Solaris ISP Server 2.0 Reference Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
U.S.A.

Part No: 805-7696–10
February 1999

Copyright 1999 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, California 94303-4900 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, SunDocs, HotJava, Java, Java Development Kit, JDK, SunScreen SKIP, Sun WebServer, Ultra, Solaris ISP Server, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Netscape and Netscape Navigator are trademarks of Netscape Communications Corporation.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227–14(g)(2)(6/87) and FAR 52.227–19(6/87), or DFAR 252.227–7015(b)(6/95) and DFAR 227.7202–3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 1999 Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, Californie 94303-4900 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, SunDocs, HotJava, Java, Java Development Kit, JDK, SunScreen SKIP, Sun WebServer, Ultra, Solaris ISP Server, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.Netscape et Netscape Navigator sont marques de Netscape Communications Corporation.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Contents

Part I Solaris ISP Server Core

ispIntro.1m(1M) 3

`hcjump`(1M) 5

`hclfmd`(1M) 7

`hcstartup`(1M) 9

ispldap(1M) 12

isprshp(1M) 15

`mchelp`(1M) 16

`mcreg`(1M) 18

`mcunreg`(1M) 27

sisploa d(1M) 29

uninstall-sisp1.0(1M) 33

ispIntro(3X) 36

ispGetLdapInfo(3X) 38

ispGetLdapServers(3X) 41

ispGetTopDn(3X) 43

IspLdapService(3X) 45

ISPMC_aar(3X) 49

ispIntro(4) 53

`hclfmd.conf`(4) 54

sisploa d.mapping(4) 56

Part II Sun Internet FTP Server

1. FTP Command-Line Procedures 73

1.1 FTP Configuration 73

1.1.1 Procedure 74

1.2 Subscriber Authentication 75

1.2.1 LDAP 75

1.3 Maintenance 77

1.3.1 Start `ftpd` 77

1.3.2 Stop `ftpd` 77

1.3.3 Show Users by Class 78

1.3.4 Delete an FTP Site 78

ftpintro.1m(1M) 81

ftpaddhost(1M) 82

ftpconfig(1M) 84

ftpcount(1M) 86

ftpshut(1M) 87

in.ftpd(1M) 89

ftpintro.4(4) 98

ftpaccess(4) 99

ftpconversions(4) 111

ftphosts(4) 113

ftpservers(4) 114

xferlog(4) 116

Part III Sun Internet Administrator

mcIntro(1M) 121

`mcadd`(1M) 124

mcaddadm(1M) 125

mcadmpwd(1M) 127

mcdsclean(1M) 129

mcdsinit(1M) 131

mchostls(1M) 133

mcrm(1M) 135

mcrmadm(1M) 137

Part IV Sun Internet News Server

2. News Command Line Procedures 141

2.1 Procedures 141

2.1.1 Start/Stop Reader/Feeder Servers 141

2.1.2 Newsgroup Tasks 142

newsIntro.1m(1M) 145

archive(1m) 149

batcher(1m) 151

buffchan(1m) 154

crosspost(1m) 157

ctlinnd(1m) 159

cvtbatch(1m) 166

expire(1m) 167

expireover(1m) 170

expirerm(1m) 172

fastrm(1m) 173

filechan(1m) 175

inncheck(1m) 177

innd(1m) 180

innstat(1m) 181

innwatch(1m) 182

innxbatch(1m) 183
innxmit(1m) 185
isppammod(1m) 188
makeactive(1m) 190
makehistory(1m) 192
news.daily(1m) 195
scanlogs(1m) 198
makeactive(1m) 199
newsrequeue(1m) 200
nntpget(1m) 202
nntpsend(1m) 203
overchan(1m) 206
prunehistory(1m) 207
rnews(1m) 209
scanlogs(1m) 211
snsd(1m) 212
snsnews(1m) 213
tally.control(1m) 214
tally.unwanted(1m) 215
writelog(1m) 216
newsIntro.4(4) 218
active(4) 220
control.ctl(4) 222
distrib.pats(4) 224
expire.ctl(4) 225
history(4) 228
hosts.nntp(4) 230
inn.conf(4) 232

innwatch.ctl(4) 235

moderators(4) 239

newsadmconfig(4) 240

newsfeeds(4) 242

newslog(4) 251

nnrp.access(4) 254

nntpsend.ctl(4) 256

overview.fmt(4) 257

passwd.nntp(4) 258

sns.conf(4) 259

Part V Sun WebServer

htIntro(1M) 263

/usr/bin/htaccess (1M) 267

/usr/bin/htcontent (1M) 273

/usr/bin/hthost(1M) 280

/usr/bin/htmap(1M) 284

/usr/bin/htpasswd(1M) 288

/usr/bin/htrealm(1M) 291

/usr/bin/htserver(1M) 297

/usr/bin/htservlet(1M) 302

/usr/lib/httpd(1M) 313

htIntro(4) 315

access.conf(4) 319

content.conf(4) 324

groups(4) 329

httpd-instances.conf(4) 331

httpd.conf(4) 334

httpd.event.logs(4) 354

httpd.request.logs(4) 357

httpd.servlet.logs and httpd.cgi.logs(4) 362

httpd.site.conf(4) 365

map.conf(4) 382

realms.conf(4) 386

servlets.properties(4) 392

users(4) 394

Index 397

PART I Solaris ISP Server Core

Solaris ISP Server™ 2.0 core man pages.

CHAPTER

man Pages(1m): Maintenance Commands

NAME

ispIntro.1m – Introduction to the command-line utilities for Solaris ISP Server

DESCRIPTION

The man pages offer detailed instruction and examples on the options and subcommands for each utility. The command-line utilities are available to start and run the host configuration tool that installs Solaris ISP Server components and configures the system; to register and unregister services on this host with the Sun Internet Administrator console; and to bulk load subscriber entries from an existing data source into the Solaris ISP Server directory service.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWispm
Interface Stability	Evolving

FILES

`/opt/SUNWisp/sbin/hcjump`

`/opt/SUNWisp/lib/hclfmd`

`/opt/SUNWisp/sbin/hcstartup`

`/opt/SUNWisp/sbin/ispladp`

`/opt/SUNWisp/sbin/isprshp`

`/opt/SUNWisp/sbin/mchelp`

`/opt/SUNWisp/sbin/mcreg`

`/opt/SUNWisp/sbin/mcunreg`

`/opt/SUNWisp/ldap/sunds/sbin/sisploa`d

`/opt/SUNWisp/sbin/uninstall-sisp1.0`

SEE ALSO

hclmfd.conf(4), **sisploa**d**.mapping**(4)

NOTES

hcjump(1m) performs a non-interactive Solaris ISP Server software installation and configuration. `hcjump` is designed to be called from a JumpStart custom configuration finish script.

hclfmd(1m) maintains log files written by `syslogd` and `auditd`, and periodically cycles and archives log files. `hclfmd` detects intrusion attempts and notifies.

hcstartup(1m)	starts the host configuration tool and Web-based user interface for installation and configuration of Solaris ISP Server software. `hcstartup` performs setup and initialization, and from a browser presents the host configuration user interface.
ispldap(1m)	creates entries in configuration files, for access by the ISP IDIA library functions (which return information required for accessing directory services). Also creates required directory services entries.
isprshp(1m)	sets the port on which the Solaris ISP Server remote command execution daemon listens.
mchelp(1m)	displays the release version of Sun™ Internet Administrator™ installed on the system and lists all utilities associated with it.
mcreg(1m)	registers a software component making it available for management through the Sun Internet Administrator console. Overwrites any previous registration of the same component.
mcunreg(1m)	unregisters a software component GUI, making it unavailable to the Sun Internet Administrator console.
sispload**(1m)**	converts a file of user entries into an ldif file for loading into the Solaris ISP Server directory service.
uninstall-sisp1.0(1m)	performs a non-interactive uninstall of Solaris™ for ISPs™ 1.0 components and saves the old data.

NAME

`hcjump` – Perform a non-interactive Solaris ISP Server™ installation and configuration session.

SYNOPSIS

hcjump *directory*

DESCRIPTION

Performs setup and initialization for installation and configuration. `hcjump` is designed to be called from a JumpStart custom configuration finish script. You must have root access to run `hcjump`. It can also be used to perform a replicated installation and upgrade.

OPERANDS

directory The path to the root directory of the installation media for Solaris ISP Server.

EXTENDED DESCRIPTION

`hcjump` does the following:

- Copies the host configuration archive files into `/tmp`.
- Copies information from the installation media into `/var/opt/SUNWisp/hc/media`.
- Removes any existing configuration information in `/var/opt/SUNWisp/hc/media` to copy information from the installation media.
- Copies files from the exported scenario directory into `/var/opt/SUNWisp/hc/scenario`.
- Implements the scenario, installing and removing software and reconfiguring as specified.
- Cleans up `/tmp`.

When you save an exported scenario from the host configuration GUI, a copy of `hcjump` is saved in its root directory. Invoke this version of `hcjump` from your finish script.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

FILES

`/var/opt/SUNWisp/hc` Both `hcjump` and `hcstartup` store configuration information in this directory. When `hcjump` is invoked, all data in the host configuration directory is removed and replaced with new data.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving

SEE ALSO

ispIntro(1M) **hclfmd**(1m), **hcstartup**(1m), **uninstall-sisp1.0**(1m), **hclmfd.conf**(4)

NOTES

When `hcjump` runs, it removes the information on the current state of the system in `/var/opt/SUNWisp/hc` saved by the host configuration software (`hcstartup`). For this reason, invoke `hcjump` before invoking `hcstartup` and not vice versa.

NAME

`hclfmd` – Monitor syslog files for suspicious intrusion attempts; periodically cycle and archive log files.

SYNOPSIS

hclfmd

DESCRIPTION

The host configuration log file management daemon is a resident daemon that is started when the system boots. It perfoms three basic functions:

- `hclfmd` maintains log files written by `syslogd`:
  - Deletes weekly archives older than one month.
  - Creates a weekly archive by compiling and compressing a week's worth of daily logs and deleting the daily logs. The weekly log is named: `name.YYYYMMDDHHMM-YYYYMMDDHHMM.tar.z`.
  - Cycles the daily logs by renaming the existing logs and sending a hangup (SIGHUP) signal.

  `hclfmd` finds these files by reading `/etc/syslog.conf`.
- `hclfmd` maintains log files written by `auditd`:
  - Deletes weekly archives older than one month.
  - Creates a weekly archive by compiling and compressing a week's worth of daily logs and deleting the daily logs. The weekly log is named: `name.YYYYMMDDHHMM-YYYYMMDDHHMM.tar.z`.
  - Cycles the daily logs by running `audit –n`.

  `hclfmd` finds these files by reading `/etc/security/audit_control`.
- By default, every minute, `hclfmd` examines its log files for intrusion attempts. The interval at which `hclfmd` examines its log files and the files checked for intrusion attempts are defined in `hclfmd.conf`. If `hclfmd` discovers any intrusion attempts, it runs the notification command listed in the configuration file.

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

FILES

`/etc/opt/SUNWisp/hc/hclfmd.conf`	When `hclfmd` finds an intrusion attempt in a log file, it reads this file to discover a notification mechanism.

`/etc/security/audit_control`	`hclfmd` reads this file to discover audit logs to monitor.
`/etc/syslog.conf`	`hclfmd` reads this file to discover syslog logs to monitor.
`/var/spool/cron/crontabls/root`	`hclfmd` removes the `cron` entry for `/usr/lib/newsyslog` because the log file management functionality is an effective replacement for it.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving

SEE ALSO

`ispIntro`(1M) **`hcjump`**(1m), **`hcstartup`**(1m), **`hclmfd.conf`**(4), **`uninstall-sisp1.0`**(1m), **`syslogd`**(4), **`syslog.conf`**(4)

NAME

`hcstartup` – Start the host configuration tool and Web-based user interface for installation and configuration of Solaris ISP Server™ software.

SYNOPSIS

hcstartup

DESCRIPTION

Performs setup and initialization, and from a browser, accesses the host configuration user interface. You must have root access to run `hcstartup`.

`hcstartup` **Prompts**

```
There are Solaris for ISPs 1.0 components installed on
this machine. Before installing Solaris ISP Server 2.0,
this program will uninstall the old version. The
configuration data for these components will be saved and
they will be uninstalled (you may then select them for
reinstallation and the saved data will be restored).
Do you wish to proceed? (y/n) y
```

This message indicates that to proceed with the installation, Solaris for ISPs 1.0 components running on the machine will be uninstalled, data saved and used for upgrading to Solaris ISP Server 2.0.

```
Enter path to installation media
(enter ''none'' if no media) current working directory
```

The current working directory is displayed and is the default value (if you press Return). If there is no distribution media available, only uninstall options may be performed.

```
Enter a port number for the temporary web server [8000]
```

Port 8000 is the default port for the temporary Web server. `hcstartup` checks to see if the port is available, and prompts for another if it is in use.

```
Please choose one of the following options:
```

The default selection is `1` which will start HotJava browser for host configuration (if you press Return). The second option allows you to open a browser of your choice for the host configuration process. You can also abort the installation by selecting the third option.

`hcstartup` **Actions**

Behind the scenes, `hcstartup` does the following:

- Executes `uninstall-sisp1.0` script to:
  - Uninstall 1.0 components currently found installed on the machine.
  - Save the 1.0 configuration data for upgrading to 2.0.
- Copies host configuration archive information into `/tmp`.

- Starts a temporary web server for the graphical user interface.
- Copies information from the distribution media to `/var/opt/SUNWisp/hc/media`.
- Provides browser options to begin host configuration process for installation.
- When the configuration process is complete, cleans up `/tmp`.

FILES

`/tmp/hcstartup.running`	The presence of this file indicates that the utility is already running. `hcstartup` checks for this file when first invoked. If the file exists, another install and configuration process is already running, or one has exited improperly.
`/tmp/hcbi.running`	The presence of this file indicates that the batch install utility is running. `hcstartup` checks for this file when first invoked. If the file exists, another install and configuration process is already running, or one has exited improperly.
`/tmp/hc`*pid*	A number of temporary directories and files are created here. `hcstartup` removes them as a part of its cleanup.
`/var/opt/SUNWisp/hc/media`	Distributed media files are copied to this directory.

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving

SEE ALSO **ispIntro**(1M), **uninstall-sisp1.0**(1M), **hcjump**(1M), **hclfmd**(1M), **hclmfd.conf**(4)

NOTES If you run this command remotely, set the DISPLAY environment properly.

NAME

ispldap – creates entries in configuration files, for access by the ISP IDIA library functions (which return information required for accessing directory services). Also creates required directory services entries

SYNOPSIS

ispldap –d *RootBindDn* [–w *Password*] –s *"ServerName[:PortNumber] [Servername2[:PortNumber2] ...]"* –t *TopDN* [–q]

ispldap –h

DESCRIPTION

Records access information about LDAP servers configured on the network, for use by the ISP Directory Information API. Creates the `ou=`*componentID* and `ispVersion=`*versionNo* entries in the directory for services that have recorded their need for them during service installation. Root access is required to run this command.

An application that requires access to the directory services should create an empty file in `/etc/opt/SUNWisp/ldap/ispconf`. The file must be named in the form *componentID-version*, where *componentID* is a string uniquely identifying the software and *version* is the software version number. After `ispldap` runs, the file contains the bind DN and password for that portion of the directory tree where application-specific entries have been made. Therefore, set permissions on the file to protect that information appropriately.

OPTIONS

–d ***RootBindDn***

Specifies the distinguished name (DN) for binding to the directory as the directory administrator.

–h

Prints usage message and exits.

–q

Suppresses output messages and executes command in quiet mode.

–s ***"ServerName[:PortNumber] [Servername2[:PortNumber2] ...]"***

Indicates the number of directory servers configured on the network. One server host name is required. Additional servers (as appropriate) can be specified. In every case, the port number is optional. If no port number is specified, the default port (389) is assumed.

Note - In Solaris ISP Server™ 2.0, only a single, unreplicated directory server is supported.

–t ***TopDN***

Specifies the DN of the top Solaris ISP Server domain entry in the ISO tree.

–w ***Password***

Indicates the password for binding to the directory as the administrator.

