an ordered, comma-separated list of profile names chosen from [prof_attr\(4\)](#). Profiles are enforced by the profile shells. See [pfexec\(1\)](#). A list of profiles can also be defined in the [/etc/security/policy.conf](#) file. See [policy.conf\(4\)](#). If no profiles are assigned, the profile shells do not allow the user to execute any commands.

project

Can be assigned a name of one project from the [project\(4\)](#) database to be used as a default project to place the user in at login time. For more information, see [getdefaultproj\(3PROJECT\)](#).

roleauth

Specifies whether the assigned role requires a role password or the password of the user who is assuming the role.

Valid values are `role` and `user`. If `roleauth` is not specified, `roleauth=role` is implied.

roles

Can be assigned a comma-separated list of role names from the set of user accounts in this database whose `type` field indicates the account is a role. If the `roles` key value is not specified, the user is not permitted to assume any role.

type

Can be assigned one of these strings: `normal`, indicating that this account is for a normal user, one who logs in; or `role`, indicating that this account is for a role. Roles can only be assumed by a normal user after the user has logged in.

The following keys are available only if the system is configured with the Trusted Extensions feature:

clearance

Contains the maximum label at which the user can operate. If unspecified, in the Defense Intelligence Agency (DIA) encodings scheme, the default is specified in [label_encodings\(4\)](#).

idlecmd

Contains one of two keywords that the Trusted Extensions window manager interprets when a workstation is idle for too long. The keyword `lock` specifies that the workstation is to be locked (thus requiring the user to re-authenticate to resume the session). The keyword `logout` specifies that session is to be terminated (thus, killing the user's processes launched in the current session). If unspecified, the default value, `lock`, is in effect.

idletime

Contains a number representing the maximum number of minutes a workstation can remain idle before the Trusted Extensions window manager attempts the task specified in `idlecmd`. A zero in this field specifies that the `idlecmd` command is never executed. If no value is specified, the default `idletime` of 30 minutes is in effect.

min_label

Contains the minimum label at which the user can log in. If unspecified, in the DIA encodings scheme, the default is specified in `label_encodings(4)`.

Except for the type `key`, the `key=value` fields in the `user_attr` database can be added using `roleadd(1M)` and `useradd(1M)`. You can use `rolemod(1M)` and `usermod(1M)` to modify these values. Modification of the type `key` is restricted as described in `rolemod` and `usermod`.

The values assigned to the `auths`, `roles`, and `profiles` keywords are cumulative. To assign the values, `/etc/user_attr` is searched first, followed by each of the profiles, in order. The other keywords (`audit_flags`, `project`, `defaultpriv`, `limitpriv`, `lock_after_retries`, `idletime`, `idlecmd`, `clearance` and `min_label`) are first matched, meaning that `/etc/user_attr` is searched first, followed by each of the profiles, in order. Once a match is found that search is over.

Privileges Keywords See [privileges\(5\)](#) for a description of privileges. The command `ppriv -l` (see [ppriv\(1\)](#)) produces a list of all supported privileges. You specify privileges as they are displayed by `ppriv`. In [privileges\(5\)](#), privileges are listed in the form `PRIV_<privilege_name>`. For example, the privilege `file_chown`, as you would specify it in `user_attr`, is listed in [privileges\(5\)](#) as `PRIV_FILE_CHOWN`.

Privileges can be specified through [usermod\(1M\)](#) and [rolemod\(1M\)](#). See [usermod\(1M\)](#) for examples of commands that modify privileges and their subsequent effect on `user_attr`.

The following authorizations are required to set the various keywords:

<code>audit_flags</code>	<code>solaris.audit.assign</code>
<code>auths</code>	<code>solaris.auth.delegate/assign</code>
<code>clearance</code>	<code>solaris.label.delegate</code>
<code>defaultpriv</code>	<code>solaris.privilege.delegate/assign</code>
<code>idlecmd</code>	<code>solaris.session.setpolicy</code>
<code>idletime</code>	<code>solaris.session.setpolicy</code>
<code>limitpriv</code>	<code>solaris.privilege.delegate/assign</code>
<code>lock_after_retries</code>	<code>solaris.passwd.setpolicy</code>

min_label	solaris.label.delegate
profiles	solaris.profile.delegate/assign
project	solaris.project.delegate/assign
roles	solaris.role.delegate/assign

The `solaris.auth.assign` authorization allows an authorized user to grant any authorization to another user. The `solaris.auth.delegate` allows an authorized user to grant only the user's authorizations to another user. The same principle applies to `roles`, `profiles`, `privileges`, and `project`.

The `clearance` and `min_label` values can only be set based on the authorized user's label range. The `defaultpriv` and `limitpriv` values can only be set based on the authorized user's granted `defaultpriv` and `limitpriv` privileges.

Examples **EXAMPLE 1** Assigning a Profile to Root

The following example entry assigns to root the All profile, which allows root to use all commands in the system, and also assigns all authorizations:

```
root:::auths=solaris.*;profiles=All;type=normal
```

The `solaris.*` wildcard authorization gives root all of the `solaris` authorizations. See [auth_attr\(4\)](#) for more about authorizations.

Files /etc/nsswitch.conf
See [nsswitch.conf\(4\)](#).

/etc/user_attr
Locally added entries. The shipped header must remain intact.

/etc/user_attr.d/*
Entries added by package installation.

Attributes See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	system/core-os
Interface Stability	See below.

The command-line syntax is Committed. The output is Uncommitted.

See Also [auths\(1\)](#), [pexec\(1\)](#), [ppriv\(1\)](#), [profiles\(1\)](#), [roles\(1\)](#), [userattr\(1\)](#), [getent\(1M\)](#), [ldapclient\(1M\)](#), [roleadd\(1M\)](#), [rolemod\(1M\)](#), [useradd\(1M\)](#), [usermod\(1M\)](#), [getdefaultproj\(3PROJECT\)](#), [getuserattr\(3C\)](#), [auth_attr\(4\)](#), [exec_attr\(4\)](#), [label_encodings\(4\)](#), [nsswitch.conf\(4\)](#), [passwd\(4\)](#), [policy.conf\(4\)](#), [prof_attr\(4\)](#), [project\(4\)](#), [attributes\(5\)](#), [audit_flags\(5\)](#), [privileges\(5\)](#)

Oracle Solaris Administration: Security Services

Notes The root user is usually defined in local databases for a number of reasons, including the fact that root needs to be able to log in and do system maintenance in single-user mode, before the network name service databases are available. For this reason, an entry should exist for root in the local `user_attr` file, and the precedence shown in the example `nsswitch.conf(4)` file entry under `EXAMPLES` is highly recommended.

Because the list of legal keys is likely to expand, any code that parses this database must be written to ignore unknown key-value pairs without error. When any new keywords are created, the names should be prefixed with a unique string, such as the company's stock symbol, to avoid potential naming conflicts.

This file should not be edited. Values are changed using `useradd(1M)` and `usermod(1M)`.

A user without an entry in `user_attr` gets the default values as defined in `/etc/security/policy.conf`.

Name utmp, wtmp – utmp and wtmp database entry formats

Synopsis `#include <utmp.h>`
`/var/adm/utmp`
`/var/adm/wtmp`

Description The utmp and wtmp database files are obsolete and are no longer present on the system. They have been superseded by the extended database contained in the utmpx and wtmpx database files. See [utmpx\(4\)](#).

It is possible for `/var/adm/utmp` to reappear on the system. This would most likely occur if a third party application that still uses utmp recreates the file if it finds it missing. This file should not be allowed to remain on the system. The user should investigate to determine which application is recreating this file.

See Also [utmpx\(4\)](#)

Name utmpx, wtmpx – utmpx and wtmpx database entry formats

Synopsis `#include <utmpx.h>`
`/var/adm/utmpx`
`/var/adm/wtmpx`

Description The utmpx and wtmpx files are extended database files that have superseded the obsolete utmp and wtmp database files.

The utmpx database contains user access and accounting information for commands such as [who\(1\)](#), [write\(1\)](#), and [login\(1\)](#). The wtmpx database contains the history of user access and accounting information for the utmpx database.

Usage Applications should not access these databases directly, but should use the functions described on the [getutxent\(3C\)](#) manual page to interact with the utmpx and wtmpx databases to ensure that they are maintained consistently.

Files `/var/adm/utmpx` user access and administration information
`/var/adm/wtmpx` history of user access and administrative information

See Also [getutxent\(3C\)](#), [wait\(3C\)](#), [wait.h\(3HEAD\)](#)

Name vfstab – table of file system defaults

Description The file `/etc/vfstab` describes defaults for each file system. The information is stored in a table with the following column headings:

device to mount	device to fsck	mount point	FS type	fsck pass	mount at boot	mount options
--------------------	-------------------	----------------	------------	--------------	------------------	------------------

The fields in the table are space-separated and show the resource name (*device to mount*), the raw device to fsck (*device to fsck*), the default mount directory (*mount point*), the name of the file system type (*FS type*), the number used by fsck to decide whether to check the file system automatically (*fsck pass*), whether the file system should be mounted automatically by `mountall` (*mount at boot*), and the file system mount options (*mount options*). (See respective mount file system man page below in SEE ALSO for *mount options*.) A '-' is used to indicate no entry in a field. This can be used when a field does not apply to the resource being mounted.

The `getvfsent(3C)` family of routines is used to read and write to `/etc/vfstab`.

`/etc/vfstab` can be used to specify swap areas. An entry so specified, (which can be a file or a device), automatically is added as a swap area by the `/usr/sbin/swapadd` script when the system boots. To specify a swap area, the *device-to-mount* field contains the name of the swap file or device, the *FS-type* is `swap`, *mount-at-boot* is `no` and all other fields have no entry. The presence in `/etc/vfstab` of the `encrypted` option, specified for swap device that is a ZFS volume or a raw device, enables encryption on that device. For a ZFS volume, encryption is enabled by the ZFS encryption property (see `zfs(1M)`); for a raw device, encryption is enabled by means of `lofi(7D)`.

iSCSI LUN can only be mounted after the iSCSI initiator SMF service, `svc:/network/iscsi/initiator`, is started. Set the *mount at boot* entries for iSCSI LUN in `/etc/vfstab` to `iscsi` instead of `yes`. This enables the iSCSI initiator SMF service to attempt to mount iSCSI LUN later.

Examples The following are `vfstab` entries for various file system types supported in the Solaris operating environment.

EXAMPLE 1 NFS and UFS Mounts

The following entry invokes NFS to automatically mount the directory `/usr/local` of the server `example1` on the client's `/usr/local` directory with read-only permission:

```
example1:/usr/local - /usr/local nfs - yes ro
```

The following example assumes a small departmental mail setup, in which clients mount `/var/mail` from a server `mailsvr`. The following entry would be listed in each client's `vfstab`:

```
mailsvr:/var/mail - /var/mail nfs - yes intr,bg
```

The following is an example for a UFS file system in which logging is enabled:

EXAMPLE 1 NFS and UFS Mounts *(Continued)*

```
/dev/dsk/c2t10d0s0 /dev/rdisk/c2t10d0s0 /export/local ufs 3 yes logging
```

See [mount_nfs\(1M\)](#) for a description of NFS mount options and [mount_ufs\(1M\)](#) for a description of UFS options.

EXAMPLE 2 pcfs Mounts

The following example mounts a pcfs file system on a fixed hard disk on an x86 machine:

```
/dev/dsk/c1t2d0p0:c - /win98 pcfs - yes -
```

The example below mounts a Jaz drive on a SPARC machine. Normally, the volume management software handles mounting of removable media, obviating a vfstab entry. Specifying a device that supports removable media in vfstab with set the mount-at-boot field to no (as shown below) disables the automatic handling of that device. Such an entry presumes you are not running volume management software.

```
/dev/dsk/c1t2d0s2:c - /jaz pcfs - no -
```

For removable media on a SPARC machine, the convention for the slice portion of the disk identifier is to specify s2, which stands for the entire medium.

For pcfs file systems on x86 machines, note that the disk identifier uses a p (p0) and a logical drive (c, in the /win98 example above) for a pcfs logical drive. See [mount_pcfs\(1M\)](#) for syntax for pcfs logical drives and for pcfs-specific mount options.

EXAMPLE 3 Loopback File System Mount

The following is an example of mounting a loopback (lofs) file system:

```
/export/test - /opt/test lofs - yes -
```

See [lofs\(7FS\)](#) for an overview of the loopback file system.

See Also [fsck\(1M\)](#), [mount\(1M\)](#), [mount_hsf\(1M\)](#), [mount_nfs\(1M\)](#), [mount_tmpfs\(1M\)](#), [mount_ufs\(1M\)](#), [swap\(1M\)](#), [zfs\(1M\)](#), [getvfsent\(3C\)](#), [lofi\(7D\)](#)

Oracle Solaris Administration: Common Tasks

-
- Name** volume-config – Solaris Volume Manager volume configuration information for top down volume creation with metassist
- Synopsis** /usr/share/lib/xml/dtd/volume-config.dtd
- Description** A volume configuration file, XML-based and compliant with the volume-config.dtd Document Type Definition, describes the detailed configuration of the volume or volumes to be created, including the names, sizes and configurations of all the components used in the volume or volumes. This configuration file can be automatically generated by running metassist with the -d option, or can be manually created.
- The volume configuration file can then be used to either generate a command file or to directly create volumes by running metassist and specifying the volume configuration file as input to the command.
- As a system administrator, you would want to change, manually create, or edit the volume configuration file only if there are small details of the configuration that you want to change. For example, you might want to change names for volumes or hot spare pools, mirror read option, or stripe interlace values.
- It would be possible to also select different devices or change slice sizes or make similar changes, but that is generally not recommended. Substantial changes to the volume-config file could result in a poor or non-functional configuration.
- With a volume-config file, you can run metassist and provide the file as input to the command to generate either a command file or to actually set up the configuration.
- Defining Volume Configuration** The top level element <volume-config> surrounds the volume configuration data. This element has no attributes. A volume configuration requires exactly one <diskset> element, which must be the first element of the volume configuration. Additionally, the volume-config can have zero or more of the following elements: <disk>, <slice>, <hsp>, <concat>, <stripe>, <mirror> as required to define the configuration of the volume to be created.
- Defining Disk Set** Within the <volume-config> element, a <diskset> element must exist. The <diskset> element, with the name attribute, specifies the name of the diskset in which to create the volume or volumes. This element and attribute are required. If this named disk set does not exist, it is created upon implementation of this volume configuration.
- Defining Slice** The volume configuration format provides for a <slice> element that defines the name of a slice to use as a component of a volume. The <slice> element requires a name attribute which specifies a full ctd name. If the <slice> is newly created as part of the volume configuration, the startsector and sizeinblocks attributes must be specified. If the slice was previously existing, these attributes need not be specified.

Defining Hot Spare Pool The `<hsp>` element that defines the name of a hot spare pool to use as a component of a configuration. The `<hsp>` element requires a name attribute which specifies a hot spare pool name.

Slices defined by `<slice>` elements contained in the `<hsp>` element are included in the hot spare pool when `metastat` creates it.

Defining Stripe The `<stripe>` element defines stripes (interlaced RAID 0 volumes) to be used in a volume. The `<stripe>` element takes a required name attribute to specify a name conforming to Solaris Volume Manager naming requirements. If the name specifies an existing stripe, no `<slice>` elements are required. If the name specifies a new stripe, the `<slice>` elements to construct the slice must be specified within the `<stripe>` element. The `<stripe>` element takes an optional `interlace` attribute as value and units (for example, 16KB, 5BLOCKS, 20MB). If this value isn't specified, the Solaris Volume Manager default value is used.

Defining Concat The `<concat>` element defines concats (non-interlaced RAID 0 volumes) to be used in a configuration. It is the same as a `<stripe>` element, except that the `interlace` attribute is not valid.

Defining Mirror The `<mirror>` element defines mirrors (RAID 1 volumes) to be used in a volume configuration. It can contain combinations of `<concat>` and `<stripe>` elements (to explicitly determine which volumes are used as submirrors).

The `<mirror>` element takes a required name attribute to specify a name conforming to Solaris Volume Manager naming requirements.

The `<mirror>` element takes an optional `read` attribute to define the mirror read options (ROUNDROBIN, GEOMETRIC, or FIRST) for the mirrors. If this attribute is not specified, the Solaris Volume Manager default value is used.