Note - If you enter the password option on the command line, it is visible to anyone who can see your screen and to anyone issuing a `ps` command while `ispldap` is running. You can omit the –w option on the command line, avoiding these risks. The command then prompts you for the password, and your entry is not echoed on the screen.

EXAMPLES

EXAMPLE 1 Storing Information about an LDAP Server

The following shows a standard use of ispldap to record information about an LDAP server on the host “Groucho” running on the default LDAP port (389).

```
# /opt/SUNWisp/bin/ispldap -d ou=admin,o=sun,c=us -s "Groucho" -t o=sun,c=us
# Password:
```

EXAMPLE 2 Storing Information about an LDAP Server and Port

The following use of `ispldap` stores the bind information about an LDAP server on the host “Chico” running on port 2000:

```
# /opt/SUNWisp/bin/ispldap -d ou=admin,o=sun,c=us -s "Chico:2000" \
-t o=sun,c=us
# Password:
```

Also in this example,

- The bind DN is “ou=admin,o=sun,c=us.”

- The bind password is omitted from the command line arguments. The utility prompts for a password and does not echo it on the command line, keeping the password secure.
- The DN for the top level domain entry for the product in this example is "o=sun,c=us."

EXAMPLE 3 Storing Server Information using an IP Address

The following use of `ispldap` stores the bind information about an LDAP server at IP address 129.146.115.159 running on the default LDAP port (389).

```
# /opt/SUNWisp/bin/ispldap -d "ou=admin,o=sun,c=us" -s "129.146.115.159"
# Password:
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamr
Interface Stability	Evolving

SEE ALSO

`ispIntro`(3x), **`ispGetLdapInfo`**(3x), **`ispGetLdapServers`**(3x), **`ispGetTopDn`**(3x), **`IspLdapService`**(3x)

NOTES

If you modify the service entries in the directory services, particularly the distinguished name or the password, you must run this command again on the service host.

NAME

isprshp – set the port on which the Solaris ISP Server™ remote command execution daemon listens.

SYNOPSIS

isprshp [–p *PortNumber*]

DESCRIPTION

Use `isprshp` to change the port on which the remote command execution daemon listens. Run this command once on each host managed by Sun™ Internet Administrator™ and set Sun Internet Administrator to the same port (using its graphical user interface).

OPTIONS

–p ***PortNumber*** Enter a valid, available port number. If this option is omitted, the remote execution daemon is set to listen on its default port (50097).

EXAMPLES

EXAMPLE 1 Setting the Port Number

```
# /opt/SUNWisp/bin/isprshp -p 60320
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamr
Interface Stability	Evolving

SEE ALSO

NOTES

NAME

`mchelp` – Display the release version of Sun™ Internet Administrator™ installed on the system and list all utilities associated with it.

SYNOPSIS

mchelp

DESCRIPTION

Lists all command-line utilities associated with the installed version of Sun Internet Administrator. Displays the release version of the product. Run mchelp on the machine where Sun Internet Administrator is installed.

EXAMPLES

EXAMPLE 1 `mchelp` Output

Invoke `mchelp` by entering it at the command line:

```
/opt/SUNWisp/sbin/mchelp
```

The following output is displayed at your terminal:

```
Sun Internet Administrator release 1.1 commands:
```

mcadd	Adds a service to be managed
mcaddadm	Creates an administrator account
mcadmpwd	Sets an administrator password
mchelp	Lists all command-line utilities
mchostls	Lists all available services on a host
mcreg	Records information about a manageable user interface
mcrm	Removes a managed service from Sun Internet Administrator
mcrmadm	Deletes an administrator account
mcunreg	Deletes information about a manageable user interface

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving

SEE ALSO

mcadd(1m), **mcaddadm**(1m), **mcadmpwd**(1m), **mcdsinit**(1m), **mcdsclean**(1m), **mchostls**(1m), **mcreg**(1m), **mcrm**(1m), **mcrmadm**(1m), **mcunreg**(1m).

NAME

`mcreg` – Register a software component making it available for management through Sun™ Internet Administrator™. Overwrites any previous registration of the same component. Root access is required to run this command.

SYNOPSIS

mcreg –A –c *componentID* –d *doc_path* –i *icon_path* –j *classpath* –l –n *name* [–r *"servlet_info"* ...] –s *CGI_path* –v *componentVersion* –w *URL* –y *productVersion*

mcreg –c *componentID* –v *componentVersion* –y *productVersion*

mcreg –c *componentID* –i *icon_path* –l –n *name* –v *componentVersion* –w *URL* –y *productVersion*

mcreg –c *componentID* –i *icon_path* –n *name* –v *componentVersion* –y *productVersion* –x *X_path* –u *user_name* –g *group_name*

mcreg –c *componentID* –i *icon_path* –n *name* –v *componentVersion* –y *productVersion* –p "*prog_path*" –a [–d *help_file*]... –u *user_name* –g *group_name*

DESCRIPTION

Records user interface information for Sun Internet Administrator. Use the different forms of `mcreg` for the various supported user interface types. See the EXTENDED DESCRIPTION for a discussion of each form.

OPTIONS

–A	Indicates that the specified component is the graphical user interface of a three-tier, web-based service.
–c ***componentID***	Specifies the unique component identifier for the service being registered. The package name is recommended because it is guaranteed to be unique.
–d ***doc_path***	Specifies the path to all GUI files with the exception of CGI scripts. Specify the CGI location with the –s option.
–i ***icon_path***	Specifies the full path to an optional GIF format file to be used as an icon in the Sun Internet Administrator GUI.
–j ***classpath***	Specifies a Java classpath for any servlets being used. Required if servlets are used. A classpath is

	one or more colon-separated paths to class or jar files. Enter the full path for each.
–l	Indicates that Sun Internet Administrator should pass the locale to the component through the URL.
–n ***name***	Specifies a simple name, such as “FTP,” for use in the Sun Internet Administrator GUI.
–p ***"prog_path"***	Specifies command-line program information. Enclose the entire entry in quotes. It takes: ■ The full path to the command-line executable. If standard parameters are required, enter them here. For example: **–p ''/usr/bin/ps –ef''**. ■ –a Enter this option if parameters are accepted from the user at runtime. ■ –d *help_file* Enter this option, and the complete path to supporting documentation for the command-line utility. ■ –g *group_name* Enter the name of the UNIX user group under which the command or X-program should run. ■ –u *user_name* Enter the UNIX user name under which the command or X-program should be run.
–r ***"servlet_info"***	Specifies information required for each servlet used by the GUI being registered. For each servlet, the following information is required: ■ The path to the servlet, relative to the document root of the administration Web server. ■ The fully-qualified Java class name of the servlet. ■ Any required servlet arguments, listed by name and value. Specify one –r option for each servlet used by the GUI.

–s ***CGI_path***	Specifies the path to CGI used by the GUI, if any.
–v ***componentVersion***	Specifies the release version of the component being registered. This is required.
–w ***URL***	Specifies the path to the main (or opening) page of a Web-based GUI. This is the page that Sun Internet Administrator displays to the administrator when it receives a legitimate request to manage the service. For a three-tier application, enter a relative path. For a two-tier application, enter the full path to the page.
–x ***X_path***	Specifies the full path to an X-based administration program. It takes: ■ –g *group_name* Enter the name of the UNIX user group under which the command or X-program should run. ■ –u *user_name* Enter the UNIX user name under which the command or X-program should be run.
–y ***productVersion***	Specifies the product version that is displayed in the Sun Internet Administrator GUI. This is required.

EXTENDED DESCRIPTION

Three-Tier Web-Based GUIs

A three-tier Web-based GUI has one software component (ASCA) located on the same machine with Sun Internet Administrator and another (ASRA) located on the machine where the service resides. You must register both pieces.

To register the ASCA information, run `mcreg` on the Sun Internet Administrator host with –A, –c, –v, and –y options (see the first command synopsis).

To register the ASRA, run `mcreg` on the service host with the –c, –v, and –y options only (the second command synopsis).

Two-Tier Web-Based GUIs

To register a legacy GUI, which has only a software component on the service host, run `mcreg` on the service host using the third form of the command.

X-Based GUIs

To register the X-based administration program of a service, run `mcreg` on the service host using the fourth form of the command (including the `–x` option).

Command-Line User Interfaces

For each command-line utility supported by a service, run `mcreg` on the service host, using the fifth form of the command (including the `–p` option).

EXAMPLES

EXAMPLE 1 Registering a Three–Tier Application on the Sun Internet Administrator Host

The following illustrates use of `mcreg` for registering Sun™ Internet FTP Server™, which uses servlets for its three-tier graphical user interface. Run this form of the command on the machine where Sun Internet Administrator is installed.

```
# /opt/SUNWisp/sbin/mcreg
-A
-c SUNWftpa
-d /opt/SUNWftpa/1.1/doc
-i /opt/SUNWftpa/images/ftp.gif
-j classes:/opt/fre/lib/classes:/opt/joe/lib/classes.jar
-l
-n ftp
-r "main FTPAdmin.MainServlet"
-r "start FTPAdmin.StartServlet name=foo,owner=bar"
-s /opt/SUNWftpa/1.1/cgi-bin
-v 1.1
-y 1.1
-w cgi-bin/ftpadmin.cgi
```

The registration of Sun Internet FTP Server shows the use of all `mcreg` options that are valid for this form of the command.

`–A` Indicates that the software component being registered is a Web-base GUI component.

`–c` `SUNWftpa` is the package name for the user interface, and thus is a unique identifier for it.

`–d` Documentation for Sun Internet FTP Server resides at `/opt/SUNWftpa/1.1/doc`.

`–i` The icon to display for Sun Internet FTP Server is at `/opt/SUNWftpa/images/ftp.gif`.

`–j` The Java classpath for administration servlets being used includes:

- `classes`

- /opt/fre/lib/classes
- /opt/joe/lib/classes.jar

−l Indicates that the locale should be passed to the component through the URL.

−n The user-friendly name to be displayed in Sun Internet Administrator is “ftp.”

−r The first servlet being used by the service is MainServlet in the Java package FTPAdmin. It takes no arguments on startup.

−r The second servlet being used by the service is StartServlet in the Java package FTPAdmin. It takes two arguments on startup: name and owner.

−s The CGI scripts used by the service reside in /opt/SUNWftpa/1.1/cgi-bin.

−v The release version is 1.1.

−w The first page to be served up to an administrator accessing this service is at cgi-bin/ftpadmin.cgi. Because Sun Internet FTP Server is a three-tier Web-based service, this path is relative to the document root of Sun Internet Administrator’s Web server.

−y The product version is 1.1.

EXAMPLE 2 Registering a Three–Tier Application on Service Host

The following illustrates use of mcreg for registering Sun™ Internet FTP Server™. Run this form of the command on the machine where Sun Internet FTP Server is installed.

```
# /opt/SUNWisp/sbin/mcreg -c SUNWftp -v 1.1 -y 1.1 -n ftp
```

The registration of Sun Internet FTP Servershows the use of all mcreg options that are valid for this form of the command.

−c SUNWftpa is the package name for the user interface, and thus is a unique identifier for it.

`–v` The release version is 1.1.

`–y` The product version is 1.1.

`–n` The user-friendly name to be displayed in Sun Internet Administrator is “ftp.”

EXAMPLE 3 Registering a Two–Tier Application

The following illustrates the use of `mcreg` for registering Sun™ WebServer™. Run this form of the command on the machine where the Sun WebServer service is installed.

```
# /opt/SUNWisp/sbin/mcreg
-c SUNWhttp
-i /opt/SUNWhttp/images/http.gif
-n Sun WebServer
-v 2.1
-y 2.1
-w http://httphost:9001/admin/main.html
```

```
# /opt/SUNWisp/sbin/mcreg
-c SUNWhttp
-i /opt/SUNWhttp/images/http.gif
-n Sun WebServer
-v 2.1
-y 2.1
-w http://httphost:9001/admin/main.html
-l
```

In the foregoing examples, the options have the following meaning:

`–c` `SUNWhttp` is the package name for the user interface, and thus is a unique identifier for it.

`–i` The icon to display for Sun WebServer is at `/opt/SUNWhttp/images/http.gif`.

`–l` Indicates that Sun Internet Administrator should pass the locale to the component through the URL. In this example, the first page to be served up to an administrator accesssing this service from Sun Internet Administrator administration console will be:

`http://httphost:9001/admin/main.html?locale=en_US` where `en_US` refers to US English.

`–n` The user-friendly name to be displayed in Sun Internet Administrator is “Sun WebServer”.

`–v` The release version is 2.1.

`–w` The first page to be served up to an administrator accesssing this service is at `http://httphost/9001/admin/main.html`. Because Sun WebServer is a two-tier Web-based service, this is a complete URL to the administration server.

`–y` The product version is 2.1.

EXAMPLE 4 Registering an X-Based Application

The following illustrates use of `mcreg` for registering an X-based product. Run this form of the command on the machine where the service is installed.

```
# /opt/SUNWisp/sbin/mcreg
-c SUNWxtpa
-n XTP
-v 3.2
-y 3.1
-x /usr/Openwin/bin/xtpadmin -u root -g sys
```

In this example, the options have the following meaning:

`–c` `SUNWxtpa` is the package name for the user interface, and thus is a unique identifier for it.

`–n` The user-friendly name to be displayed in Sun Internet Administrator is “XTP.”

`–v` The release version is 3.2.

`–x` The executable for the administration interface is at `/opt/SUNWxtpa/bin/xtpadmin`. It takes:

- `–g` *group_name* This option indicates that the UNIX user group name under which the administration interface of the X-program should run is `sys`

- –u *user_name* This option indicates that the UNIX user name under which the administration interface of the X-program should run is root.

–y The product version is 3.1.

EXAMPLE 5 Registering a Command-Line Interface

The following illustrates use of `mcreg` for registering a command-line utility. Run this form of the command on the machine where the service is installed.

```
# /opt/SUNWisp/sbin/mcreg
-c SUNWcli
-n SampleCLI
-p /usr/bin/ps
-p "/usr/bin/date" -a -d /usr/bin/datehelp.html -u root -g sys
-v 1.1
-y 1.1
```

In this example, the options have the following meaning:

–c `SUNWcli` is the package name for the user interface, and thus is a unique identifier for it.

–n The user-friendly name to be displayed in Sun Internet Administrator is “SampleCLI”.

–p Two command-line utilities are registered in this example.

In the first, `ps` is located at `/usr/bin/ps`. It takes no parameters and has no help file associated with it.

In the second, `date` is located at `/usr/bin/date`. Note that the complex argument of this option is enclosed in quotation marks. It takes:

- –a This option indicates that it accepts parameters from the user at runtime.
- –d This option indicates that its help file is located at `/usr/bin/datehelp.html`.
- –g *group_name* The option indicates that the UNIX user group name under which the X-program should run is `sys`
- –u *user_name* This option indicates that the UNIX user name under which the X-program should run is root.

`-v` The release version is 1.1.

`-y` The product version is 1.1.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving

SEE ALSO

`mcIntro`(1m) **`mcadd`**(1m), **`mcaddadm`**(1m), **`mcadmpwd`**(1m), **`mcdsinit`**(1m), **`mchelp`**(1m), **`mchostls`**(1m), **`mcdsclean`**(1m), **`mcrm`**(1m), **`mcrmadm`**(1m), **`mcunreg`**(1m)

NOTES

Use the option `-l` only if the component supports receiving locale information through the URL. We recommend using the browser to set your language preferences. This option is intended for browsers that do not support HTTP1.1 content negotiation.

All Solaris ISP Server components have the same component and product version except Sun™ Directory Services whose component version is 3.2 (for `-v`) and product version is 3.1 (for `-y`).

NAME

`mcunreg` – Unregister a software component GUI, making it unavailable to Sun™ Internet Administrator™.

SYNOPSIS

mcunreg –c *componentID* –v *componentVersion*

mcunreg –A –c *componentID* –v *componentVersion*

DESCRIPTION

Removes information about a service's user interface, so that the service cannot be managed by Sun Internet Administrator (for example, on deinstallation). For all supported user interface types, run the first version of the command, with –c and –v options, on the machine where the service is installed. For three-tier Web-based user interfaces, run the second version of the command as well, specifying –A, –c, –v options, on the machine where Sun Internet Administrator is installed. Root access is required to run this command.