The `<mirror>` element takes an optional `write` attribute to define the mirror write options (PARALLEL, SERIAL, or FIRST) for the mirrors. If this attribute is not specified, the Solaris Volume Manager default value is used. The `<mirror>` element takes an optional `passnum` attribute (0-9) to define the mirror passnum that defines the order in which mirrors are resynced at boot, if required. Smaller numbers are resynced first. If this attribute is not specified, the Solaris Volume Manager default value is used.

Examples **EXAMPLE 1** Specifying a Volume Configuration

The following is an example volume configuration:

```
<!-- Example configuration -->
<volume-config>
  <!-- Specify the existing disk set to use -->
  <diskset name="redundant"/>

  <!-- Create slices -->
```


EXAMPLE 1 Specifying a Volume Configuration *(Continued)*

```

<slice name="/dev/dsk/c0t0d1s7" startsector="1444464" \
    sizeinblocks="205632BLOCKS"/>
<slice name="/dev/dsk/c0t0d1s6" startsector="1239840" \
    sizeinblocks="102816KB"/>

<!-- Create a concat -->
<concat name="d12">
<slice name="/dev/dsk/c0t0d0s7"/>
<slice name="/dev/dsk/c0t0d0s6"/>
<slice name="/dev/dsk/c0t0d1s7"/>
<slice name="/dev/dsk/c0t0d1s6"/>

<!-- Create (and use) a HSP -->
hsp name="hsp0">
<slice name="/dev/dsk/c0t0d4s0"/>
<slice name="/dev/dsk/c0t0d4s1"/>
<slice name="/dev/dsk/c0t0d4s3"/>
<slice name="/dev/dsk/c0t0d4s4"/>
</hsp>

</concat>

<!-- Create a stripe -->
<stripe name="d15" interlace="32KB">
<slice name="/dev/dsk/c0t0d0s7"/>
<slice name="/dev/dsk/c0t0d1s7"/>

<!-- Use a previously-defined HSP -->
<hsp name="hsp0"/>
</stripe>

<!-- Create a mirror -->
<mirror name="d10">

<!-- Submirror 1: An existing stripe -->
<stripe name="d11"/>

<!-- Submirror 2: The concat defined above -->
<concat name="d12"/>

<!-- Submirror 3: A stripe defined here -->
<stripe name="d13">
<slice name="/dev/dsk/c0t0d2s6"/>
<slice name="/dev/dsk/c0t0d2s7"/>
<slice name="/dev/dsk/c0t0d3s6"/>

```

EXAMPLE 1 Specifying a Volume Configuration *(Continued)*

```
slice name="/dev/dsk/c0t0d3s7"/>
</stripe>

</mirror>

</volume-config>
```

Files ■ /usr/share/lib/xml/dtd/volume-config.dtd

See Also [metassist\(1M\)](#), [metaclear\(1M\)](#), [metadb\(1M\)](#), [metadetach\(1M\)](#), [metahs\(1M\)](#), [metainit\(1M\)](#), [metaoffline\(1M\)](#), [metaonline\(1M\)](#), [metaparam\(1M\)](#), [metarecover\(1M\)](#), [metareplace\(1M\)](#), [metaset\(1M\)](#), [metasync\(1M\)](#), [metattach\(1M\)](#), [mount_ufs\(1M\)](#), [mddb.cf\(4\)](#)

Solaris Volume Manager Administration Guide

Name	volume-request, volume-defaults – Solaris Volume Manager configuration information for top down volume creation with metassist
Synopsis	<pre> /usr/share/lib/xml/dtd/volume-request.dtd /usr/share/lib/xml/dtd/volume-defaults.dtd /etc/defaults/metassist.xml </pre>
Description	<p>A volume request file, XML-based and compliant with the <code>volume-request.dtd</code> Document Type Definition, describes the characteristics of the volumes that <code>metassist</code> should produce.</p> <p>A system administrator would use the volume request file instead of providing options at the command line to give more specific instructions about the characteristics of the volumes to create. A volume request file can request more than one volume, but all requested volumes must reside in the same disk set.</p> <p>If you start <code>metassist</code> by providing a volume-request file as input, <code>metassist</code> can implement the configuration specified in the file, can generate a command file that sets up the configuration for you to inspect or edit, or can generate a volume configuration file for you to inspect or edit.</p> <p>As a system administrator, you would want to create a volume request file if you need to reuse configurations (and do not want to reenter the same command arguments), or if you prefer to use a configuration file to specify volume characteristics.</p> <p>Volume request files must be valid XML that complies with the document type definition in the <code>volume-request.dtd</code> file, located at <code>/usr/share/lib/xml/dtd/volume-request.dtd</code>. You create a volume request file, and provide it as input to <code>metassist</code> to create volumes from the top down.</p>
Defining Volume Request	<p>The top level element <code><volume-request></code> surrounds the volume request data. This element has no attributes. A volume request requires at least one <code><diskset></code> element, which must be the first element after <code><volume-request></code>.</p> <p>Optionally, the <code><volume-request></code> element can include one or more <code><available></code> and <code><unavailable></code> elements to specify which controllers or disks associated with a specific controller can or cannot be used to create the volume.</p> <p>Optionally, the <code><volume-request></code> element can include a <code><hsp></code> element to specify characteristics of a hot spare pool if fault recovery is used.</p> <p>If not specified for a volume with fault-recovery, the first hot spare pool found in the disk set is used. If no hot spare pool exists but one is required, a hot spare pool is created.</p> <p>Optionally, the volume-request can include one or more <code><concat></code>, <code><stripe></code>, <code><mirror></code>, <code><volume></code> elements to specify volumes to create.</p>

- Defining Disk Set** Within the <volume-request> element, a <diskset> element must exist. The <diskset> element, with the name attribute, specifies the name of the disk set to be used. If this disk set does not exist, it is created. This element and the name attribute are required.
- Defining Availability** Within the <volume-request> element and within other elements, you can specify available or unavailable components (disks, or disks on a specific controller path) for use or exclusion from use in a volume or hot spare pool.

The <available> and <unavailable> elements require a name attribute which specifies either a full ctd name, or a partial ctd name that is used with the implied wildcard to complete the expression. For example, specifying c3t2d0 as available would look like:

```
<available name="/dev/dsk/c3t2d0">
```

The <available> element also makes any unnamed components unavailable. Specifying all controllers except c1 unavailable would look like:

```
<available name="c1">
```

Specifying all disks on controller 2 as unavailable would look like:

```
<unavailable name="c2">
```

The <unavailable> element can also be used to further restrict the list of available components. For example, specifying all controllers except c1 unavailable, and making all devices associated with c1t2 unavailable as well would look like this:

```
<available name="c1">  
<unavailable name="c1t2">
```

Components specified as available must be either part of the named disk set used for this volume creation, or must be unused and not in any disk set. If the components are selected for use, but are not in the specified diskset, the `metassist` command automatically adds them to the diskset.

It is unnecessary to specify components that are in other disk sets as unavailable. `metassist` automatically excludes them from consideration. However, unused components or components that are not obviously used (for example, an unmounted slice that is reserved for different uses) must be explicitly specified as unavailable, or the `metassist` command can include them in the configuration.

- Defining Hot Spare Pool** The next element within the <volume-request> element, after the <diskset> and, optionally, <available> and <unavailable> elements, is the <hsp> element. Its sole attribute specifies the name of the hot spare pool:

```
<hsp name="hsp001">
```

The hot spare pool names must start with `hsp` and conclude with a number, thus following the existing Solaris Volume Manager hot spare pool naming requirements.

Within the <hsp> element, you can specify one or more <available> and <unavailable> elements to specify which disks, or disks associated with a specific controller can or cannot be used to create the hot spares within the pool.

Also within the <hsp> element, you can use the <slice> element to specify hot spares to be included in the hot spare pool (see DEFINING SLICE). Depending on the requirements placed on the hot spare pool by other parts of the volume request, additional slices can be added to the hot spare pool.

Defining Slice The <slice> element is used to define slices to include or exclude within other elements. It requires only a name attribute to specify the ctd name of the slice, and the context of the <slice> element determines the function of the element. Sample slice elements might look like:

```
<slice name="c0t1d0s2" />
<slice name="c0t12938567201lkj29561sllkj381d0s2" />
```

Defining Stripe The <stripe> element defines stripes (interlaced RAID 0 volumes) to be used in a volume. It can contain either slice elements (to explicitly determine which slices are used), or appropriate combinations of available and unavailable elements if the specific determination of slices is to be left to the metassist command.