OPTIONS

–A	Specifies an administrative Web interface is being unregistered. Required in that case.
–c ***componentID***	Specifies the software component being unregistered, by unique component identifier (package name). This option is required, and must match the *componentID* specified when `mcreg` was called to register the component.
–v ***componentVersion***	Specifies the release version of the software component being unregistered This option is required, and must match the *componentVersion* specified when `mcreg` was called to register the component.

EXAMPLES

EXAMPLE 1 `mcunreg` on the Service Host

To unregister all supported types of user interfaces, run this form of `mcunreg` on the host where the service is installed.

```
# /opt/SUNWisp/sbin/mcunreg
-c SUNWftpa
-v 1.1
```

When running `mcunreg`, the –c and –v options must exactly match those used to register the software (`mcreg`).

EXAMPLE 2 `mcunreg` on the Sun Internet Administrator Host

To unregister a three-tier Web-based user interface, run this form of `mcunreg` on the Sun Internet Administrator host in addition to running the other form on the service host.

```
# /opt/SUNWisp/sbin/mcunreg
-A
-c SUNWftpa
-v 1.1
```

When running `mcunreg`, the –`c` and –`v` options must exactly match those used to register the software (`mcreg`). On the Sun Internet Administrator host, use the –`A` option to indicate that this is the ASCA software component for the service.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving

SEE ALSO

`mcIntro`(1m) **`mcadd`**(1m), **`mcaddadm`**(1m), **`mcadmpwd`**(1m), **`mcdsinit`**(1m), **`mchelp`**(1m), **`mchostls`**(1m), **`mcdsclean`**(1m), **`mcreg`**(1m), **`mcrm`**(1m), **`mcrmadm`**(1m).

NOTES

Running this command completely disables the service for administration from Sun Internet Administrator.

NAME

sisploa d – converts a file of user entries into an ldif file for loading into the Solaris ISP Server directory service

SYNOPSIS

sisploa d [–b *baseDN*] [–d *debugLevel*] –f *InputFile* [–h *DirectoryHost*] [–m *mappingFile*] [–o *outputFile*] [–t *table*] [–V *variable=value...*]

DESCRIPTION

This utility takes a data file with subscriber entries in the format described in **sisploa d.mapping**(4) and creates an ldif file that can be used to create the subscribers in a Sun Directory Services data store.

The mapping file defines how sisploa d will interpret each field in the input file and how those fields will be translated into entries in the directory service.

The output file can be used with **ldapmodify**(1M) to create the directory entries.

OPTIONS

–b *baseDN*

Specifies a base distinguished name (DN) to use as the root of all entries. The default is specified in the BASE_DN entry in the **sisploa d.mapping**(4) file, which is set to the base DN named when Solaris ISP Server was installed. Use this option to override the setting in the mapping file.

–d *debugLevel*

Sets the logging level for the **dsservd**(1M) daemon. You can set any combination of the following levels:

Mask	Description
1	Trace
2	Packets
4	Arguments
8	Connections
16	BER
32	Filters
64	Configuration
128	Access Control
256	Statistics (summary)

	512	Statistics (detailed)
	1024	Not Used
	2048	Parse
	65535	All information
–f *InputFile*	Specifies the file which contains raw data to be mapped to ldif entries. The raw data comes from some other data source, such as a database or NIS files. Each line in the input file corresponds to an entry or a change to an entry to be made in the directory service. See **sispload.mapping**(4) for information on the syntax rules for this file.	
–h *DirectoryHost*	Specifies the host name of the directory server to which the user entries will be added. New entries will be checked against the directory on this host to determine whether new entries should be added or should modify existing user entries. If no host is specified, the local machine is the default.	
–m *mappingFile*	Specifies the mapping file to use to translate input entries into directory entries. If no mapping file is specified, `sispload.mapping` in `/etc/opt/SUNWconn/ldap/current/mapping` is used. See **sispload.mapping**(4) for details on the format of the mapping file.	
–o *outputFile*	Specifies a file root where output will be sent instead of `STDOUT` and `STDERR`. Two files will be created: *outputFile*`.ldif` contains ldif-format changes to make to the directory, and *outputFile*`.err` contains error and warning messages.	
–t *table*	Specifies the table in the mapping file to use to create user entries. Valid parameters are ■ `sisp` - to create users with the `ispSubscriber` object class. ■ `sisprad` - to create users with the `ispSubscriber` and `remoteUser` (RADIUS user) object classes.	

- `sispsims` - to create users with the `ispSubscriber` and `emailPerson` object classes.
- `sispradsims` - to create users with the `ispSubscriber`, `remoteUser`, and `emailPerson` object classes.

If `–t` is not specified, the `sisp` table is used.

`–V` *variable=value...* Sets variables for use in the conversion. Repeat the `–V` flag to define multiple variables. For example:

```
-V BASE_DN=o=myISP,c=US \
-V SUNMS=mailbox
```

EXTENDED DESCRIPTION

The `sispload` utility is designed to migrate large sets of user data from one source into `ispSubscriber` entries in the Solaris ISP Server directory server. Any source data may be used, as long as it can be exported into an ASCII file in the format understood by `sispload`. The manual page for **`sispload.mapping`**(4) describes the syntax for the input file.

For each line in the input file, `sispload` attempts to construct an appropriate directory in ldif format. The entry contains the distinguished name, object class definitions, and all applicable attributes for the object classes. Data may not be available for some attributes. Use blank fields in the input file, and `sispload` will either ignore the attribute or set it to an appropriate default value.

For each entry created, `sispload` checks the directory on the named host (or the local host) to discover whether an entry already exists with matching data. If a matching subscriber entry is found, `sispload` attempts to create an ldif modify record instead of an add. If neither a modify nor an add record cannot be created, an error will be logged in the error file.

Before you make actual changes to the directory, you should run `ldapmodify` with the `–n` option to preview the changes that would be made. For example:

```
# ldapmodify -D 'cn=admin,o=XYZ,c=US' -w secret -h ldap2.xyz.com \-n -f bulkload.ldif
```

If there are errors, you may need to modify some of the entries that are being made, or you may need to delete some entries from the ldif file and modify them manually.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixsds
Interface Stability	Evolving

SEE ALSO

`sisploa d.mapping`(4), **`dsimport`**(1M)

NAME

uninstall-sisp1.0 – Perform a non-interactive uninstall of Solaris for ISPs 1.0 components and save the old data.

SYNOPSIS

uninstall-sisp1.0 [*media_root*]

DESCRIPTION

Uninstalls Solaris for ISPs 1.0 components, if found installed, and saves the old data for upgrading to Solaris ISP Server 2.0. You must have root access to run this script. If you are upgrading from the:

- Host configuration software (using `hcstartup`), this script is automatically executed.
- Command line, you must execute this script before upgrading to Solaris ISP Server 2.0 software.

OPERANDS

media_root	The path to the root of the installation media for invoking this script.

`uninstall-sisp1.0` **Prompts**

```
There are Solaris for ISPs 1.0 components installed on
this machine. Before installing Solaris ISP Server 2.0,
this program will uninstall the old version. The
configuration data for these components will be saved and
they will be uninstalled (you may then select them for
reinstallation and the saved data will be restored).
Do you wish to proceed? (y/n)
```

If the script is executed by `hcstartup`, this prompt asks you to specify whether or not you wish to uninstall Solaris for ISPs 1.0 components, currently running on the machine, and save the data for an upgrade. You will not get this prompt if you execute this command from the command line.

`uninstall-sisp1.0` **Actions**

When invoked,

- From the command line:
  - Uninstalls 1.0 components currently installed on the machine.
  - Saves 1.0 data in a component specific directory.
- By `hcstartup`, requests your consent before proceeding to:
  - Uninstall 1.0 components currently found installed on the machine.
  - Save the 1.0 component data in a component specific directory.

ENVIRONMENT VARIABLES

See **environ**(5) for descriptions of the following environment variables that affect the execution of `command_name`: `NLSPATH`.

EXIT STATUS

The following exit values are returned:

`0` Successful completion. Uninstalled 1.0 components and saved data successfully.

`1` Exited silently. Did not find 1.0 components installed for upgrading.

`2` An error occurred or the root to the installation media is incorrect.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving

SEE ALSO

ispIntro(1M), **hcstartup**(1M), **hcjump**(1M), **hclfmd**(1M), **hclmfd.conf**(4)

CHAPTER

man Pages(3x): Miscellaneous Library Functions

NAME

ispIntro – introduction to the ISP Directory Information API (IDIA) man pages

DESCRIPTION

The ISP Directory Information API (IDIA) contains C language library functions and a Java class that provide information to enable read-write access to information stored in the Lightweight Directory Access Protocol (LDAP) directory service. It supports ISP services that need to bind to the LDAP server and update service information.

The information provided by IDIA is determined during the Solaris ISP Server host configuration process and directory services initialization. The `ispldap` command-line utility stores this information for use by the API.

RETURN VALUES

See individual man pages for the return values of the C functions.

Each method in the Java class `ispLdapService` throws a Java `IOException`.

ATTRIBUTES

See **`attributes`**(5) for descriptions of the following attributes.

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWispm

FILES

/opt/SUNWisp/include/isp_dir_api.h	The C header file for the LDAP directory services access API.
com.sun.isp.idia	The Java package for the `IspLdapService` class.
/opt/SUNWisp/include/mcauth.h	The C header file for **ISPMC_aar()**.

SEE ALSO

`ispldap`(1M), **`mcreg`**(1M)

NOTES

`ispGetLdapInfo(3X)`	This C function provides the distinguished name and password for binding to the LDAP server with access to a particular region of the directory information tree (DIT).
`ispGetLdapServers(3X)`	This C function provides the names and port numbers of LDAP directory servers configured on the network.
`ispGetTopDn(3X)`	This C function provides the distinguished name of the root domain (top-level domain entry in the DIT, under which Solaris ISP Server information is stored).

ispLdapService(3X)	This Java class provides information on LDAP servers configured on the network. Various class methods return the root domain entry in the DIT and distinguished names and passwords for binding to the directory.
ISPMC_aar(3X)	This C function accepts the user name and password of an administrator of Sun Internet Administrator and validates against the entry in the directory services. It first checks the name and password, to authenticate that the user exists. Then it checks the component identifier and version provided to determine whether this user is authorized to access that service. The results of the check are passed back by means of the *access* parameter.

NAME

ispGetLdapInfo – get the disinguished name and password for binding to an Lightweight Directory Access Protocol (LDAP) server with rights to update a particular region of the directory information tree (DIT)

SYNOPSIS

```
#include <isp_dir_api.h>

int ispGetLdapInfo(const char *componentId, const char *versionNo, char **ldapBindDn,
char **ldapBindPasswd, char **compConfigDn);
```

DESCRIPTION

Accepts the component identifier and version number for a particular software component and provides the distinguished name and password for binding to an LDAP server with access to update a particular portion of the DIT. Also provides the distinguished name of a component configuration entry if one exists.

PARAMETERS

The parameter descriptions follow in alphabetical order.

compConfigDn An output argument that points to a Null-terminated string containing the distinguished name of a component configuration entry (if one exists) associated with the *componentId* and *versionNo*. If no such entry exists, this pointer is set to NULL.

componentId An input argument that points to a character constant containing the identifier of the particular software component (service) of interest. Typical component identifiers are package names (or slight variants) as these are guaranteed to be unique.

ldapBindDn An output argument that points to a Null-terminated string containing the distinguished name for binding to the LDAP server with access to the portion of the DIT associated with the component passed in *componentId* and *versionNo*.

ldapBindPasswd An output argument that points to a Null-terminated string containing the password for binding to the LDAP server with access to the portion of the DIT associated with the component passed in *componentId* and *versionNo*.

versionNo An input argument that points to a character constant containing the version number of the particular component of interest. Typical component version numbers are in the form *major.minor* (for example, 1.0).

EXTENDED DESCRIPTION

Use the information provided by the **ispGetLdapInfo()** function to bind to a known LDAP server (using the LDAP client library API), and to access component-specific configuration information if it exists.

Services that need read-only access can bind to the directory unauthenticated and perform searches and compares. This API provides information required to gain write access. Remember that the server access controls must be set to allow write access.

The caller of **ispGetLdapInfo()** is reponsible for calling **free()** on *compConfigDn*, *ldapBindDn*, and *ldapBindPasswd*.

RETURN VALUES

The **ispGetLdapInfo()** function returns the following.

`0` Successful completion.

`-1` An error occurred. The error number in `errno` is set as shown in ERRORS.

ERRORS

The **ispGetLdapInfo()** function sets the following `errno` values when an error occurs.

`ENOENT` No entry exists for the specified *componentId* and *versionNo*.

`EIO` The function is unable to read LDAP information from the disk. The underlying file may not exist. Use `ispldap(1m)` to create it.

`EACCESS` The user does not have permission to access the LDAP information file on disk.

EXAMPLES

EXAMPLE 1 Getting a Bind DN for Directory Services

The following shows a typical use of **ispGetLdapInfo()**.

```
char *ftpLdapDn;
char *ftpLdapPasswd;
char *ftpCompDn;
if(ispGetLdapInfo(
                  "SUNWftp",
                  "1.0",
                  &ftpLdapDn,
                  &ftpLdapPasswd,
                  &ftpCompDn)==0)
{
    ldap_bind(ld, ftpLdapDN, ftpLdapPasswd);

    /*Do other processing as appropriate
      Use the ISP-wide root domain DN to find the
      configuration information.
    */
```

```
    if(ftpLdapDn) free(ftpLdapDn);
    if(ftpLdapPasswd) free(ftpLdapPasswd);
    if(ftpCompDn) free(ftpCompDn);
}
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving
MT_Level	Safe

FILES

/opt/SUNWisp/include/isp_dir_api.h C header file for the LDAP directory services access API.

SEE ALSO

ispIntro(3X), **ispGetLdapServers**(3X), **ispGetTopDn**(3X), **IspLdapService**(3X), **ISPMC_aar**(3X)

ispldap(1M), **mcreg**(1M)

NAME

ispGetLdapServers – get the names and port numbers of Lightweight Directory Access Protocol (LDAP) servers configured on the network

SYNOPSIS

```
#include <isp_dir_api.h>

int ispGetLdapServers(char **listOfLdapServers);
```

DESCRIPTION

Provides a pointer to a Null-terminated character string containing a list of LDAP servers and port numbers.

The caller of **ispGetLdapServers()** is reponsible for calling **free()** on the *listOfLdapServers* string.

PARAMETERS

The parameter descriptions follow in alphabetical order.

listOfLdapServers An output argument that points to a Null-terminated character string containing a list of LDAP servers and port numbers in the following syntax:

```
hostName1[:portNum1] [hostName2[:portNum2] … ]
```

Note that *hostName* and its associated port number are separated by a colon (:), while different *hostNames* are separated by spaces.

The server host name is returned in a form that matches the parameter accepted by the LDAP client library, which could be an IP address in dot notation or a DNS name, and can be passed directly through to those function calls. If no LDAP servers are configured, *listOfLdapServers* is an empty string. If *listOfLdapServers* lists a server without a port number, that server is using the default LDAP port (389).

Note - In Solaris ISP Server™ 2.0, only a single directory server is supported. The only replication supported is when configuring the dirctory for operation with Sun Internet Mail Server.

RETURN VALUES

The **ispGetLdapServers()** function returns the following.

`0` Successful completion.

`-1` An error occurred. The error number in `errno` is set as shown in .

ERRORS

The **ispGetLdapServers()** function sets the following `errno` values when an error occurs.

`EIO` The function is unable to read LDAP information from the disk. The underlying file may not exist.

`EACCESS` The user does not have permission to access the LDAP information file on disk.

EXAMPLES

EXAMPLE 1 Getting a List of LDAP Servers

The following shows a standard use of **ispGetLdapServers()**.