The <stripe> element takes an optional name attribute to specify a name. If the name is not specified, an available name is automatically selected from available Solaris Volume Manager names. If possible, names for related components are related.

The <stripe> element takes an optional size attribute that specifies the size as value and units (for example, 10TB, 5GB). If slices for the <stripe> are explicitly specified, the size attribute is ignored. The <available> and <unavailable> elements can be used to constrain slices for use in a stripe.

The <stripe> elements takes optional mincomp and maxcomp attributes to specify both the minimum and maximum number of components that can be included in it. As with size, if slices for the <stripe> are explicitly specified, the mincomp and maxcomp attributes are ignored.

The <stripe> elements takes an optional interlace attribute as value and units (for example, 16KB, 5BLOCKS, 20KB). If this value is not specified, the Solaris Volume Manager default value is used.

The <stripe> element takes an optional usehsp attribute to specify if a hot spare pool should be associated with this component. This attribute is specified as a boolean value, as usehsp="TRUE". If the component is not a submirror, this attribute is ignored.

Defining Concat The <concat> element defines concats (non-interlaced RAID 0 volumes) to be used in a configuration. It is specified in the same way as a <stripe> element, except that the mincomp, maxcomp, and interlace attributes are not valid.

Defining Mirror The <mirror> element defines mirrors (RAID 1 volumes) to be used in a volume configuration. It can contain combinations of <concat> and <stripe> elements (to explicitly determine which volumes are used as submirrors). Alternatively, it can have a size attribute specified, along with the appropriate combinations of available and unavailable elements to leave the specific determination of components to the metassist command.

The <mirror> element takes an optional name attribute to specify a name. If the name is not specified, an available name is automatically selected.

The <mirror> element takes an optional size attribute that specifies the size as value and units (for example, 10TB, 5GB). If <stripe> and <concat> elements for the mirror are not specified, this attribute is required. Otherwise, it is ignored.

The <mirror> element takes an optional nsubmirrors attribute to define the number of submirrors (1-4) to include. Like the size attribute, this attribute is ignored if the underlying <concat> and <stripe> submirrors are explicitly specified. The <mirror> element takes an optional read attribute to define the mirror read options (ROUNDROBIN, GEOMETRIC, or FIRST) for the mirror. If this attribute is not specified, the Solaris Volume Manager default value is used.

The <mirror> element takes an optional write attribute to define the mirror write options (PARALLEL, SERIAL, or FIRST) for the mirror. If this attribute is not specified, the Solaris Volume Manager default value is used.

The <mirror> element takes an optional usehsp attribute to specify if a hot spare pool should be associated with each submirror. This attribute is specified as a boolean value, as usehsp="TRUE". If the usehsp attribute is specified in the configuration of the <stripe> or <concat> element used as a submirror, it overrides the value of usehsp attributes for the mirror as a whole.

Defining Volume by Quality of Service The <volume> element defines volumes (high-level) by the quality of service they should provide. (The <volume> element offers the same functionality that options on the metassist command line can provide.)

The <volume> element can contain combinations of <available> and <unavailable> elements to determine which components can be included in the configuration.

The <volume> element takes an optional name attribute to specify a name. If the name is not specified, an available name is automatically selected.

The <volume> element takes a required size attribute that specifies the size as value and units (for example, 10TB, 5GB).

The <volume> element takes an optional redundancy attribute to define the number of additional copies of data (1-4) to include. In a worst-case scenario, a volume can suffer failure of $n - 1$ components without data loss, where $\text{redundancy} = n$. With fault recovery options, the volume could withstand up to $n + \text{hsps} - 1$ non-concurrent failures without data loss. Specifying $\text{redundancy} = 0$ results in a RAID 0 volume being created (a stripe, specifically).

The <volume> element takes an optional faultrecovery attribute to determine if additional components should be allocated to recover from component failures in the volume. This is used to determine whether the volume is associated with a hot spare pool. The faultrecovery attribute is a boolean attribute, with a default value of FALSE.

The <volume> element takes an optional datapaths attribute to determine if multiple data paths should be required to access the volume. The datapaths attribute should be set to a numeric value.

Defining Default Values Globally

Global defaults can be set in `/etc/default/metassist.xml`. This volume-defaults file can contain most of the same elements as a volume-request file, but differs structurally from a volume-request file:

- The container element must be <volume-defaults>, not <volume-request>.
- The <volume-defaults> element can contain <available>, <unavailable>, <hsp>, <concat>, <stripe>, <mirror>, or <volume> elements.

Attributes specified by these elements define global default values, unless overridden by the corresponding attributes and elements in a volume-request. None of these elements is a container element.

- The <volume-defaults> element can contain one or more <diskset> elements to provide disk set-specific defaults. The <diskset> element can contain <available>, <unavailable>, <hsp>, <concat>, <stripe>, <mirror>, or <volume> elements.
- Settings specified outside of a <diskset> element apply to all disk sets, but can be overridden within each <diskset> element.

Examples EXAMPLE 1 Creating a Redundant Volume

The following example shows a volume request file used to create a redundant and fault tolerant volume of 1TB.

```
<volume-request>
  <diskset name="sparestorage"/>
  <volume size="1TB" redundancy="2" faultrecovery="TRUE">
    <available name="c2" />
    <available name="c3" />
    <unavailable name="c2t2d0" />
  </volume>
</volume-request>
```

EXAMPLE 2 Creating a Complex Configuration

The following example shows a sample volume-request file that specifies a disk set name, and specifically itemizes characteristics of components to create.

```
<volume-request>

  <!-- Specify the disk set to use -->
  <diskset name="mailspool"/>

  <!-- Generally available devices -->
  <available name="c0"/>

  <!-- Create a 3-way mirror with redundant datapaths and HSPs /
  via QoS -->
  <volume size="10GB" redundancy="3" datapaths="2" /
  faultrecovery="TRUE"/>

  <!-- Create a 1-way mirror with a HSP via QoS -->
  <volume size="10GB" faultrecovery="TRUE"/>

  <!-- Create a stripe via QoS -->
  <volume size="100GB"/>

</volume-request>
```

Boundary Values	Attribute	Minimum	Maximum
	mincomp	1	N/A
	maxcomp	N/A	32
	nsubmirrors	1	4
	passnum	0	9
	datapaths	1	4
	redundancy	0	4

- Files**
- /usr/share/lib/xml/dtd/volume-request.dtd
 - /usr/share/lib/xml/dtd/volume-defaults.dtd
 - /etc/defaults/metassist.xml

See Also [metassist\(1M\)](#), [metaclear\(1M\)](#), [metadb\(1M\)](#), [metadetach\(1M\)](#), [metahs\(1M\)](#), [metainit\(1M\)](#), [metaoffline\(1M\)](#), [metaonline\(1M\)](#), [metaparam\(1M\)](#), [metarecover\(1M\)](#), [metareplace\(1M\)](#), [metaset\(1M\)](#), [metasync\(1M\)](#), [metattach\(1M\)](#), [mount_ufs\(1M\)](#), [mddb.cf\(4\)](#)

Solaris Volume Manager Administration Guide

-
- Name** wanboot.conf – repository for WANboot configuration data
- Synopsis** /etc/netboot/wanboot.conf
- Description** The wanboot.conf file is set up by a system administrator for one or more WANboot clients. The file contains information used to drive the WANboot process. The CGI program that serves up the bootstrap (wanboot) and the boot and root filesystems use information contained in the file to determine file paths, encryption and signing policies, and other characteristics of the operating environment.
- A copy of wanboot.conf is incorporated in the boot filesystem that is transmitted to the client. This is used by the bootstrap (wanboot) to determine SSL authentication policy, and other security conditions.
- You should use the [bootconfchk\(1M\)](#) utility to check the format and content of a wanboot.conf file prior to deployment.
- File Format** Entries in wanboot.conf are written one per line; an entry cannot be continued onto another line. Blank lines are ignored, as is anything following a hash mark character (#), which allows you to insert comments.
- Each non-blank, non-comment line must take the form:
- ```
parameter=value
```
- where *value* is terminated by the end-of-line, a space, or the hash mark character. The value can be quoted if it contains a space or a hash mark, using single or double quotes.
- The parameters currently supported and their meanings are as follows:
- |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>boot_file</i>   | Specifies the path of the bootstrap file relative to the directory from which the web server serves files. This parameter must be given if the bootstrap file (wanboot) is to be served via HTTP, and must be specified with a leading slash (/).                                                                                                                                                                                                                                                                                                                                                                                              |
| <i>root_server</i> | Specifies the location of the CGI program that will serve up the information about the root filesystem that will be transmitted to the client. If present, the value must be a URL in one of the following forms: <pre>http://host:port/some_path/wanboot-cgi</pre> <pre>https://host:port/some_path/wanboot-cgi</pre> <p>where <code>http</code> specifies insecure download of the root filesystem; <code>https</code> specifies secure download of the root filesystem; <i>host</i> is the name of the system which will serve the root filesystem; <i>port</i> is the port through which the web server will serve the root filesystem</p> |

image; *some-path* is the directory which contains the *wanboot-cgi* CGI program which will serve information about the root filesystem. For example:

```
http://webserver:8080/cgi-bin/wanboot-cgi
```

|                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>root_file</i>             | Specifies the path of the root filesystem image relative to the directory from which the web server serves files. This parameter must be given if the root filesystem is to be served by means of HTTP, and must be specified with a leading /.                                                                                                                                                                                                                                                               |
| <i>signature_type</i>        | <p>Specifies the signing algorithm to be used when signing the bootstrap (that is, wanboot), the boot filesystem, and the root filesystem (assuming the last is not being sent using secure HTTP), prior to transmission to the client. If absent, or the value is empty, no signing will be performed. If present, its value must be: sha1.</p> <p>If <i>signature_type</i> is set, the client system being booted must also be setup with a client key for that algorithm.</p>                              |
| <i>encryption_type</i>       | <p>Specifies the encryption algorithm to be used when encrypting the boot filesystem prior to transmission to the client. If absent, or the value is empty, no encryption of the boot filesystem will be performed. If present, its value must be one of: 3des or aes.</p> <p>If <i>encryption_type</i> is set to one of the above algorithms, then the client system being booted must also be setup with a client key for that algorithm and a non-empty <i>encryption_type</i> must also be specified.</p> |
| <i>server_authentication</i> | Specifies whether server authentication should be requested during SSL connection setup. If absent, or the value is empty, server authentication will not be requested. If present, its value must be one of: yes or no.                                                                                                                                                                                                                                                                                      |
| <i>client_authentication</i> | <p>Specifies whether client authentication should be requested during SSL connection setup. If absent, or the value is empty, client authentication will not be requested. If present, its value must be one of: yes or no.</p> <p>If <i>client_authentication</i> is yes, then encryption and signing algorithms must also be specified, the URL scheme in <i>root_server</i> must be https, and <i>server_authentication</i> must also be yes.</p>                                                          |
| <i>resolve_hosts</i>         | Used to specify any host names that might need to be resolved for the client system. Host names appearing in URLs in <i>wanboot.conf</i> and any discovered in certificates associated with the client will                                                                                                                                                                                                                                                                                                   |

automatically be resolved and do not need to be specified here. The value should be a comma-separated list of host names.

A typical use of this parameter would be to name hosts used by the installer that differ from any of those used by the bootstrap.

#### *boot\_logger*

Specifies the URL of a system to which logging messages will be sent. If absent, or the value is empty, then logging will be to the system console only. If present it must specify a URL in one of the following forms:

```
http://host:port/some_path/bootlog-cgi
https://host:port/some_path/bootlog-cgi
```

where the constituent parts are as defined for *root\_server*, above.

Logging can be insecure or secure.

#### *system\_conf*

Specifies the name of a file in the `/etc/netboot` hierarchy that will be incorporated in the boot filesystem named `system.conf` and which is intended for use by the system startup scripts only.

### Examples EXAMPLE 1 Sample File

The following is a sample `wanboot.conf` file:

```
#####
#
Copyright 2003 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
#
#ident "@(#)wanboot.conf 1.12 03/01/30 SMI"
#
#####
wanboot.conf(4): boot configuration file.
#
Please consult wanboot.conf(4) for further information. Note that
this interface is "Committed" as defined by attributes(5).
#
Anything after a '#' is comment. Values may be quoted (e.g. "val").
#
<empty> means there is no value, i.e. null. The absence of any
parameter implies that it takes a default value (<empty> unless
otherwise specified).
#
<url> is of the form http://... or https://...
#####

The path of the bootstrap file (within htdocs) which is served up
```

**EXAMPLE 1** SampleFile (Continued)

```
by wanboot-cgi(bootfile).
#
boot_file=/bootfiles/wanboot # <absolute pathname>

These are used by wanboot-cgi(bootfile|bootfs|rootfs) to determine
whether boot_file or the bootfs is to be sent encrypted/signed, or
root_file is to be sent signed; the client must be setup with the
corresponding encryption/signature key(s) (which cannot be auto-
matically verified).
#
If an encryption_type is specified then a signature_type must also
be specified.
#
encryption_type=3des # 3des | aes | <empty>
signature_type=sha1 # sha1 | <empty>

This is used by wanboot-cgi(bootfs) and WANboot to determine whether
server authentication should be requested during SSL connection
setup.
#
server_authentication=yes # yes | no

This is used by wanboot-cgi(bootfs) and wanboot to determine whether
client authentication should be requested during SSL connection
setup. If client_authentication is "yes", then server_authentication
must also be "yes".
#
client_authentication=yes # yes | no

wanboot-cgi(bootfs) will construct a hosts file which resolves any
hostnames specified in any of the URLs in the wanboot.conf file,
plus those found in certificates, etc. The following parameter
may be used to add additional mappings to the hosts file.
#
resolve_hosts= # <hostname>[,<hostname>*] | <empty>

This is used to specify the URL of wanboot-cgi on the server on which
the root_file exists, and used by wanboot to obtain the root server's
URL; wanboot substitutes root_file for the pathname part of the URL.
If the schema is http://... then the root_file will be signed if there
is a non-empty signature_type. If server_authentication is "yes", the
schema must be https://...; otherwise it must be http://...
#
root_server=https://www.example.com:1234/cgi-bin/wanboot-cgi # <url> \
```

**EXAMPLE 1** SampleFile (Continued)

```

| <empty>

This is used by wanboot-cgi(rootfs) to locate the path of the
rootfs image (within htdocs) on the root_server.
#
root_file=/rootimages/miniroot # <absolute pathname> | <empty>

This is used by wanboot to determine the URL of the boot_logger
(and whether logging traffic should be sent using http or https),
or whether it should simply be sent to the console.
#
boot_logger=http://www.example.com:1234/cgi-bin/bootlog.cgi # <url> \
| <empty>

This is used by the system startup scripts.
#
system_conf=system.conf

```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Committed       |

**See Also** [bootconfchk\(1M\)](#), [attributes\(5\)](#)

**Name** warn.conf – Kerberos warning configuration file

**Synopsis** /etc/krb5/warn.conf

**Description** The warn.conf file contains configuration information specifying how users will be warned by the ktkd\_warnd daemon about ticket expiration. In addition, this file can be used to auto-renew the user's Ticket-Granting Ticket (TGT) instead of warning the user. Credential expiration warnings and auto-renew results are sent, by means of syslog, to auth.notice.

Each Kerberos client host must have a warn.conf file in order for users on that host to get Kerberos warnings from the client. Entries in the warn.conf file must have the following format:

```
principal [renew[:opt1,...optN]] syslog|terminal time
```

or:

```
principal [renew[:opt1,...optN]] mail time [email address]
```

*principal* Specifies the principal name to be warned. The asterisk (\*) wildcard can be used to specify groups of principals.

*renew* Automatically renew the credentials (TGT) until renewable lifetime expires. This is equivalent to the user running `kinit -R`.

The renew options include:

*log-success* Log the result of the renew attempt on success using the specified method (*syslog|terminal|mail*).

*log-failure* Log the result of the renew attempt on failure using the specified method (*syslog|terminal|mail*). Some renew failure conditions are: TGT renewable lifetime has expired, the KDCs are unavailable, or the cred cache file has been removed.

*log* Same as specifying both *log-success* and *log-failure*.

**Note** – If no log options are given, no logging is done.

*syslog* Sends the warnings to the system's syslog. Depending on the /etc/syslog.conf file, syslog entries are written to the /var/adm/messages file and/or displayed on the terminal.

*terminal* Sends the warnings to display on the terminal.