```
char* ldapServers;
if(ispGetLdapServers(&ldapServers)==0)
{
    ldap_init(ldapServers, LDAP_PORT);
    /*other use of the directory server*/
    free(ldapServers);
}
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving
MT_Level	Safe

FILES

/opt/SUNWisp/include/isp_dir_api.h C header file for the LDAP directory services access API.

SEE ALSO

ispIntro(3X), **ispGetLdapInfo**(3X), **ispGetTopDn**(3X), **IspLdapService**(3X), **ISPMC_aar**(3X)

ispldap(1M), **mcreg**(1M)

NAME

ispGetTopDn – get the distinguished name (DN) of the Solaris ISP Server™ top level entry

SYNOPSIS

```
#include <isp_dir_api.h>

int ispGetTopDn(char **ispGlobalDn);
```

DESCRIPTION

Provides a pointer to a Null-terminated string containing the DN of the top-level Solaris ISP Server entry in the directory information tree (DIT). Sun™ Internet Administrator™ uses the *ispGlobalDn* to locate component-specific configuration information and administrator information in the directory. Service components should use the *compDn* returned by the **ispGetLdapInfo()** function.

The caller of **ispGetTopDn()** is reponsible for calling **free()** on *ispGlobalDn*.

PARAMETERS

The parameter descriptions follow in alphabetical order.

ispGlobalDn An output argument that points to a Null-terminated string containing the distinguished name of the root domain entry in the DIT.

RETURN VALUES

The **ispGetTopDn()** function returns the following.

`0` Successful completion.

`-1` An error occurred. The error number in `errno` is set as shown in ERRORS.

ERRORS

The **ispGetTopDn()** function sets the following `errno` values when an error occurs.

`ENOENT` No entry exists in the configuration file, or the file does not exist.

`EIO` The function is unable to read Lightweight Directory Access Protocol (LDAP) information from the disk. The underlying file may not exist.

`EACCESS` The user does not have permission to access the LDAP information file on disk.

EXAMPLES

EXAMPLE 1 Getting the root DN

The following shows a standard use of **ispGetTopDn()**.

```
char * ispTopDn = NULL;
if((ispGetTopDn(&ispTopDn)) < 0 ) {
```

```
  /check errno*/
}
else {
  printf("isp Top Dn = %s \n", ispTopDn);
  /*other processing using ispTopDn*/
}
if(ispTopDn) free(ispTopDn);
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving
MT_Level	Safe

FILES

/opt/SUNWisp/include/isp_dir_api.h C header file for the LDAP directory services access API.

SEE ALSO

ispIntro(3X), **ispGetLdapInfo**(3X), **ispGetLdapServers**(3X), **IspLdapService**(3X), **ISPMC_aar**(3X)

ispldap(1M), **mcreg**(1M)

NAME

IspLdapService – a Java class that provides information on Lightweight Directory Access Protocol (LDAP) servers configured on the network by Solaris ISP Server

SYNOPSIS

```
package com.sun.isp.idia;
public class IspLdapService()   {

  public IspLdapService(String Svc, String Vers)

      throws IOException {}

  public String Servers() throws IOException {}

  public String TopDn()  throws IOException {}

  public String BindDN() throws IOException {}

  public String BindPasswd() throws IOException {}

}
```

DESCRIPTION

The `IspLdapService` class uses Java Native methods Interface (JNI) to deliver the same information provided by the C library functions **ispGetLdapServers()**, **ispGetLdapInfo()**, and **ispGetTopDn()**. Because of the use of native methods, and for security reasons (`IspLdapService` accesses files on disk), it cannot be used from an applet.

METHODS

The public methods of `IspLdapService` follow in alphabetical order.

BindDN()

public `String` BindDN();

Returns a `String` object containing the DN to be used by the service to bind to the LDAP server. This DN (and its password) is associated with regions of the directory information tree (DIT) defined by the *Service* and *Version* parameters passed when this `IspLdapService` object was constructed. When a service requires privileged access (for example, to update entries in the directory) it binds using this DN.

BindPasswd()

public `String` BindPasswd();

Returns a `String` object containing the password to be used by the service to bind to the LDAP server. The DN and password are associated with regions of the DIT defined by the *Service* and *Version* parameters passed when this `IspLdapService` object was constructed.

IspLdapService()

public `IspLdapService(String` *Service*, `String` *Version*);

Constructs an `IspLdapService` object where

Service Is a `String` object containing the component identifier of the service whose directory information is needed. A component identifier must be unique, and may be a package name or variant.

Note - The component identifier must match that recorded by the `mcreg` command when the service software was installed and configured.

Version Is a `String` object containing the version number of the service whose directory information is needed. The version number is typically in the form *major.minor* (for example, 1.0).

Note - The version number must match that recorded by the `mcreg` command when the service software was installed and configured.

Each component that uses the directory services has its own region of the DIT for its component-specific information. The constructor initializes the `String` object returned by **BindDN()** to the distinguished name for binding to that portion of the DIT. It initializes the `String` object returned by **BindPasswd()** to the password attribute for that entry.

Servers()

public `String` Servers() ;

Returns a `String` object containing a list of LDAP servers configured on the network. If no servers are configured, an empty `String` is returned. The `String` has the following syntax:

hostName1[:portNum1] [hostName2[:portNum2] . . .]

Note that *hostName* and its associated port number are separated by a colon (:), while different *hostNames* are separated by spaces. The server host name is returned in a form that matches the parameter accepted by the LDAP client library, which could be an IP address in dot notation or a DNS name, and can be passed directly through to those function calls.

Note - In Solaris ISP Server™ 2.0, only a single, directory server is supported. The only replication supported is when configuring the directory to operate with Sun Internet Mail Server.

TopDn()

public `String` TopDn() ;

Returns a `String` object containing the distinguished name of the root domain entry in the DIT (the top-level entry under which Solaris ISP Server information is stored).

EXTENDED DESCRIPTION

The `IspLdapService` class is designed to work with Java Naming and Directory Interface (JNDI) to access information from any standard directory service without requiring specific directory knowledge on the part of the application. JNDI uses several JNDI Environment Properties to specify the LDAP server and the user name and credentials the application will use to bind to it. These are:

- java.naming.provider.url (Context.PROVIDER_URL)
- java.naming.dns.url (Context.DNS_URL)
- java.naming.security.principal (Context.SECURITY_PRINCIPAL)
- java.naming.security.credentials (Context.SECURITY_CREDENTIALS)

Each of these takes a Java `String` object and can be set as env.put(Context.XXX), by the application. An example of how to use `IspLdapService` with JNDI is included below.

EXCEPTIONS

Each of the public methods of `IspLdapService` throws a Java `IOException` when an error occurs in reading the configuration file from disk. Remember to handle these exceptions in your code.

EXAMPLES

EXAMPLE 1 Using `IspLdapService` with JNDI

```
// JNDI packages
import javax.naming.*;
import javax.naming.directory.*;
// ISP directory information classes
import com.sun.isp.idia.*;
...
{
  ...
  String ldapServers;
  String topDN;
  String bindDN;
  String bindPwd;
```

```
    // Fetch LDAP server, bindDN and password
    // from Isp Dir Info API-classes and bind
    // to the LDAP server
    IspLdapService ftpService = new IspLdapService("SUNWftp", "1.0");

      try {
        ldapServers = ftpService.Servers();
        topDN = ftpService.TopDN();
        bindDN = ftpService.BindDN();
        bindPwd = ftpService.BindPasswd();
      }
      catch (IOException e) {
        System.err.println("Error determining Ldap information");
          return;
      }

    // LDAP initialization
    Hashtable env = new Hashtable(5, 0.75f);
    env.put(javax.naming.Context.INITIAL_CONTEXT_FACTORY,
    "com.sun.jndi.ldap.LdapCtxFactory");
    env.put(javax.naming.Context.PROVIDER_URL, "ldap://" + ldapServers);
    env.put(javax.naming.Context.SECURITY_PRINCIPAL, bindDN);
    env.put(javax.naming.Context.SECURITY_CREDENTIALS, bindPwd);

    ...

  }
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving
MT_Level	Safe

FILES

com.sun.isp.idia The Java package for the `IspLdapService` class.

SEE ALSO

ispIntro(3X), **ispGetLdapServers**(3X), **ispGetLdapInfo**(3X), **ispGetTopDn**(3X), **ISPMC_aar**(3X)

ispldap(1M), **mcreg**(1M)

NAME

ISPMC_aar – authenticate a user of Sun Internet Administrator and verify access authorization to the given software component

SYNOPSIS

```
#include <mcauth.h>

int ISPMC_aar(char *admin_name, char *password, char *componentID, char *version, int *access);
```

DESCRIPTION

This function accepts the user name and password of a user of Sun Internet Administrator and validates against the entry in the directory services. It first checks the name and password, to authenticate that the user exists. Then it checks the component identifier and version provided to determine whether this user is authorized to access that service. The results of the check are passed back by means of the *access* parameter.

PARAMETERS

admin_name	Points to a string containing the user name of the administrator being authenticated.
password	Points to a string containing this administrator's password.
componentID	Points to a string containing a unique component identifier of the software component the administrator is accessing.
version	Points to a string containing version number (major.minor) of the software component the administrator is accessing.
access	An output argument that points to an integer specifying the results of the check. If *access* is 1, the administrator is authenticated and authorized for access to the desired service. If *access* is zero (0), the administrator is not authorized, or the administrator entry does not exist (this is an invalid user).

RETURN VALUES

The **ISPMC_aar()** function function returns the following.

`0`	The function has exited properly. The results of the authentication and access check are in the *access* parameter.
`nonzero`	The function has exited with an error. The user has not been validated and the contents of *access* are insignificant.

Note - Because this value can be set by a number of different programs, all using similar error numbers for different reasons, you should not rely on this value to determine a specific reason for the failure.

EXAMPLES

EXAMPLE 1

```
#include <nl_types.h>
#include "mcauth.h"
nl_catd          catd;

main()
{
  char *name = "john";
  char *pwd = "smith";
  char *comp = "SUNWfoobar";
  char *version = "1.0";
  int  err, access;

  if ( (err = ISPMC_aar(name, pwd, comp, version, &access))== 0) {
    if (access == 1)
      printf ("Authenticated and authorized\n");
    else
      printf ("Authentication or authorization failed\n");
  } else {
      printf ("Error encountered\n");
  }
}
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamr
Interface Stability	Evolving

FILES

/opt/SUNWisp/include/mcauth.h The C header file for **ISPMC_aar()**.

SEE ALSO

ispIntro(3X), **ispGetLdapServers**(3X), **ispGetLdapInfo**(3X), **ispGetTopDn**(3X), **IspLdapService**(3X)

ispldap(1M), **mcreg**(1M)

man Pages(4): File Formats

NAME

ispIntro – introduce the man pages for Solaris ISP Server foundation configuration files

DESCRIPTION

This page describes all of the man pages available for configuration files associated with Solaris ISP Server platform components.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWispm
Interface Stability	Stable

FILES

`/etc/opt/SUNWisp/hclfmd.conf`

`/etc/opt/SUNWisp/ldap/lib/sisploa d.mapping`

SEE ALSO

hclmfd(1m), **sisploa d**(1m)

NOTES

hclmfd.conf(4) configures the Solaris ISP Server host configuration log file management daemon.

sisploa d.mapping(4) defines the translation methods used by **sisploa d**(1m) to convert user records from a structured text file to ldif entries.

NAME

`hclfmd.conf` – Host configuration log file management daemon configuration file.

SYNOPSIS

`/etc/opt/SUNWisp/hclfmd.conf`

DESCRIPTION

The `hclfmd.conf` file contains the list of log files to be monitored by `hclfmd`. Each entry in the file has the following form:

`interval:`*minutes*
log_file`:`*command*

where

interval Specifies the interval (in minutes) at which the listed log files are monitored. The interval can range from 1 minute to 10080 minutes (1 week).

minutes Is the interval at which the log files are monitored. By default, the interval is set to one minute; that is, the log files are monitored every minute. You can modify the interval at which the log files are monitored by changing the interval values.

log_file Is the complete path to the log file to be monitored. There should be an `auth.err` entry in `/etc/syslog.conf` for this file.

command Is the notification command to run if new entries are found in the monitored file. The command is passed to `/usr/bin/sh` for execution, and may contain the following variables:

"%f" Replaced with the name of the file where the new entries are found.

%c Is a temporary file containing the new contents of the monitored log file.

Note - If your command requires a literal percent sign (%), enter **%%**.

EXAMPLES

EXAMPLE 1 Default `hclfmd.conf` entry

```
interval:1
/var/log/badauth:/usr/bin/mailx -s %f root < %c
```

checks the log file (badauth) every minute and runs the `mailx` command every time it detects new entries in the log file with subject set to "%f" and the mail body being whatever is in file "%c".

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWisp
Interface Stability	Evolving

SEE ALSO

ispIntro(1M) **hcjump**(1M), **hclfmd**(1M), **hcstartup**(1M)

NAME

sisploadmapping – defines the translation of user records from a structured text file to ldif records for adding Solaris ISP Server subscribers

SYNOPSIS

`/etc/opt/SUNWisp/ldap/sunds/default/mapping/sisploadmapping`

DESCRIPTION

The `sisploadmapping` file contains the default rules for translating a file of user records into ldif records. The file is used with **`sisploadmapping`**(1m) to create an output file of ldif records for each user. The resulting ldif file can be used with **`ldapadd`**(1m) to create records for all of the user records in the directory server.

The Extended Description section explains the default rules defined in `sisploadmapping` and the expected syntax of the input files. You need to create a new mapping file if your input files are in another format or your directory schema has been altered from the Solaris ISP Server default. The mapping file follows the syntax of mapping files used by **`dsimport`**(1m), and you should read the `dsimport` man page or see the *Sun Directory Services 3.1 Administration Guide*.

Note - The syntax of the mapping file is complicated. It may be very difficult to create a mapping file that properly converts data into records for your directory. Editing the mapping file is not recommended.

EXTENDED DESCRIPTION

This section explains the syntax of the input file containing user data, the default mapping file, and the way the output ldif file is constructed from the input.

You must extract your current user data into an ASCII file with one entry per line according to the syntax described here. If you cannot get data that matches this syntax exactly, you may need to create a new mapping file to match the syntax of your file. You should change only the way lines are parsed in the `LINE` entries of the Extract section in the mapping file.

For more details on the use, syntax, and valid data for the attributes that are constructed, please refer to the following sources:

- *Solaris ISP Server 2.0 Administration Guide* for information about the `ispSubscriber` object class. All of the `sisploadmapping` conversion tables use this class.
- *Sun Directory Services 3.1 Administration Guide* for information about the `remoteUser` and `emailPerson` object classes used to create RADIUS and Sun Internet Mail Server users.

Input File Syntax

Each line in the input file corresponds to a resulting record in the ldif file and the directory. Each line is a series of fields separated by a delimiter.

The default delimiter is an exclamation point (!). You may change the delimiter by changing the value of SEPARATOR in the Dynamic section of the mapping file. You must also change the LINE entries because they are constructed with ! as the delimiter. Do not use a comma (,) or a space as the delimiter because records which contain distinguished name data will use commas and spaces as part of the data.

If your data does not include one of the expected fields, you may be able to use a null field between separators in the input file. For example, if your data does not include the third field expected in the input:

```
!data1!data2!!data4!...
```

Only optional fields may be omitted. Optional data is marked with an asterisk (*) at the end of the token name in the mapping file, and noted as optional in the descriptions below.

This section describes the form of the input file. For details on the object classes and attributes constructed from the input file see the Output section below.

There are four forms for the input file corresponding to four translation tables:

- sisp is used to create user records using the ispSubscriber object class.
- sisprad is used to create user records using the remoteUser object class and the ispSubscriber object class so that RADIUS authentication data is included.
- sispsims is used to create user records using the emailPerson object class and the ispSubscriber object class so that Sun Internet Mail Server data is included. Use this table if you have configured a Sun Internet Mail Serverdirectory as a slave of the Solaris ISP Server directory.
- sispradsims is used to create user records using the emailPerson, remoteUser, andispSubscriber object classes so that Sun Internet Mail Server data and RADIUS authentication data is included.

The table that is used is selected with the –t option for sispload. The default table is sisp.

The following subsections describe the input fields for each table.

sisp Input

Each line in the input file for the sisp table has the following syntax:

```
!LastName!FirstName!Nickname!Username!Password!DomainName!\
EmailAddress!Services!UID!GID!ContentDirectory!
```

Where:

LastName Contains the real last name.

FirstName Contains the real first name.

Nickname (Optional) Contains an optional nickname or alias.

Username Contains the user name used to log on to the network.

Password (Optional) Contains the user's password.

DomainName (Optional) Contains the fully qualified domain name to which this user belongs.

E-mailAddress (Optional) Contains this user's fully qualified email address, including the domain name.

Services (Optional) Contains a space-separated list of Solaris ISP Server `ispService` distinguished name entries that correspond to services that the user is allowed to use. For example, if the user has an FTP directory, there may be an entry of the form "ispversion=1.0, ou=SUNWftp, ou=Services, o=myisp, c=us."

UID (Optional) Contains a UNIX user id number.

GID (Optional) Contains a UNIX group id number.

ContentDirectory (Optional) Identifies the directory used by the user for personal content (FTP uploads and personal web documents).

sisprad Input

Each line in the input file for the `sisprad` table has the following syntax:

```
!LastName!FirstName!Nickname!Username!Password!DomainName!\
E-mailAddress!Services!UID!GID!ContentDirectory!\
AuthSuffix!GroupCheck!GroupReply!
```

Where the fields from "Last Name" through "Content Directory" are the same as in the "sisp Input" section, and

AuthSuffix (Optional) Defines a string that may be appended to the user name used for authentication when the RADIUS server processes the user for authentication. For example, the RADIUS server may append a domain string to authenticate users who belong to a particular domain. For example, a user logging in as "jdoe" might be authenticated as "jdoe@othernet.com."

GroupCheck (Optional) Defines a list of attributes in a user's record that the RADIUS server will check against data supplied during authentication. If no GroupCheck information is supplied, RADIUS will check all of the remote user's attributes before granting access. An example attribute for GroupCheck is "userPassword." Separate multiple attributes by spaces in the input file.

GroupReply (Optional) Defines a list of attributes returned by the RADIUS server with an access-accept or access-reject response. Example attributes that might be used in the GroupReply field are "radiusPppProfile" and "dynamicIPAddress." Separate multiple attributes by spaces in the input file.

sispsims Input

Each line in the input file for the `sispsims` table has the following syntax:

```
!LastName!FirstName!Nickname!Username!Password!DomainName!\
E-mailAddress!Services!UID!GID!ContentDirectory!\
AuthSuffix!GroupCheck!GroupReply!\
DataSource!MailServe!MailDomain!LegacyMail!\
MailDeliveryOption!RFC822Mailbox!MailFolderMap!MailQuota!\
ChannelType!
```

Where the fields from "Last Name" through "Content Directory" are the same as in the "sisp Input" section, and

DataSource Contains a string identifying the original source of this data for reference.

MailServer Defines the fully qualified domain name of the mail server to which mail for the user should be delivered.

LegacyMail Defines an object class for users that use a legacy mail gateway channel. Valid object classes supported by Sun Internet Mail Server are `gatewayCCMailUser`, `gatewayMSMailUser`, and `gatewayProfsUser`.

MailDeliveryOption Specifies the value of the `mailDeliveryOption` attribute for the user. The variable `MAILDEV` in the `Common` section will be used if there is no input in this field.

RFC822Mailbox Specifies all of the email addresses defined for the user. This field may be left blank if the value of

EmailAddress is the only valid address for the user.

MailFolderMap — Defines the message store for the user's mail folder. This field may be "`UNIX V7`" to use `/var/mail` or "`Sun-MS`" to use the Sun Internet Mail Server message store. The variable `SUNMS` in the `Common` section will be used if there is no input in this field.

MailQuota — Defines the maximum size (in bytes) of the message store for the user.

ChannelType — Defines the type of legacy mail channel for the user. This is an optional field for users that use a legacy mail gateway channel; valid data in this field are:

- 0 for CC:mail
- 1 for Microsoft Mail
- 4 for an SMTP mail system
- 8 for IBM PROFS

sispradsims Input

Each line in the input file for the `sispsims` table has the following syntax:

```
!LastName!FirstName!Nickname!Username!Password!DomainName!\
EmailAddress!Services!UID!GID!ContentDirectory!\
AuthSuffix!GroupCheck!GroupReply!\
DataSource!MailServer!MailDomain!LegacyMail!\
MailDeliveryOption!RFC822Mailbox!MailFolder!MailQuota!\
ChannelType!
```

The `sispradsims` input line is composed of all of the fields from the other three tables, which are described above.

Mapping File Syntax

The default mapping file is located in `/etc/opt/SUNWisp/ldap/sunds/default/mapping/sisploadmapping`.

The mapping file contains the rules for parsing the input file lines, assigning the fields to variables, and then using those variables to construct an ldif-format record for the directory server. In most cases, you will not need to edit this file. If you have a non-default input file format or if you have extended the default Solaris ISP Server schema, you may need to make changes to the mapping file. Please read this document and the man page for **`dsimport`**(1m) before editing the mapping file.

Note - The syntax of the mapping file is complicated. It may be very difficult to create a mapping file that properly converts data into records for your directory. Editing the mapping file is not recommended.

The mapping file contains four `Table` sections which define how input will be parsed and converted to `ldif` records. The four tables are `sisp`, `sisprad`, `sispsims`, and `sispradsims`.

See the Input File Syntax section for details on each table's function.

See the **`dsimport`**(1m) manual page for more detailed information on the contents of each table and the general rules for constructing a mapping.

The following token and variable definitions may help you adjust the mapping if you have trouble with the output or with creating valid input:

`BASE_DN` — Defines the distinguished name (dn) in the directory naming context where user records will be created. The default is the base dn specified when Solaris ISP Server was installed. The dn of each user will be `cn=`*FirstName LastName (Username)*`,ou="People", ou=`*domain* `[ou=`*domain*`...], BASE_DN`.

`DIRPREF` — Defines a path prefix to all user ISP content directories. The default is `/home`. The value of the content directory field from the input file is appended to this value to get the full path for the `ispContentDirectory` attribute. The `ispDirectoryRoot` attribute for an `ispService` plus the `ispContentDirectory` forms the full path to the actual directory.

`MATCH_FILTER` — Defines the criteria for determining whether a record already exists in the directory. By default, a new record from the input data is considered a duplicate if there is already a record with a matching `cn=`*FirstName LastName* and `mail=`*EmailAddress*. Refer to **`dsimport`**(1m) for details on constructing a match filter if the default does not meet your needs.

`SEPARATOR` — Defines the field delimiter in the input data. If you change the `SEPARATOR` you must also

change the delimiter used in the `LINE` definitions in the table's `Dynamic` and `Extract` sections.

Output

The output from `sispload` is stored in two files: *filename*`.err` contains error messages and *filename*`.ldif` contains ldif records for adding records to the directory.

The majority of the ldif file will consist of user records to add or modify. The format of the records depends on the translation table used. The user records for each table are explained below.

In addition to user records, there may be entries to create intermediate nodes where user data is stored. The ldif file may contain records that add `organizationalUnit` object classes to fill in the tree from the base DN to the `ou=People` node where users are stored.

sisp Output

The object class of user records when the `sisp` table is used is `ispSubscriber`.

The following list describes how the input data is used to construct attributes in the ldif user record. Refer to the "Input" section for information on the meaning of the input fields (shown in a *different style*).

`dn`

Specifies the complete distinguished name for the user record. The `dn` is constructed as follows:

`dn=cn="`*FirstName LastName (Username)*`", ou="People", ou=`*DomainName*`[ou=`*domain*`...], $BASE_DN`.

`cn`

Specifies an additional `commonName` attribute.

`cn=`*FirstName LastName*.

`sn`

Specifies the family name or surname.

`sn=`*LastName*.

`givenname`

Specifies the first or given name.

`givenname=`*FirstName*.

`userid`

Specifies the user name used for authentication.

`userid=`*Username*.

`userPassword`

Specifies the password used for authentication. If no password is given in the input data, the user name is used.

`userid=`*Password* `||` *Username*.

`uidNumber`

Specifies the user ID number used for processes and files owned by the user.

`uidNumber=`*UID*.

`gidNumber`

Specifies the group ID number used for processes and files owned by the user.

`gidNumber=`*GID*.

`ispContentDirectory`

Specifies a path prefix for the actual file system directory where the user stores personal content (such as web pages or FTP content). The `ispContentDirectory` attribute is built from appending the field from the input file to the value of `DIRPREF` in the mapping file `Common` section. If `DIRPREF` is null and there is no data in the input field, this attribute is not set.

`ispContentDirectory= $DIRPREF/`*ContentDirectory*.

`ispAuthorizedServices`

Specifies the distinguished name of each ISP service this user is allowed to use. Each distinguished name in the input file should reference the

`ispVersion` attribute of an existing `ispService` record. Multiple services can be separated by spaces between the delimiters in the input file. An example of an entry in the input file is "ispversion=1.0,ou=SUNWftp,ou=Services,o=myisp,c=us."

`ispAuthorizedServices=`*Services*.

`mail`

Specifies the full email address.

`mail=`*EmailAddress*.

sisprad Additional Output

The object classes of user records when the `sisprad` table is used are `remoteUser` and `ispSubscriber`.

The ldif record contains all of the attributes described in the sisp Output section. In addition, the `sisprad` table generates the additional attributes for a `remoteUser` object listed below. Refer to the "Input" section for information on the meaning of the input fields (shown in a *different style*).

`authsuffixname`

Specifies the `authSuffixName` attribute. This string is added to the user name by the RADIUS server when it checks authentication. If there is no value in the input file, the user's *DomainName* is used.

`authSuffixName=`*AuthSuffix* `||`*DomainName*.

`grpCheckInfo`

Specifies a list of attributes that the RADIUS server checks against the data supplied by a user session at login. If there is no value in the input file, the attributes `authSuffixName` and `userPassword` are used.

`grpCheckInfo=`*GroupCheck* `|| "authSuffixName userPassword"`.

`authServiceProtocol`

Specifies the type of service for the user. This attribute takes its value from the `AUTHPROT` variable defined in the `Common` section; the default value is `Framed-User`.

`authServiceProtocol=$AUTHPROT`.

`framedrouting`

Specifies the routing method for the user, when the user is a router to a network. This attribute takes its value from the FRAMEDROUTING variable defined in the Common section; the default value is None.

framedrouting=$FRAMEDROUTING.

framedprotocol

Specifies the framing to be used for framed access. This attribute takes its value from the FRAMEDPROTOCOL variable defined in the Common section; the default value is PPP.

framedprotocol=$FRAMEDPROTOCOL.

grpReplyInfo

Specifies a list of attributes returned by the RADIUS server with an access-accept or an access-reject response. If there is no value in the input file, the attributes authServiceProtocol, framedProtocol, and framedRouting are used.

grpReplyInfo=*GroupReply* ||
"authServiceProtocol framedProtocol framedRouting".

sispsims and sispradsims Additional Output

The object classes of user records when the sispsims table is used are emailPerson and ispSubscriber. If there is an input entry for *LegacyMail*, such as "gatewayCCMailUser," then this additional object class is also created for the user. When the sispradsims table is used, a remoteUser object class is created for each user record.

The ldif records from the sispsims table contain all of the attributes described in the "sisp Output" section. Records from the sispradsims table also include the records described in the "sisprad Output" section.

In addition, the sispsims and sispradsims tables generate the additional attributes for an emailPerson object listed below. Refer to the "Input" section for information on the meaning of the input fields (shown in a *different style*).

dataSource

Specifies the original source of the user data. This is an optional field for reference purpose. If the data source is blank in the input file, the string "Solaris for ISP Server(tm) sisploads utility" is used.

dataSource=*DataSource* ||
"Solaris for ISP Server(tm) sisploads utility"

`mailhost`

Specifies the fully qualified domain name of the SMTP/MIME mail server.

`mailhost=`*MailServer*

`mailQuota`

Specifies the maximum size in bytes of the user's message store, or 0 (zero) for unlimited size.

`mailQuota=`*MailQuota*

`mailDeliveryOption`

Specifies the method for delivering mail to the user. If there is no value in the input file, the attribute takes its value from the `MAILDELIV` variable defined in the `Common` section; the default value is `mailbox`. See "Appendix D. SIMS Directory Schema and Directory Information Tree " in *Sun Internet Mail Server 3.5 Administrator's Guide* for a complete list of valid values.

`mailDeliveryOption=`*MailDeliveryOption* `|| $MAILDELIV`

`mailFolderMap`

Specifies the type of message store for the user's mail folders. If there is no value in the input file, the attribute takes its value from the `SUNMS` variable defined in the `Common` section; the default value is `"SUN-MS"`.

`mailFolderMap=`*MailDeliveryOption* `|| $SUNMS`.

`preferredrfc822recipient`

Specifies the user's internal email address. This attribute is built by combining the *Username* and *MailServer* tokens from the input file.

`preferredrfc822recipient=`*UserName@MailServer*.

`rfc822Mailbox`

Specifies an email address for the user. There may be multiple valid addresses separated by spaces in the input file; a separate `rfc822Mailbox` attribute is added for each address.

`rfc822Mailbox=`*RFC822Mailbox*.

`channelType`

Defines the type of legacy mail channel for the user.

`channelType=`*ChannelType*.

EXAMPLES

EXAMPLE 1

The following listing shows an example input file with two user records. The records will be converted using the `sisprad` table.

```
!Smith!Joseph!Joe!jsmith!!myisp!\
jsmith@myisp.net!ispversion=1.0,ou=SUNWftp,ou=Services,o=MyISP,c=US!\
89091!10!jsmith!@myisp.net!authSuffixName userPassword!\
authServiceProtocol framedProtocol framedRouting\
radiusPppProfile!
!Jones!Janet!!janjones!!myisp!\
janjones@myisp.net!ispversion=1.0,ou=SUNWftp,ou=Services,o=MyISP,c=US!\
89092!10!janjones!@myisp.net!authSuffixName userPassword!\
authServiceProtocol framedProtocol framedRouting\
radiusPppProfile!
```

If the above data is in a file named `/tmp/radius.input` it could be converted to ldif records using `sispload`:

```
% cd /opt/SUNWisp/ldap/sunds/sbin
% ./sispload -b o=MyISP,c=US -t sisprad -f /tmp/radius
```

The file `radius.ldif` should contain the following records if the translation is successful and the user records do not already exist:

```
dn: cn=Joseph Smith (jsmith),ou=People,ou=myisp,o=MyISP,c=US
changetype: add
sn: Smith
cn: Joseph Smith
userid: jsmith
userPassword: jsmith
objectClass: top
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: ispSubscriber
objectClass: remoteUser
uidNumber: 89091
gidNumber: 10
ispContentDirectory: home/jsmith
ispauthorizedServices: ispversion=1.0,ou=SUNWftp,\
```

```
ou=Services,o=MyISP,c=US
mail: jsmith@myisp.net
authsuffixname: @myisp.net
grpCheckInfo: authSuffixName
grpCheckInfo: userPassword
authServiceProtocol: Framed-User
framedrouting: None
framedprotocol: PPP
grpReplyInfo: authServiceProtocol
grpReplyInfo: framedProtocol
grpReplyInfo: framedRouting
grpReplyInfo: radiusPppProfile
dn: cn=Janet Jones (janjones),ou=People,ou=myisp,o=MyISP,c=US
changetype: add
sn: Jones
cn: Janet Jones
userid: janjones
userPassword: janjones
objectClass: top
objectClass: person
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: ispSubscriber
objectClass: remoteUser
uidNumber: 89092
gidNumber: 10
ispContentDirectory: home/janjones
ispauthorizedServices: ispversion=1.0,ou=SUNWftp,\
ou=Services,o=MyISP,c=US
mail: janjones@myisp.net
authsuffixname: @myisp.net
grpCheckInfo: authSuffixName
grpCheckInfo: userPassword
authServiceProtocol: Framed-User
framedrouting: None
framedprotocol: PPP
grpReplyInfo: authServiceProtocol
grpReplyInfo: framedProtocol
grpReplyInfo: framedRouting
grpReplyInfo: radiusPppProfile
```

Finally, these records can be added to the directory using **ldapmodify**(1m):

```
% cd /opt/SUNWconn/bin
% ./ldapmodify -D cn=admin,o=MyISP,c=US -w secret -f /tmp/radius.ldif
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixsds
Interface Stability	Evolving

SEE ALSO

sisploadmapping(1m), **dsimport**(1m)

Sun Directory Services 3.1 Administration Guide

Sun Internet Mail Server 3.5 Administrator's Guide

PART II Sun Internet FTP Server

Sun™ Internet FTP Server™ 1.1 man pages.