*mail* Sends the warnings as email to the address specified by *email\_address*.

*time* Specifies how much time before the TGT expires when a warning should be sent. The default time value is seconds, but you can specify h (hours) and m (minutes) after the number to specify other time values.

*email\_address* Specifies the email address at which to send the warnings. This field must be specified only with the `mail` field.

**Examples** EXAMPLE 1 Specifying Warnings

The following `warn.conf` entry

```
* syslog 5m
```

specifies that warnings will be sent to the syslog five minutes before the expiration of the TGT for all principals. The form of the message is:

```
jdb@ACME.COM: your kerberos credentials expire in 5 minutes
```

EXAMPLE 2 Specifying Renewal

The following `warn.conf` entry:

```
* renew:log terminal 30m
```

...specifies that renew results will be sent to the user's terminal 30 minutes before the expiration of the TGT for all principals. The form of the message (on renew success) is:

```
myname@ACME.COM: your kerberos credentials have been renewed
```

**Files** `/usr/lib/krb5/kttk_warnd` Kerberos warning daemon

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Committed       |

**See Also** [kinit\(1\)](#), [kdestroy\(1\)](#), [kttk\\_warnd\(1M\)](#), [syslog.conf\(4\)](#), [utmpx\(4\)](#), [attributes\(5\)](#), [kerberos\(5\)](#), [pam\\_krb5\(5\)](#)

**Notes** The auto-renew of the TGT is attempted only if the user is logged-in, as determined by examining [utmpx\(4\)](#).

**Name** ypfiles – Network Information Service Version 2, formerly known as YP

**Description** The NIS network information service uses a distributed, replicated database of dbm files, in ASCII form, that are contained in the `/var/yp` directory hierarchy on each NIS server.

A dbm database served by the NIS server is called a NIS *map*. An NIS *domain* is a subdirectory of `/var/yp` that contains a set of NIS maps on each NIS server.

Standard nicknames are defined in the file `/var/yp/nicknames`. These names can be used in place of the full map name in the `ypmatch` and `ypcat` commands. Use the command `ypwhich -x` to display the current set of nicknames. Use the command `ypwhich -m` to display all the available maps. Each line of the nickname file contains two fields separated by white space. The first field is the nickname, and the second field is the name of the map that it expands to. The nickname cannot contain a dot (`.`).

**SMF Configuration** The configuration for NIS is managed by the following SMF services:

```
svc:/network/nis/domain
svc:/network/nis/client
svc:/network/nis/server
svc:/network/nis/xfr
svc:/network/nis/passwd
svc:/network/nis/update
```

```
svc:/network/nis/domain
```

Represents the shared configuration used by all NIS components. The properties managed by this service are contained in the `config` property group. Those properties are:

```
config/domainname
```

```
NIS domainname
```

This is a single valued property containing the NIS domainname (`/etc/defaultdomain`)

```
config/ypservers
```

```
ypservers Hosts/IP addresses
```

This is a multi-valued property. Each value contains a server to which the NIS client is allowed to bind.

```
config/securenets
```

```
securenets network data
```

This is a multi-valued property. Each value defines the hosts and networks that are granted access to information in the served domain.

**NIS to LDAP (N2L)** If the `/var/yp/NISLDAPmapping` configuration file is present, the NIS server operates in NIS to LDAP (N2L) mode. In this mode, NIS maps are stored in a new set of DBM files, prepended by the `LDAP_` prefix, at `/var/yp/domainname`. These files are used as a cache backed by information from an LDAP server. Additional DBM files are created in the same directory to hold the cache's TTL values.

N2L mode enables NIS clients to be supported in an LDAP environment.



In N2L mode, the old style DBM files, NIS source files, and the [ypmake\(1M\)](#) utility have to role. They are retained to enable easy conversion back to the traditional mode, if required.

Converting from N2L to  
Traditional NIS

When NIS is operating in N2L mode, it uses a new set of NIS maps with an LDAP\_ prefix, based on the contents of the LDAP DIT. The NIS source files are unused and become out of date. If you wish to convert back to the traditional NIS mode, the N2L configuration file should be deleted. The system then returns to using the standard map files. Optionally, the N2L mode map files, `/var/yp/*/LDAP_*` can also be deleted.

If you want to run the system in traditional mode with information based on the DIT, then the NIS source files must be regenerated based on the N2L maps. To regenerate the NIS source files based on the N2L maps, run [ypmap2src\(1M\)](#).

|              |                                                 |                                                                                                                                                                                |
|--------------|-------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Files</b> | <code>/var/yp</code>                            | Directory containing NIS configuration files.                                                                                                                                  |
|              | <code>/var/yp/binding</code>                    | Stores the information required to bind the NIS client to the NIS server.                                                                                                      |
|              | <code>/var/yp/binding/ypdomain/ypservers</code> | Contains the servers to which the NIS client is allowed to bind.                                                                                                               |
|              | <code>/var/yp/Makefile</code>                   | Builds the NIS ndbm databases.                                                                                                                                                 |
|              | <code>/var/yp/nicknames</code>                  | Nicknames file.                                                                                                                                                                |
|              | <code>/var/yp/securenets</code>                 | Defines the hosts and networks that are granted access to information in the served domain. This file is read at startup time by <code>ypserv</code> and <code>ypxfrd</code> . |
|              | <code>/var/yp/ypdomain</code>                   | Directory containing ndbm databases.                                                                                                                                           |
|              | <code>/var/yp/NISLDAPmapping</code>             | NIS to LDAP configuration file                                                                                                                                                 |
|              | <code>/var/yp/*/LDAP_*</code>                   | NIS to LDAP mode map files                                                                                                                                                     |

**See Also** [makedbm\(1M\)](#), [ypbind\(1M\)](#), [ypinit\(1M\)](#), [ypmake\(1M\)](#), [ypmap2src\(1M\)](#), [ypserv\(1M\)](#), [ypxfrd\(1M\)](#), [ndbm\(3C\)](#), [ypclnt\(3NSL\)](#), [ldap\(5\)](#)

**Name** yppasswdd – configuration file for rpc.yppasswdd (NIS password daemon)

**Synopsis** /etc/default/yppasswdd

**Description** The yppasswdd file contains a parameter that modifies the behavior of the [rpc.yppasswdd\(1M\)](#) daemon.

The yppasswdd file contains a single parameter:

```
#check_restricted_shell_name=1
```

By default in the current release, this line in yppasswdd is commented out. If you uncomment the line, when a user attempts to change his default shell using 'passwd -r nis -e' (see [passwd\(1\)](#)), the rpc.yppasswdd daemon checks whether the name of the user's current shell begins with an 'r'. rpc.yppasswdd considers any shell whose name begins with an 'r' (for example, rcs) to be a restricted shell. If a user's shell does begin with 'r', his attempt to change from the default shell will fail.

If the line in the yppasswdd file is commented out (the default), the rpc.yppasswdd daemon does not perform the restricted shell check.

The yppasswdd file is editable only by root or a member of the sys group.

**Files** /etc/default/yppasswdd configuration file for rpc.yppasswdd daemon

**See Also** [rpc.yppasswdd\(1M\)](#)

---

|                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                 |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>                 | ypserv – configuration file for NIS to LDAP transition daemons                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                 |
| <b>Synopsis</b>             | /etc/default/ypserv                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                                                                                                                                                                 |
| <b>Description</b>          | <p>The ypserv file specifies configuration information for the <a href="#">ypserv(1M)</a> daemon. Configuration information can come from LDAP or be specified in the ypserv file.</p> <p>You can create a simple ypserv file by running <a href="#">inityp2l(1M)</a>. The ypserv file can then be customized as required.</p> <p>A related NISLDAPmapping file contains mapping information that converts NIS entries into LDAP entries. See the <a href="#">NISLDAPmapping(4)</a> man page for an overview of the setup that is needed to map NIS data to or from LDAP.</p> |                                                                                                                                                                 |
| <b>Extended Description</b> | <p>The <a href="#">ypserv(1M)</a> server recognizes the attributes that follow. Values specified for these attributes in the ypserv file, including any empty values, override values that are obtained from LDAP. However, the nisLDAPconfig* values are read from the ypserv file only</p>                                                                                                                                                                                                                                                                                  |                                                                                                                                                                 |
| <b>Attributes</b>           | The following are attributes that are used for initial configuration.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                 |
|                             | nisLDAPconfigDN                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | The DN for configuration information. If nisLDAPconfigDN is empty, all other nisLDAPConfig* values are ignored.                                                 |
|                             | nisLDAPconfigPreferredServerList                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | The list of servers to use for the configuration phase. There is no default value. The following is an example of a value for nisLDAPconfigPreferredServerList: |
|                             | nisLDAPconfigPreferredServerList=127.0.0.1:389                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                 |
|                             | nisLDAPconfigAuthenticationMethod                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | The authentication method used to obtain the configuration information. The recognized values for nisLDAPconfigAuthenticationMethod are:                        |
|                             | none                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | No authentication attempted                                                                                                                                     |
|                             | simple                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Password of proxy user sent in the clear to the LDAP server                                                                                                     |
|                             | sasl/cram-md5                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Use SASL/CRAM-MD5 authentication. This authentication method may not be supported by                                                                            |