CHAPTER 1

FTP Command-Line Procedures

This section provides the Sun™ Internet FTP Server™ command line procedures for configuration. subscriber authentication, and maintenance.

1.1 FTP Configuration

FTP configuration is comprised of the following steps:

- Create the anonymous FTP directory. This directory contains the subdirectories and binaries required for FTP support, and is cloned to create each virtual host's root FTP directory.
- Create the virtual host directory structure: This contains a cloned tree of the anonymous FTP directory as well as the FTP access file for the virtual host.

This section assumes:

- Solaris ISP Services installation has been completed
- FTP has been registered with the Sun Internet Admistrator (see the online help for the *Sun Internet Administrator Register Services Screen*)
- Sun Internet FTP Server is installed, but not configured
- The virtual host has not been created
- The directories `/usr/sbin` and `/usr/lib` are in your root $PATH.

The examples used assume:

- The ISP anonymous FTP directory will be created as `/opt/IspFtpDir.` This is an example only; you need to determine the actual name and location
- The FTP virtual host is `myVH.org.` This is an example only; you need to provide the actual virtual host name
- The FTP virtual host root directory will be created in `/export/home`

1.1.1 Procedure

1. Use `ftpconfig(1m)` to create the anonymous FTP directory:

   `ftpconfig -d /opt/IspFtpDir`

   This creates the following in the specified directory:
   `bin/, dev/, etc/, pub/, usr/, var/,` and `Welcome.` The directory `bin/` is a symbolic link to `/usr/bin,` and `Welcome` is the welcome message displayed to the FTP user on successful login.

Note - The `ftpconfig(1m)` command only needs to be run once regardless of the number of virtual hosts you need to create. Ensure you do not create the ISP anonymous FTP directory in `/tmp`.

---

2. Use `ftpaddhost(1m)` to create the virtual host directory:

   `ftpaddhost /opt/IspFtpDir /export/home/ myVH.org`

   This creates the directory `/export/home/myVH.org`. The contents of the anonymous FTP directory `/opt/IspFtpDir` are cloned to create the required file hierarchy within `/export/home/myVH.org`:
   `dev/, etc/, pub/, usr/, var/,` and the file `Welcome` .

   The FTP access configuration file `/etc/inet/ftpaccess` is copied to the virtual host `etc/` directory (`/etc/inet/`*hostname*`/`). Thus, each virtual host you define using `ftpaddhost(1m)` inherits the configuration defined in `/etc/inet/ftpaccess.`

3. Create the subscriber (real user) account in `/etc/passwd`. See `passwd(4)` and `admintool(1M)` for further information. Make note of the user ID associated with the user login name.
4. Edit `/etc/group` and enter a unique group ID for the subscriber (real user) account in the file, then add the user login name to the newly created group. See `group(4)` for further information.

1.2 Subscriber Authentication

The following procedures for configures the virtual host for LDAP authentication.

1.2.1 LDAP

1. Determine the UID and GID you will assign to the FTP subscriber and subscriber directories.
2. Create an LDIF file defining the virtual host type (org, net, com, edu, and so on), virtual host name, country code, subscriber directory, UID, and GID.

   You can use the following example as a template, replacing:

   - *country_code* with the two-letter code for your country, for example `us`, `de`, `ca`, and so on.
   - *virtual_host_type* with the virtual host type.

     For example if the virtual host is accessed via *myVH.org*, you would replace *virtual_host__type* with `org`
   - *Your_ISP_Name* with the name of your ISP.
   - *Virtual_host_Name* with the name of the virtual host.

     For example if the virtual host is accessed via *myVH.org*, you would replace *virtual_host__name* with `myVH`.
   - *virtual_host_dir* with the full path name to the virtual host root directory.

     For example if the virtual host *myVH.org* is located in `/export/home/myVH.org`, you would replace *virtual_host__dir* with `/export/home/myVH.org`.

```
dn: dc=virtual_host_type
dc: virtual_host_type
objectclass: domain

dn: ou=virtual_host_name,o=Your_ISP_Name,c=country_code
ou: virtual_host_name
associateddomain: virtual_host_name.virtual_host_type
objectclass: organizationalUnit
objectclass: domainRelatedObject
objectclass: top

dn: dc=virtual_host_name,dc=virtual_host_type
```

(continued)

(Continuation)

```
dc: virtual_host_name
objectclass: domain
objectclass: labeledURIObject
associatedname: ou=virtual_host_name,o=Your_ISP_Name,c=country_code
description: DNS to DN mapping for virtual_host_name.virtual_host_type
labeleduri: ldap:///ou=virtual_host_name,o=Your_ISP_Name,c=country_code??sub

dn: ou=Services,ou=virtual_host_name,o=Your_ISP_Name,c=country_code
ou: Services
objectclass: organizationalUnit

dn: ou=Groups,ou=virtual_host_name,o=Your_ISP_Name,c=country_code
ou: Groups
objectclass: organizationalUnit

dn: ou=People,ou=virtual_host_name,o=Your_ISP_Name,c=country_code
ou: People
objectclass: organizationalUnit

dn: ou=SUNWftp,ou=Services,ou=virtual_host_name,o=Your_ISP_Name,c=country_code
ou: SUNWftp
objectclass: organizationalUnit

dn: ispversion=1.0,ou=SUNWftp,ou=Services,ou=virtual_host_name,o=Your_ISP_Name,c=country_code
ispversion: 1.0
cn: SUNWftp
objectclass: ispservice
ispdirectoryroot:virtual_host_dir
dn: cn=ftp,ou=People,ou=virtual_host_name,o=Your_ISP_Name,c=country_code
commonname: ftp
uid: ftp
sn: ftp
userpassword: ftp
objectclass: ispSubscriber
uidnumber: 60001
gidnumber: 70001
ispcontentdirectory: virtual_host_dir
```

Caution - Before saving the file, ensure there are no trailing blanks. If there are any trailing blanks, `ldapadd` will report a syntax error.

3. Save the file as *virtual_host_name*`.ldif`, for example, `myVH.ldif`
4. The virtual host and subscriber information is added using the `ldappadd` command syntax:

   `# ldapadd -v -D"`*`bindDN`*`" -w` ***`bindPassword`*** **`-f`** ***`filename`*****`.ldif`**

CODE EXAMPLE 1–1

Assume:

- You are logged in as root on the computer where Sun Directory Services and the Solaris FTP server are installed.
- The password of the Directory Service administrator is `ftp555`.
- The country code is `us`.
- The Directory Service administrator distinguished name is "`cn=admin,o=intra,c=us`".
- Your corporate domain name is `intra.net`.
- You have created and saved the LDAP configuration file as `myVH.ldif`.

The command to add the virtual host and subscriber information to LDAP would then be:

```
# ldapadd -v -D"cn=admin,o=intra,c=us" -w ftp555 -f myVH.ldif
```

1.3 Maintenance

1.3.1 Start `ftpd`

The standard FTP installation places an entry in `/etc/inetd.conf`, thereby automatically starting the FTP server when an FTP connection is made.

The `ftpshut(1m)` command creates the file `/var/ftp/shutdown` which disables FTP.

To reenable FTP, enter the command:

```
ftpshut enable
```

This deletes the `/var/ftp/shutdown` file, allowing subsequent FTP connections to automatically start the FTP daemon.

1.3.2 Stop `ftpd`

The **`ftpshut`**(1m) command is used to shut down the FTP server. You can optionally specify the number of minutes to new user lockout and disconnect of

existing users as well as the logout message, or you can specify a configuration file containing this information.

The **ftpshut**(1m) command creates the file/var/ftp/shutdown. This blocks the restart of the FTP server until you enter the command `ftpshut enable`.

Examples:

- Shut down FTP immediately:

  ftpshut now

- Shut down FTP in 15 minutes, deny access to new users in 5 minutes, disconnect users not in file transfer mode in 10 minutes:

  root# ftpshut -l 5 -d 10 15 System going down in 15 minutes

---

Note - The shutdown message is limited to 76 characters maximum.

Please refer to the **ftpshut**(1m) man page for information on creating and using a shutdown configuration file.

1.3.3 Show Users by Class

Refer to **ftpaccess**(4) for the definition of `class` and procedures for defining new classes.

Use the **ftpcount**(1m) to display the number of active users per class:

/usr/sbin/ftpcount

The number of users per class and the class maximums are displayed:

```
root# ftpcount
Service class anon                -   2 users ( 10 maximum)
Service class guest               -   0 users ( 10 maximum)
Service class real                -   0 users
root#
```

1.3.4 Delete an FTP Site

Removal of an FTP virtual host depends on the type of subscriber authentication; and whether or not the virtual host is also serving as a web hosting site.

The following procedure uses the examples:

- The FTP virtual host is `myVH.org`.
- the FTP virtual host chroot directory is`/export/home/myVH.org`.

1. Remove the FTP virtual host entry from `/etc/inet/ftpservers`. For example, you would delete the line **`myVH.org /etc/inet/myVH.org/ftpaccess`**.
2. If this is an FTP-only virtual host:
   a. Remove the FTP virtual host entry from `/etc/inet/hosts`. For example, you would delete the line containing **`myVH.org`** from `/etc/inet/hosts`.
   b. Remove the virtual host chroot directory; in this example, `/export/home/myVH.org`.
3. If this is an FTP / web site host:
   a. Change directory to the virtual host chroot directory, in this example `/export/home/myVH.org`.
   b. Remove the file `Welcome`, and remove the following directories *only* if they do not contain web data:`dev/`, `etc/`, `pub/`, `usr/`, and `var/`.
4. Remove the FTP virtual host entries from LDAP using Deja, or via the command line as described by **`ldapdelete`**(1m).

CHAPTER 1

man Pages(1m): Maintenance Commands

NAME

ftpintro.1m – introduction to the host configuration software command-line utilities for the Sun Internet FTP Server

DESCRIPTION

The man pages offer detailed instruction and examples on options and subcommands for each utility. The command-line utilities are available to start and run the host configuration tool that installs Solaris for ISPs components and configures the system.

LIST OF COMMANDS

ftpaddhost(1m)	The ftpaddhost command provides an automated procedure to set up a virtual anonymous FTP server.
ftpconfig(1m)	The ftpconfig command provides an automated procedure to set up anonymous FTP.
ftpcount(1m)	The `ftpcount` command shows the current number of users logged on and the login limit for each class defined in the **ftpaccess**(4) file.
ftpshut(1m)	The `ftpshut` command provides an automated shutdown procedure that a superuser can use to notify ftp users when the ftp server is shutting down.
in.ftpd(1m)	`in.ftpd` is the Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the ‘‘ftp’’ service specification. To deny login for a particular user, add the user’s login to the `/etc/inet/ftpusers` file.

EXIT STATUS

Upon termination, each command returns the following exit values:

`0`	Successful completion.
`>0`	An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu

SEE ALSO

ftpIntro(4)

NAME

ftpaddhost – add a virtual anonymous FTP server

SYNOPSIS

ftpaddhost *template_dir root_dir hostname* [*anon_dir*]

DESCRIPTION

The `ftpaddhost` command provides an automated procedure to set up a virtual anonymous FTP server. It uses the anonymous FTP area in *template_dir* to create a cloned tree in a new directory under *root_dir*. The cloned tree contains hard links to all the system files in *template_dir*, so they must both reside in the same file system. The *template_dir* can be the anonymous file area created by the **`ftpconfig`**(1M) command. The virtual server to which service is provided is identified by its *hostname*. The *hostname* also determines the name of the directory created under *root_dir*. If the *hostname* is `ftp.corp.com`, then the anonymous area created is in `virtual_dir/ftp.corp.com`. The command copies the file `Welcome` from the *template_dir* to the file area for the virtual server, and adds an entry for the new *hostname* to the **`ftpservers`**(4) file.

If `ftpaddhost` is used in conjunction with an LDAP user connection, the *hostname* must be a fully qualified domain name.

OPERANDS

The following operands are supported:

template_dir An existing anonymous FTP setup directory.

root_dir The top directory for the virtual host.

hostname The host name of the virtual server. $root_dir/$homename will be the root directory for this virtual host. An IP address can be used for hostname; however, if the LDAP server is being used for authentication, the *hostname* must be the fully qualified domain name.

anon_dir The anonymous user's home directory on the virtual host.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

FILES

`/etc/inet/ftp.$hostname.conf` The configuration file for the virtual host.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu
Interface Stability	Evolving

SEE ALSO **chroot**(1M), **ftpconfig**(1M), **ftpd**(1M), **attributes**(5), **ftpaccess**(4)

NAME

ftpconfig – configure anonymous FTP

SYNOPSIS

ftpconfig [–d] [–u] [*ftpdir*]

DESCRIPTION

The `ftpconfig` command provides an automated procedure to set up anonymous File Transfer Protocol (“FTP”).

Anonymous FTP allows users to remotely log on the FTP server by specifying the user name "ftp" or "anonymous" and the user’s email address as password. The anonymous users are logged on to the server and given access to a restricted file area with its own file system root. See **`chroot`**(1). The FTP area has its own minimal system files.

The `ftpconfig` command will copy and set up all the components needed to operate an anonymous FTP server, including creating the FTP user account, creating device nodes, copying `usr/lib` files, copying timezone data, and configuring `etc/nsswitch.conf` and `etc/pam.conf`. The `passwd` and `group` files set up contain no real user names, to prevent malicious users from finding login names on the server. The anonymous file area is placed in `ftpdir`. If the FTP user account already exists, then the current `FTP` area is used, and the system files in it are updated. All other files are left untouched. The `ftpconfig` command should be run to update the anonymous FTP area’s configuration whenever a system patch is installed, or the system is upgraded.

If the –`d` option is used, `ftpconfig` creates only an anonymous FTP directory, without adding or updating the FTP user account. This option is useful for creating template directories that can be customized later and used with **`ftpaddhost`**(1M) to create virtual servers.

If the –`u` option is specified, then `ftpconfig` will only perform an update. If an update is not possible, the command will print an error and exit. The *ftpdir* argument should be omitted with the –`u` option.

The anonymous login name is always "ftp".

OPTIONS

–`d`	Create anonymous FTP directory only.
–`u`	Update configuration.

OPERANDS

The following operands are supported:

ftpdir	The directory in which to create an anonymous FTP.

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu
Interface Stability	Evolving

SEE ALSO

chroot(1), **ftpaddhost**(1M), **ftpd**(1M), **attributes**(5)

NAME

ftpcount – show current number of users for each class

SYNOPSIS

ftpcount

DESCRIPTION

The `ftpcount` command shows the current number of users logged on and the login limit for each class defined in the **ftpaccess**(4) file.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu
Interface Stability	Evolving

SEE ALSO

ftpd(1M), **attributes**(5), **ftpaccess**(4)

NAME

ftpshut – shut down the ftp servers at a given time

SYNOPSIS

ftpshut [–F *config_file*] [–l *min*] [–d *min*] *time* [*warning-message*...]

DESCRIPTION

The `ftpshut` command provides an automated shutdown procedure that a superuser can use to notify `ftp` users when the `ftp` server is shutting down. The *time* is the time at which to bring the `ftp` server down. It may be the word 'now,' indicating an immediate shutdown, or specify a future time in one of two formats: + number or HHMM. The first form brings the `ftp` servers down in number minutes. The second brings the `ftp` servers down at the time of day indicated, using a 24-hour clock format.

Ten minutes before shutdown, or immediately if *time* is less than ten minutes, any new `ftp` access will be disabled. This time may be adjusted through the –`l` flag. Five minutes before shutdown, or immediately if is timed for less than five minutes, all current `ftp` connections will be disconnected.This time may be adjusted through the –`d` flag. The *warning-message* will be formatted to be 75 characters wide. The following format controls can be embedded in *warning-message*:

%s Time system is going to shut down.

%r Time new connections will be denied.

%d Time current connections will be dropped.

%C Current working directory.

%E The administrator's email address.

%F Free space in partition of CWD, in kilobytes.

%L Local host name.

%M Maximum allowed number of users in the class to which a login was determined to belong. See the CLASS configuration keyword in **`ftpaccess`**(4).

%N Current number of users in the class to which a login was determined to belong. See the CLASS configuration keyword in **`ftpaccess`**(4).

%R Remote host name.

%T Local time (form Thu Nov 15 17:12:42 1990).

%U Username given at login time.

OPTIONS

–F ***config_file*** The name of a file containing access control definitions following the same format of the ftpaccess configuration file.

This enables specification of alternate hosts when there are virtual hosts defined on a server.

`-l` ***min*** The time ahead of shutdown, in minutes, that new connections will be refused.

`-d` ***min*** The time ahead of shutdow, in minutes, that existing connections not in file transfer will be disconnected.

OPERANDS

The following operands are supported:

time The time at which to bring the FTP server down

warning-message Message to display that warns of the imminent shutdown.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu
Interface Stability	Evolving

SEE ALSO

`ftpd`(1M), **`shutdown`**(1M), **`attributes`**(5), **`ftpaccess`**(4)

NAME

in.ftpd, ftpd – Internet File Transfer Protocol server

SYNOPSIS

in.ftpd [–d] [–l] [–t *timeout*] [–T *maxtimeout*] [–u *mask*] [–r] [–s] [–w *timeout*] [–L] [–i] [–o]

DESCRIPTION

`in.ftpd` is the Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the ‘‘ftp’’ service specification. See **`services`**(4).

To deny login for a particular user, add the user's login to the `/etc/inet/ftpusers` file.

`ftpaccess`(4) is the configuration file. For backward compatibility, umask and banner can be set as follows. The umask, which is used to create files during `PUT` operations, may be set by adding the following line to `/etc/default/ftpd`:

```
UMASK=nnn
```

The banner returned by `in.ftpd` in the parenthetical portion of its greeting is configurable. The default is equivalent to '`uname`–`sr`' and will be used if no banner is sent in `/etc/default/ftpd`. To set the banner, add a line to `/etc/default/ftpd` of the form:

```
BANNER="..."
```

Non-empty banner strings are fed to shells for evaluation. The default banner may be obtained by:

```
BANNER="‘uname -s‘ ‘uname -r‘"
```

No banner will be printed if `/etc/default/ftpd` contains:

```
BANNER=""
```

The FTP server currently supports the following `ftp` requests; case is not distinguished.

REQUEST	DESCRIPTION
ABOR	Abort previous command.
ACCT	Specify account (ignored) .
ALLO	Allocate storage (vacuously) .
APPE	Append to a file.
CDUP	Change to parent of current working directory.

CWD	Change working directory.
DELE	Delete a file.
HELP	Give help information.
LIST	Give list of files in a directory (‘‘`ls -la`’’).
MKD	Make a directory.
MDTM	Show last modification time of file.
MODE	Specify data transfer mode.
NLST	Give name list of files in directory.
NOOP	Do nothing.
PASS	Specify password.
PASV	Prepare for server-to-server transfer.
PORT	Specify data connection port.
PWD	Print the current working directory.
QUIT	Terminate session.
REST	Restart incomplete transfer
RETR	Retrieve a file.
RMD	Remove a directory.
RNFR	Specify rename-from file name.
RNTO	Specify rename-to file name.
SITE	Non-standard commands (see next section).
SIZE	Return size of file.
STAT	Return status of server.
STOR	Store a file.
STOU	Store a file with a unique name.
STRU	Specify data transfer structure.
SYST	Show operating system type of server system.

TYPE	Specify data transfer type.
USER	Specify user name.
XCUP	Change to parent of current working directory.
XCWD	Change working directory (deprecated).
XMKD	Make a directory (deprecated).
XPWD	print the current working directory (deprecated).
XRMD	remove a directory (deprecated).

The following non-standard or UNIX specific commands are supported by the SITE request:

REQUEST	DESCRIPTION
UMASK	Change umask, for example, **SITE UMASK 002**.
IDLE	Set idle-time, for example, **SITE IDLE 60**.
CHMOD	Change mode of a file, for example, **SITE CHMOD 755** ***filename***.
HELP	Give help information, for example, **SITE HELP**.
NEWER	List files newer than a particular date.
MINFO	Like **SITE NEWER**, but gives extra information.
GROUP	Request special group access, for example, **SITE GROUP** ***foo***.
GPASS	Give special group access password, for example, **SITE GPASS** ***bar***.
EXEC	Execute a program, for example, **SITE EXEC** ***program params***.

The remaining ftp requests specified in *Internet RFC 959* are recognized but not implemented. MDTM and SIZE are not specified in *Internet RFC 959*, but they will appear in a future protocol version.

The FTP server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in *Internet RFC 959*. If a STAT command is received during a data transfer, preceded by a Telnet IP and Synch, transfer status will be returned.

`in.ftpd` interprets file names according to the ‘‘globbing’’ conventions used by **`csh`**(1). This allows the use of the metacharacters ‘‘`*?[]{}~`’’ in file names.

`in.ftpd` authenticates users according to four rules.

1. The user name must be in the password data base, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.
2. The user name must not appear in the file `/etc/inet/ftpusers`.
3. The user must have a standard shell returned by **`getusershell`**(3C).
4. If the user name is ‘‘anonymous’’ or ‘‘ftp’’, an anonymous `ftp` account must be present in the password file (user ‘‘ftp’’). In this case the user is allowed to log in by specifying any password (by convention this is given as the client host’s name).

In the last case, `in.ftpd` takes special measures to restrict the client’s access privileges. The server performs a **`chroot`**(2) command to the home directory of the ‘‘ftp’’ user. In order that system security not be breached, it is recommended that the ‘‘ftp’’ subtree be constructed with care; the following rules are recommended. Note that the **`ftpconfig`**(1M) script will set these up automatically.

`/home/ftp`	Make the home directory owned by superuser and unwritable by anyone.
`/home/ftp/bin`	Make this directory owned by the superuser and unwritable by anyone. The program **`ls`**(1) must be present to support the list command. This program should have mode 111.
`/home/ftp/etc`	Make this directory owned by the superuser and unwritable by anyone. The files **`passwd`**(5) and **`group`**(5) must be present for the `ls` command to be able to produce owner names rather than numbers. The password field in `passwd` is not used, and should not contain real encrypted passwords. These files should be mode 444 and owned by the superuser. Do not use th system’s `/etc/passwd` file as the password file or the system’s `/etc/group` file as the group file in the `/home/ftp/etc` directory.
`/home/ftp/pub`	Create a subdirectory in `/home/ftp/pub` with the appropriate mode (777 or 733) if you want to allow normal users to upload files.

General FTP Extensions

The FTP server has functionality to allow a client to request automatic file type conversion. See **ftpconversions**(4). If the client asks for Specified Filename below, it gets True Filename with Action performed on it. For instance, if there is a directory name "src," the client can download "src.tar;" it gets the "src" directory with the action "tar" performed. In short, it gets a tar file of the directory.

True Filename	Specified Filename	Action
<filename>.Z	<filename>	`uncompress` file before transmitting.
<filename>	<filename>.Z	`compress` <filename> before transmitting.
<filename>	<filename>.tar	`tar` <filename> before transmitting.
<filename>	<filename>.tar.Z	`tar` and `compress` <filename> before transmitting

The FTP server will check passwords supplied with anonymous logins for valid e-mail addresses and produce a chiding message if the password does not pass the test. A dash as the first character of the password can be used to disable multiline message in anonymous logins, and it is useful with a few older clients that get confused by these.

The FTP server can also log all file transmission and reception, keeping the following information for each file transmission that takes place:

```
Mon Dec 3 18:52:41 1990 1 wuarchive.wustl.edu 568881
  /files.lst.Z a _ o a chris@wugate.wustl.edu ftp 0 *
  %.24s %d %s %d %s %c %s %c %c %s %s %d %s
  1 2 3 4 5 6 7 8 9 10 11 12 13
```

Field Number	Description
1	Current time in the form `DDD MMM dd hh:mm:ss YYYY`.
2	Transfer time in seconds.
3	Remote host name.
4	File size in bytes.
5	Name of file.
6	Transfer type (a>scii, b>inary).

7	Special action flags (concatenated as needed):
	C File was compressed.
	U File was uncompressed.
	T File was tar file.
	– No action taken.
8	File was sent to user (o>utgoing) or received from user (i>ncoming).
9	Accessed anonymously (r>eal, a>nonymous); mostly for FTP.
10	Local username or, if guest, ID string given (anonymous FTP password)
11	Service name ('ftp', other).
12	Authentication method (bitmask) . Always zero (0).
	0 None.
13	Always "*".

OPTIONS

–d	Debugging information is written to the `syslog`.
–l	Each `ftp`session is logged in the `syslog`. The `ftp` server will timeout after 15 minutes if the session is inactive.
–t ***timeout***	Set the maximum inactive period to *timeout* seconds. A client may also request a different *timeout* period.
–T ***maxtimeout***	Set the maximum inactive period to *maxtimeout* seconds. The default limit is 2 hours.
–u ***mask***	Set the default `umask` for file creation to *mask*. The default `umask` is 022.
–r	Disable all file modification and writing. Files are read-only. This option is available to all tuser types, whether real, guest, or anonymous, and it overrides the readonly keyword in the **`ftpaccess`**(4) file.
–s	Run in standalone mode. Instead of being invoked from **`inetd`**(1M), `in.ftpd` permanently listens for connections

itself. For a very busy server this greatly reduces overhead by eliminating unnecessary **exec**(2) operations.

–w *timeout* Run as an **inetd**(1M) wait server. This is variant of standalone; manages its own connections, but after a timeout period exits and returns, control to `inetd` *timeout* is specified in seconds. This is system default.

–L Log commands sent to the `in.ftpd` to the `syslog`. The **ftpaccess**(4) file overrides this option. If the –L flag is used, command logging will be on by default as soon as the FTP server is invoked.This will cause the server to log all `USER` commands, which if a user accidentally enters a password for that command insteadof the username, will cause passwords to be logged by way of `syslog`.

–i Log files received by the `in.ftpd` server to `/var/ftp/xferlog.data`. The **ftpaccess**(4) file overrides this option.

–o Log files transmitted by the `in.ftpd` server to the `syslog`. The **ftpaccess**(4) file overrides this option.

SECURITY

`in.ftpd` uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the module to be used for `in.ftpd.` The following is a partical `pam.conf` file with entries for the `in.ftpd` command, using the UNIX authentication, account management, and session management modules:

```
ftp auth  required /usr/lib/security/pam_unix.so.1
ftp account required /usr/lib/security/pam_unix.so.1
ftp session required /usr/lib/security/pam_unix.so.1
```

If there are no entries for the `ftp` service, then the entried for the “other” service will be used. Unlike `login`, `passwd`, and other commands, the `ftp` protocol will only support a single password. Using multiple modules will prevent `in.ftpd` from working properly.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

FILES

`/etc/inet/ftpusers` Logins explicitly denied access.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpu
Interface Stability	Evolving

SEE ALSO

ftp(1M), **ftpaddhost**(1M), **ftpcount**(1M), **ftpshut**(1M), **syslogd**(1M), **getusershell**(3C), **ftpaccess**(4), **ftpconfig**(4), **ftpconversions**(4), **ftphosts**(4), **ftpservers**(4), **shells**(4), **attributes**(5)

CHAPTER 1

man Pages(4): File Formats

NAME

ftpintro.4 – introduction to the host configuration files for the Sun Internet FTP Server

DESCRIPTION

The man pages offer detailed instruction and examples on keywords and parameters for each configuration file.

LIST OF FILES

ftpaccess(4)	The ftpaccess file is used to configure the operation of the FTP server.
ftpconversions(4)	The ftpd file-conversions database, used to specify the prefix, postfix, type, and conversion command to i **in.ftpd**(1M).
ftphosts(4)	The `ftphosts` file is used to deny access to certain accounts from various hosts.
ftpservers(4)	The `ftpservers` file contains a list of virtual servers. Each virtual server listed has its own configuration file that completely replaces ftpaccess(4).
xferlog(4)	The `xferlog` file contains logging information from the FTP server daemon, **in.ftpd**(1M)

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpr
Interface Stability	Evolving

SEE ALSO

ftpIntro(1M), **attributes**(5)

NAME

ftpaccess – ftpd configuration file

SYNOPSIS

`/etc/inet/ftpaccess`

DESCRIPTION

The `ftpaccess` file is used to configure the operation of the File Transfer Protocol (“FTP ”) server.

Access Capabilities Parameter Descriptions

In the keyword descriptions below, the following general parameters are used:

<addrglob>

A pattern to match a host name of the form host.subdomain.domain. An asterisk ("*") can be put first to match only the tail of a host name; it can be put in the middle to match exactly one component between two dots; or it can be put last to match only initial components. Examples are: "*.domain", "host.*", and "host.*.domain". Note that if reverse lookups are disabled, the host names are based on the IP address (for example, `10.207.82.135`), and there are exactly four components separated by dots.

<class>

A class name, defined by the CLASS keyword. Each login is matched against the defined classes, and considered to belong to the class it first matches.

<classglob>

A pattern to match against a class name as defined by the CLASS keyword.The pattern is as described in **`fnmatch`**(5).

<cmd>

A command to execute, for example, `/bin/ls`.

<dir>

A directory name, for example, `/etc`.

<filename>

A filename, for example, `/etc/Welcome`.

<groupname>

A system group, as defined in `/etc/group` or equivalent.

<message_file>

The location of a file containing a message to be output.

<string>

Any sequence of printable characters. Matching pairs of single or double quotes can be used to include blanks. Backslash can be used to quote any single character or include the special characters CR (\\r) and LF (\\n).

<typelist>

One or more of the words, “real,” “guest,” or “anonymous,” separated by commas.

<yes|no>

One of the words "yes" or "no". The words "on"/"off" are equivalent.

Access Capabilities Keyword Descriptions

autogroup <groupname><class>[<class>...]

If an ANONYMOUS user is a member of any of <class> the ftp server will perform a **setegid**(2) to <groupname>. This allows access to group-and-owner-read-only files and directories to a particular class of anonymous users. <groupname> is a valid group from `/etc/group`.

class<class><typelist><addrglob>[<addrglob> ...]

Define a class of users, with source addresses of the form <addrglob> . Multiple members of <class> may be defined. There may be multiple "class" commands listing additional members of the class. If multiple "class" definitions can apply to a session, the first one listed in the access file is used. Failing to define a valid class for a host will cause access to be denied. <typelist> is a comma-separated list of any of the keywords "anonymous," "guest," and "real."

If the "real" keyword is included, the class can match users using FTP to access real accounts, and if the "anonymous" keyword is included the class can match users using anonymous FTP. The "guest" keyword matches guest access accounts. See "guestgroup" for more information.

<addrglob> may be a globbed domain name or a globbed numeric address.

deny<addrglob><message_file>

Always deny access to host(s) matching <addrglob>. <message_file> is displayed. <addrglob> may be "!nameserved" to deny access to sites without a working nameserver.

guest<addrglob><message_file>

If a real user is a member, the session is set up exactly as with anonymous FTP. In other words, **chroot**(1M) is done, and the user is no longer permitted to issue the USER and PASS commands.

The user's home directory must be properly set up, exactly as anonymous FTP would be. The home directory field of the passwd entry is divided into two directories.

The first field is the root directory which will be the argument to the **chroot**(1M) command. The second field is the user's home directory relative to the root directory. The two fields are separated by a "`/./`".