|                                   |                                                                                                                                                                                                                                                                                                                       |      |                                                                   |     |                                                                    |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-------------------------------------------------------------------|-----|--------------------------------------------------------------------|
|                                   | all LDAP servers. A password must be supplied.                                                                                                                                                                                                                                                                        |      |                                                                   |     |                                                                    |
|                                   | sasl/digest-md5 Use SASL/DIGEST-MD5 authentication. The SASL/CRAM-MD5 authentication method may not be supported by all LDAP servers. A password must be supplied.                                                                                                                                                    |      |                                                                   |     |                                                                    |
|                                   | <p>nisLDAPconfigAuthenticationMethod has no default value. The following is an example of a value for nisLDAPconfigAuthenticationMethod:</p> <pre>nisLDAPconfigAuthenticationMethod=simple</pre>                                                                                                                      |      |                                                                   |     |                                                                    |
| nisLDAPconfigTLS                  | <p>The transport layer security used for the connection to the server. The recognized values are:</p> <table><tr><td>none</td><td>No encryption of transport layer data. The default value is none.</td></tr><tr><td>ssl</td><td>SSL encryption of transport layer data. A certificate is required.</td></tr></table> | none | No encryption of transport layer data. The default value is none. | ssl | SSL encryption of transport layer data. A certificate is required. |
| none                              | No encryption of transport layer data. The default value is none.                                                                                                                                                                                                                                                     |      |                                                                   |     |                                                                    |
| ssl                               | SSL encryption of transport layer data. A certificate is required.                                                                                                                                                                                                                                                    |      |                                                                   |     |                                                                    |
|                                   | <p>Export and import control restrictions might limit the availability of transport layer security.</p>                                                                                                                                                                                                               |      |                                                                   |     |                                                                    |
| nisLDAPconfigTLSCertificateDBPath | The name of the directory that contains the certificate database. The default path is /var/yp.                                                                                                                                                                                                                        |      |                                                                   |     |                                                                    |
| nisLDAPconfigProxyUser            | <p>The proxy user used to obtain configuration information. nisLDAPconfigProxyUser has no default value. If the value ends with a comma, the value of the nisLDAPconfigDN attribute is appended. For example:</p> <pre>nisLDAPconfigProxyUser=cn=nisAdmin,ou=People,</pre>                                            |      |                                                                   |     |                                                                    |
| nisLDAPconfigProxyPassword        | <p>The password that should be supplied to LDAP for the proxy user when the authentication method requires one. To avoid exposing this password publicly on the machine, the password should only appear in the configuration file, and the file should have an appropriate owner, group,</p>                         |      |                                                                   |     |                                                                    |

and file mode. `nisLDAPconfigProxyPassword` has no default value.

The following are attributes used for data retrieval. The object class name used for these attributes is `nisLDAPconfig`.

|                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>preferredServerList</code>         | The list of servers to use to read or to write mapped NIS data from or to LDAP. <code>preferredServerList</code> has no default value. For example:<br><br><code>preferredServerList=127.0.0.1:389</code>                                                                                                                                                                                                                                                                                                              |
| <code>authenticationMethod</code>        | The authentication method to use to read or to write mapped NIS data from or to LDAP. For recognized values, see the <code>LDAPconfigAuthenticationMethod</code> attribute. <code>authenticationMethod</code> has no default value. For example:<br><br><code>authenticationMethod=simple</code>                                                                                                                                                                                                                       |
| <code>nisLDAPTLS</code>                  | The transport layer security to use to read or to write NIS data from or to LDAP. For recognized values, see the <code>nisLDAPconfigTLS</code> attribute. The default value is none. Export and import control restrictions might limit the availability of transport layer security.                                                                                                                                                                                                                                  |
| <code>nisLDAPTLSCertificateDBPath</code> | The name of the directory that contains the certificate DB. For recognized and default values for <code>nisLDAPTLSCertificateDBPath</code> , see the <code>nisLDAPconfigTLSCertificateDBPath</code> attribute.                                                                                                                                                                                                                                                                                                         |
| <code>nisLDAPproxyUser</code>            | Proxy user used by <code>ypserv(1M)</code> , <code>ypxfrd(1M)</code> and <code>yppasswdd(1M)</code> to read or to write from or to LDAP. Assumed to have the appropriate permission to read and modify LDAP data. There is no default value. If the value ends in a comma, the value of the context for the current domain, as defined by a <code>nisLDAPdomainContext</code> attribute, is appended. See <a href="#">NISLDAPmapping(4)</a> . For example:<br><br><code>nisLDAPproxyUser=cn=nisAdmin,ou=People,</code> |
| <code>nisLDAPproxyPassword</code>        | The password that should be supplied to LDAP for the proxy user when the authentication method so requires. To avoid exposing this password publicly on the machine, the password should only appear in the                                                                                                                                                                                                                                                                                                            |

|                                                                                                                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                                                             | configuration file, and the file must have an appropriate owner, group, and file mode. <code>nisLDAPproxyPassword</code> has no default value.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>nisLDAPsearchTimeout</code>                                                                                                                                                                           | Establishes the timeout for the LDAP search operation. The default value for <code>nisLDAPsearchTimeout</code> is 180 seconds.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>nisLDAPbindTimeout</code><br><code>nisLDAPmodifyTimeout</code><br><code>nisLDAPaddTimeout</code><br><code>nisLDAPdeleteTimeout</code>                                                                 | Establish timeouts for LDAP bind, modify, add, and delete operations, respectively. The default value is 15 seconds for each attribute. Decimal values are allowed.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>nisLDAPsearchTimeLimit</code>                                                                                                                                                                         | Establish a value for the <code>LDAP_OPT_TIMELIMIT</code> option, which suggests a time limit for the search operation on the LDAP server. The server may impose its own constraints on possible values. See your LDAP server documentation. The default is the <code>nisLDAPsearchTimeout</code> value. Only integer values are allowed.<br><br>Since the <code>nisLDAPsearchTimeout</code> limits the amount of time the client <code>ypserv</code> will wait for completion of a search operation, do not set the value of <code>nisLDAPsearchTimeLimit</code> larger than the value of <code>nisLDAPsearchTimeout</code> . |
| <code>nisLDAPsearchSizeLimit</code>                                                                                                                                                                         | Establish a value for the <code>LDAP_OPT_SIZELIMIT</code> option, which suggests a size limit, in bytes, for the search results on the LDAP server. The server may impose its own constraints on possible values. See your LDAP server documentation. The default value for <code>nisLDAPsearchSizeLimit</code> is zero, which means the size limit is unlimited. Only integer values are allowed.                                                                                                                                                                                                                             |
| <code>nisLDAPfollowReferral</code>                                                                                                                                                                          | Determines if the <code>ypserv</code> should follow referrals or not. Recognized values for <code>nisLDAPfollowReferral</code> are <code>yes</code> and <code>no</code> . The default value for <code>nisLDAPfollowReferral</code> is <code>no</code> .                                                                                                                                                                                                                                                                                                                                                                        |
| <p>The following attributes specify the action to be taken when some event occurs. The values are all of the form <code>event=action</code>. The default action is the first one listed for each event.</p> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>nisLDAPretrieveErrorAction</code>                                                                                                                                                                     | If an error occurs while trying to retrieve an entry from LDAP, one of the following actions can be selected:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

|                                           |                         |                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------------------------|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                           | <code>use_cached</code> | <p>Retry the retrieval the number of time specified by <code>nisLDAPretrieveErrorAttempts</code>, with the <code>nisLDAPretrieveErrorTimeout</code> value controlling the wait between each attempt.</p> <p>If all attempts fail, then a warning is logged and the value currently in the cache is returned to the client.</p>                     |
|                                           | <code>fail</code>       | <p>Proceed as for <code>use_cached</code>, but if all attempts fail, a <code>YPERR_YPERR</code> error is returned to the client.</p>                                                                                                                                                                                                               |
| <code>nisLDAPretrieveErrorAttempts</code> |                         | <p>The number of times a failed retrieval should be retried. The default value for <code>nisLDAPretrieveErrorAttempts</code> is unlimited. While retries are made the <code>ypserv</code> daemon will be prevented from servicing further requests. <code>nisLDAPretrieveErrorAttempts</code> values other than 1 should be used with caution.</p> |
| <code>nisLDAPretrieveErrorTimeout</code>  |                         | <p>The timeout in seconds between each new attempt to retrieve LDAP data. The default value for <code>nisLDAPretrieveErrorTimeout</code> is 15 seconds.</p>                                                                                                                                                                                        |
| <code>nisLDAPstoreErrorAction</code>      |                         | <p>An error occurred while trying to store data to the LDAP repository.</p>                                                                                                                                                                                                                                                                        |
|                                           | <code>retry</code>      | <p>Retry operation <code>nisLDAPstoreErrorAttempts</code> times with <code>nisLDAPstoreErrorTimeout</code> seconds between each attempt. While retries are made, the NIS daemon will be prevented from servicing further requests. Use with caution.</p>                                                                                           |
|                                           | <code>fail</code>       | <p>Return <code>YPERR_YPERR</code> error to the client.</p>                                                                                                                                                                                                                                                                                        |
| <code>nisLDAPstoreErrorAttempts</code>    |                         | <p>The number of times a failed attempt to store should be retried. The default value for <code>nisLDAPstoreErrorAttempts</code> is unlimited. The value for <code>nisLDAPstoreErrorAttempts</code> is ignored unless <code>nisLDAPstoreErrorAction=retry</code>.</p>                                                                              |
| <code>nisLDAPstoreErrortimeout</code>     |                         | <p>The timeout, in seconds, between each new attempt to store LDAP data. The default value for</p>                                                                                                                                                                                                                                                 |

nisLDAPstoreErrortimeout is 15 seconds. The nisLDAPstoreErrortimeout value is ignored unless nisLDAPstoreErrorAction=retry.

Storing Configuration Attributes in LDAP Most attributes described on this man page, as well as those described on [NISLDAPmapping\(4\)](#), can be stored in LDAP. In order to do so, you will need to add the following definitions to your LDAP server, which are described here in LDIF format suitable for use by [ldapadd\(1\)](#). The attribute and objectclass OIDs are examples only.

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.4.1.11.1.3.1.1.2 NAME 'preferredServerList' \
DESC 'Preferred LDAP server host addresses used by DUA' \
EQUALITY caseIgnoreMatch \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.11.1.3.1.1.6 NAME 'authenticationMethod' \
DESC 'Authentication method used to contact the DSA' \
EQUALITY caseIgnoreMatch \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE)
```

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.0 \
NAME 'nisLDAPTLS' \
DESC 'Transport Layer Security' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.1 \
NAME 'nisLDAPTLSCertificateDBPath' \
DESC 'Certificate file' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.2 \
NAME 'nisLDAPproxyUser' \
DESC 'Proxy user for data store/retrieval' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.3 \
NAME 'nisLDAPproxyPassword' \
DESC 'Password/key/shared secret for proxy user' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.6 \
NAME 'nisLDAPretrieveErrorAction' \
DESC 'Action following an LDAP search error' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.7 \
NAME 'nisLDAPretrieveErrorAttempts' \
DESC 'Number of times to retry an LDAP search' \
```



```

 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.8 \
 NAME 'nisLDAPretrieveErrorTimeout' \
 DESC 'Timeout between each search attempt' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.9 \
 NAME 'nisLDAPstoreErrorAction' \
 DESC 'Action following an LDAP store error' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.10 \
 NAME 'nisLDAPstoreErrorAttempts' \
 DESC 'Number of times to retry an LDAP store' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.11 \
 NAME 'nisLDAPstoreErrorTimeout' \
 DESC 'Timeout between each store attempt' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.12 \
 NAME 'nisLDAPdomainContext' \
 DESC 'Context for a single domain' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.13 \
 NAME 'nisLDAPyppasswddDomains' \
 DESC 'List of domains for which password changes are made' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.14 \
 NAME 'nisLDAPdatabaseIdMapping' \
 DESC 'Defines a database id for a NIS object' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.15 \
 NAME 'nisLDAPentryTtl' \
 DESC 'TTL for cached objects derived from LDAP' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.16 \
 NAME 'nisLDAPobjectDN' \
 DESC 'Location in LDAP tree where NIS data is stored' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.17) \
 NAME 'nisLDAPnameFields' \
 DESC 'Rules for breaking NIS entries into fields' \ \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)
attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.18) \
 NAME 'nisLDAPsplitFields' \
 DESC 'Rules for breaking fields into sub fields' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.19 \

```

```

 NAME 'nisLDAPattributeFromField' \
 DESC 'Rules for mapping fields to LDAP attributes' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.20 \
 NAME 'nisLDAPfieldFromAttribute' \
 DESC 'Rules for mapping fields to LDAP attributes' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.21 \
 NAME 'nisLDAPrepeatedFieldSeparators' \
 DESC 'Rules for mapping fields to LDAP attributes' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.22 \
 NAME 'nisLDAPcommentChar' \
 DESC 'Rules for mapping fields to LDAP attributes' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

attributetypes: (1.3.6.1.4.1.42.2.27.5.42.43.1.23 \
 NAME 'nisLDAPmapFlags' \
 DESC 'Rules for mapping fields to LDAP attributes' \
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: (1.3.6.1.4.1.42.2.27.5.42.43.1.0 NAME 'nisLDAPconfig' \
 DESC 'NIS/LDAP mapping configuration' \
 SUP top STRUCTURAL \
 MAY (cn $ preferredServerList $
 authenticationMethod $ nisLDAPTLS $
 nisLDAPTSLSCertificateDBPath $
 nisLDAPproxyUser $ nisLDAPproxyPassword $
 nisLDAPretrieveErrorAction $
 nisLDAPretrieveErrorAttempts $
 nisLDAPretrieveErrorTimeout $
 nisLDAPstoreErrorAction $
 nisLDAPstoreErrorAttempts $
 nisLDAPstoreErrorTimeout $
 nisLDAPdomainContext $
 nisLDAPpyppasswddDomains $
 nisLDAPdatabaseIdMapping $
 nisLDAPentryTtl $
 nisLDAPobjectDN $
 nisLDAPnameFields $
 nisLDAPsplitFields $

```

```

nisLDAPattributeFromField $
nisLDAPfieldFromAttribute $
nisLDAPrepeatedFieldSeparators $
nisLDAPcommentChar $
nisLDAPmapFlags))

```

Create a file containing the following LDIF data. Substitute your actual `nisLDAPconfigDN` for `configDN`:

```

dn: configDN
objectClass: top
objectClass: nisLDAPconfig

```

Use this file as input to the `ldapadd(1)` command in order to create the NIS to LDAP configuration entry. Initially, the entry is empty. You can use the `ldapmodify(1)` command to add configuration attributes.

**Examples** EXAMPLE 1 Creating a NIS to LDAP Configuration Entry

To set the server list to port 389 on 127.0.0.1, create the following file and use it as input to `ldapmodify(1)`:

```

dn: configDN
preferredServerList: 127.0.0.1:389

```

**Attributes** See [attributes\(5\)](#) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE     |
|---------------------|---------------------|
| Availability        | service/network/nis |
| Interface Stability | Obsolete            |

**See Also** [ldapadd\(1\)](#), [ldapmodify\(1\)](#), [inittyp2l\(1M\)](#), [ypasswdd\(1M\)](#), [ypserv\(1M\)](#), [ypxfrd\(1M\)](#), [attributes\(5\)](#)

*Oracle Solaris Administration: Naming and Directory Services*

**Name** zoneinfo – timezone information

**Description** For notes regarding the zoneinfo timezones, see `/usr/share/lib/zoneinfo/src/README`.