For example, in `/etc/passwd`, the real entry is: `guest1::100:92:GuestAccount:/ftp/./incoming:/etc/ftponly` . When guest1 successfully logs in, the FTP server will `chroot` ("/ftp") and then **chdir**(1M) ("/incoming"). The guest user will only be able to access the directory structure under `/ftp`, which will look and act as `/` to guest1, just as an anonymous FTP user would.

limit<class><n><times><message_file>

Limit <class> to <n> users at times <times>, displaying <message_file> if user is denied access. Limit check is performed at login time only. If multiple "limit" commands can apply to the current session, the first applicable one is used. Failing to define a valid limit, or a limit of -1, is equivalent to unlimited. <times> is day-of-week and time-of-day when this class may connect, for example, `MoTuTh0800-1700`. Use "Any" for any day. Use "Never" for classes that may never login. If a day is specified but no time, then any time that day is assumed.

loginfails<n>

After login failures, log a "repeated login failures" message and terminate the FTP connection.The default value is 5.

maxusers <n>

Sets the bound on the total number of users of all classes logged on simultaneously to <n>.

noretrieve<filename><filename> ...

Always deny retrievability of these files. If the files are an absolute path specification, then only those files are marked ungettable, otherwise all files with matching filename are refused transfer. For example, "`noretrieve /etc/passwd core`" specifies no one will be able to get the file `/etc/passwd` whereas a file '`passwd`' may be transferred if it is not in `/etc`. On the other hand, no one will be able to get files named '`core`' regardless of where they are. No globbing is done.

private<yes|no>

After user logs in, the SITE GROUP and SITE GPASS commands may be used to specify an enhanced access group and associated password. If the group name and pass word are valid, the user becomes (by way of **`setegid`**(2)) a member of the group specified in the group access file `/etc/inet/ftpgroups`.The format of the group access file is

```
access_group_name:encrypted_password:real_group_name
```

where `access_group_name` is a string consisting of all printable characters except colon(`:`). `encrypted_password` is the password encrypted by way of **`crypt`**(1), exactly like in `/etc/shadow`. `real_group_name` is the name of a valid group listed in `/etc/group`. Note that this keyword is obsolete and is provided only for compatibility.

Informational Capabilities

banner <message_file>

Works similarly to the message command, except that the banner is displayed before the user enters the username/password. The <message_file> is relative to the real system root, not the base of the anonymous FTP directory.

Use of this command can completely prevent non-compliant FTP clients from making use of the FTP server. Not all clients can handle multi-line responses, which is how the banner is displayed.

email <string>

Defines the email address of the FTP archive maintainer. This string will be printed every time the %E magic cookie is used in message files.

hostname <string>

Specifies that the FTP server should identify itself with <string>. The default is a hostname that resolves to the address of an interface on the server host.

message <message_file>{<when> {<class> ...}}

Define a file with <message_file> that that **in.ftpd**(1M) will display to the user at login time or upon using the change working directory command. The <when> parameter may be "LOGIN" or "CWD=". If <when> is "CWD=" then <dir> specifies the new default directory which will trigger the notification.

The optional specification allows the message to be displayed only to members of a particular class. More than one class may be specified.

There can be "magic cookies" in the readme file which cause the FTP server to replace the cookie with a specified text string:

%T Local time; for example, Thu Nov 15 17:12:42.

%F Free space in partition of CWD, in megabytes.

%C Current working directory.

%E The maintainer's email address as defined in `ftpaccess`.

%R Remote host name.

%L Local host name.

%U Username given at login time.

%u Same as %U.

%M Maximum allowed number of users in this class.

%N Current number of users in this class.

%s Time when FTP shutdown began.

%r Time when new logins will be refused because of shutdown.

%d Time when current logins will be disconnected.

%Z Local timezone.

%% A single percent (%) character.

The message will only be displayed once to avoid annoying the user. Remember that when messages are triggered by an anonymous FTP user, they must be relative to the base of the anonymous FTP directory tree.

readme <message_file> {<when> {<class>}}

Define a file <message_file> that **in.ftpd**(1M) will display at login time or upon using the change working directory command that indicates the file exists and was modified on the specified date. The <when> parameter may be "LOGIN" or "CWD=". If <when> is "CWD=", <dir> specifies the new default directory which will trigger the notification. The message will only be displayed once, to avoid bothering users. Remember that when README messages are triggered by an anonymous FTP user, the <message_file> must be relative to the base of the anonymous FTP directory tree.

The optional specification allows the message to be displayed only to members of a particular class. More than one class may be specified.

sysident <string>

Sets the system identification returned in the banner string. Overrides any value set in /etc/default/ftp.

Logging Capabilities

log commands <typelist>

Enables logging of individual commands by users. <typelist> is a comma-separated list of any of the keywords "anonymous," "guest" and "real." If the "real" keyword is included, logging will be done for users using FTP to access real accounts, and if the "anonymous" keyword is included logging will done for users using anonymous FTP. The "guest" keyword specifies guest access accounts. See "guestgroup" for more information.

log transfers <typelist><directions>

Enables logging of file transfers for either real or anonymous FTP users. Logging of transfers to the server (incoming) can be enabled separately from transfers from the server (outbound).

<typelist> is a comma-separated list of any of the keywords "anonymous", "guest" and "real". If the "real" keyword is included, logging will be done for users using FTP to access real accounts, and if the "anonymous" keyword is included logging will done for users using anonymous FTP. The "guest" keyword matches guest access accounts. See "guestgroup" for more information. <directions> is a comma-separated list of any of the two keywords "inbound" and "outbound," and will respectively cause transfers to be logged for files sent to the server and sent from the server.

logfile <filename>

Sets the file to which to log file transfers. <filename> is an absolute path on the server.

Miscellaneous Capabilities

alias <string><dir>

Defines an alias, <string>, for a directory. It can be used to add the concept of logical directories. For example: `alias rfc: /pub/doc/rfc` would allow the user to access `/pub/doc/rfc` from any directory by the command "`cd rfc:`".

Aliases only apply to the `cd` command. NOTE: This functionality is provided for compatibility with `wu-ftpd` and is obsolete. Use symlinks instead.

cdpath <dir>

Defines an entry in the cdpath. This defines a search path that is used when changing directories. For example:

```
example% cdpath /pub/packages
example%cdpath /.aliases
```

would allow the user to move into any directory directly under either the `/pub/packages` or the `/.aliases` directories. The search path is defined by the order the lines appear in the `ftpaccess` file.

If the user were to give the command:

```
example% cd foo
```

the directory will be searched for in the following order:

`./foo` (an alias called "foo")

`/pub/packages/foo`

`/.aliases/foo`

The cdpath is only available with the cd command. If you have a large number of aliases, you might want to set up an aliases directory with links to all of the areas you wish to make available to users.

compress <yes|no><classglob>[<classglob> ...}

Enables **compress** (1) or **tar**(1) capabilities for any class matching any of <classglob>. The actual conversions are defined in the external file `/etc/inet/ftpconversions`.

fastanon <yes|no>

Specifies that anonymous logins do not have to be proper logins. Instead, only the uid, group, and fs root are set up. Does not perform a `keylogin`, update `utmp` or `wtmp`, or do any of the other usual login processing.

idletimeout <n>

Sets the time in seconds after which an idle login is disconnected. The default is 900.

lslong <cmd><arguments>

Specifies the command and arguments used for a "long" file listing. It can be "`*ls`", in which case a built-in function is used. The built-in command knows the options– `1`, –`C`, –`F`, –`l`, –`a`, and–`d`, and it produces output similar to **ls**(1).

lsshort <cmd><arguments>

Specifies the command and arguments used for a "short" file listing. It can be "`*ls`", in which case a built-in function is used. The built-in command knows the options– `1`, –`C`, –`F`, –`l`, –`a`, and–`d`, and it produces output similar to **ls**(1).

path <dir>:<dir> ...

Sets the contents of PATH environment variable as passed down to the program run (for example, `ls`, `tar`, or `compress`). The default path is "`/bin`".

readonly <yes|no>

"Readonly yes" specifies that a server is readonly. No files can be changed, deleted, renamed, or uploaded; no directories can be created. All operations that would modify a file or directory are effectively disabled, and they fail

with a cause message of "readonly server". This configuration command can be overridden by the –r option to **in.ftpd**(1M).

rlookup <yes|no>

Turns on/off reverse IP addr-to-hostname lookups. With "rlookup no", the "remote host" name becomes its IP address in decimal dot notation. This will then appear in logs, and it can be used for access control in place of the host name. The default is "no".

root <dir>

Specifies the root directory of a virtual server. Use only in server-specific configuration files referenced from **ftpservers**(4); elsewhere, it is ignored.

shell <cmd>

Sets the value of the SHELL environment variable as passed down to exec'd programs. The default path is /bin/sh.

shutdown <message_file>

If the file pointed to by <message_file> exists, the server will check the file regularly to see if the server is going to be shut down. If a shutdown is planned, the user is notified, new connections are denied after a specified time before shutdown and current connections are dropped at a specified time before shutdown. The external program **ftpshut**(1M) uses this file to communicate shutdown data.

tar <yes|no><classglob>[<classglob> ...}

Enables **compress**(1) or **tar**(1) capabilities for any class matching any of .of <classglob>. The actual conversions are defined in the external file /etc/inet/ftpconversions.

timezone <tz>

Sets the timezone for the network daemon process and all children. This affects the time output in file listings. Note that for anonymous FTP, the timezone description file must be found in ~ftp/usr/share/lib/timezone. Unless specified, this is inherited from the parent used to start **in.ftpd**(1M).

uselocks <yes|no>

Specifies that file locking is used during uploads. When an upload begins, the file is locked and write-only; when the upload finishes, the file mode bits are set to their final value and the file is unlocked.

An interrupted transfer leaves a partially uploaded file that is not readable. The upload can then later be restarted and finished.

The purpose is to prevent anonymous users from downloading files which are in progress of being uploaded, or files for which the upload was interrupted or failed due to an error on the server. The default is `yes`.

virtual <address><dir><message_file>

Enables the virtual FTP server capabilities. The <address> is the IP address of the virtual server. The <dir> parameter defines the root of the anonymous FTP area for the virtual server, and <message_file> is the banner to print on connection.

Note that all virtual servers defined this way share the same `ftpaccess` file.

Permission Capabilities

chmod <yes|no><typelist>

Allows or disallows the ability to perform the specified function. By default, all users are allowed.

delete <yes|no><typelist>

Allows or disallows the ability to perform the specified function. By default, all users are allowed.

overwrite <yes|no><typelist>

Allows or disallows the ability to perform the specified function. By default, all users are allowed.

rename <yes|no><typelist>

Allows or disallows the ability to perform the specified function. By default, all users are allowed.

passwd-check <none|trivial|rfc822> (<enforce|warn>)

Define the level and enforcement of password checking done by the server for anonymous FTP.

none No password checking performed.

trivial password must contain an '@'.

rfc822 password must be RFC 822 compliant.

warn warn, but permit login.

enforce notify and deny login.

path-filter <typelist> <message_file> <allowed_charset> <disallowed_charset1><disallowed_charset2>...

For users in <typelist>, path-filter defines regular expressions that control what a filename can or cannot be. There may be multiple disallowed regular expressions.

If a filename is invalid due to failure to match the `regexp` criteria, <message_file> will be displayed to the user. For example:

```
path-filter anonymous /etc/inet/ftp.pathmsg ^[-A-Za-z0-9._]*$ ^\\. ^-
```

specifies that all upload filenames for anonymous users must be made of only the characters A-Z, a-z, 0-9, and "._-" and may not begin with a "." or a "-". If the filename is invalid, `/etc/inet/ftp.pathmsg` will be displayed. Note that to enter a "\" for regular expressions, it must be escaped using another "\".

umask <yes|no><typelist>

Allows or disallows the ability to perform the specified function. By default, all users are allowed.

upload <root-dir> <dirglob> <yes|no> <owner><group><mode> ["dirs"|"nodirs"]

Define a directory with <dirglob> that permits or denies uploads. <dirglob> is a pattern as described in **`fnmatch`**(5) .

If it does permit uploads, all files will be owned by <owner> and <group> and will have the permissions set according to <mode>.

Directories are matched on a best-match basis. For example:

```
example% upload /var/ftp * no
example% upload /var/ftp /incoming yes ftp daemon 0666
```

```
example% upload /var/ftp /incoming/gifs yes jlc guest 0600 nodirs
```

The optional "dirs" and "nodirs" keywords can be specified to allow or disallow the creation of new sub-directories using the **mkdir**(1) command. The upload keyword only applies to users who have a home directory (the argument to the **chroot**(1) of <root—dir>.

FILES

`/etc/inet/ftpaccess`

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpr
Interface Stability	Evolving

SEE ALSO

ls(1), **ftpaddhost**(1M), **ftpshut**(1M) **in.ftpd**(1M), **ftpconversions**(4), **ftpservers**(4) **timezone**(4), **attributes**(5), **fnmatch**(5)

NAME

ftpconversions – ftpd conversions database

SYNOPSIS

`/etc/inet/ftpconversions`

DESCRIPTION

The conversions known by **in.ftpd**(1M) and its attributes are stored in an ASCII file that is structured as below. Each line in the file provides a description for a single conversion. Fields are separated by colons (:). Note that comments cannot be put in this file.

```
%s:%s:%s:%s:%s:%s:%s:%s
 1 2 3 4 5 6 7 8
```

The field descriptions are as follows:

Field	*Description*
1	Strip prefix.
2	Strip postfix. A postfix (one or more file name extensions) that is removed by running the external command. For example, the external command "`tar xf %s`" would remove ".tar".
3	Addon prefix.
4	Addon postfix. Similar to the strip postfix above, except the external command adds it instead. For example, the external command "`tar cf %s`" would add ".tar".
5	External command. This is a command to run to produce an output file from an input file. It may contain exactly one "%s", which is replaced with the input file name. It must automatically create a correctly name output file. If no "%s" is used, the file name is appended to the command, separated by a blank.
6	Types. This specifies what file types this conversion can be performed on. It can be `T_REG` for any regular file, `T_DIR` for a directory, and `T_ASCII` that it can be performed in ASCII transfer mode. See **in.ftpd**(1M). If `T_ASCII` is not specified, it is assumed the conversion requires a binary (Image) transfer mode. Multiple values can be combined with a vertical bar, for example,`T_REG|T_DIR` would specify either a regular file or a directory are acceptable, and that the transfer mode cannot be ASCII.

7 Options. This explains to the FTP server what allow or deny the operation. See the keywords `COMPRESS` and `TAR` in **`ftpaccess`**(4). The value of this field is one of the words `O_TAR`, `O_COMPRESS`, or `O_UNCOMPRESS`. If O_TAR is specified, the `TAR` keyword in the `ftpaccess` file specifies whether to permit the operation. In addition, if a directory contains a file named "`.notar`", then to `tar` that directory is always denied. `O_COMPRESS` and `O_UNCOMPRESS` are equivalent, and are allowed or denied according to the `COMPRESS` keyword in the `ftpaccess` file.

8 Description. This part is not used by the FTP server, and is available to store a comment.

EXAMPLES

EXAMPLE 1 Using `ftpconversions`

The line

```
:::.tar:tar -cf %s:T_REG|T_DIR:O_TAR: Tar file or dir
```

specifies that to add "`.tar`" to a file or directory name, run the command "`tar -cf %s`"; that it can be performed on regular files or directories; implicitly that it can only be performed in binary transfer mode; and that it is a "`tar`" operation, that is, access is controlled through `/etc/inet/ftpaccess`. See **`in.ftpd`**(1M).

FILES

`/etc/inet/ftpconversions`

`/etc/inet/ftpaccess`

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpr
Interface Stability	Evolving

SEE ALSO

`in.ftpd`(1M), **`ftpaccess`**(4), **`attributes`**(5)

NAME

ftphosts – ftpd individual user host access file

SYNOPSIS

`/etc/inet`

DESCRIPTION

The `ftphosts` file is used to deny access to certain accounts from various hosts.

The following illustrates the usage for the `ftphosts` file:

`allow<username><addrglob>[<addrglob>...]`

Only allow host(s) matching <addrglob> to log in as <usrname>.

`deny<username><addrglob>[<addrglob>...]`

Always deny hosts() matching <addrglob> to log in as <username>.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpr
Interface Stability	Evolving

SEE ALSO

attributes(5)

NAME

ftpservers – FTP Virtual Server list

SYNOPSIS

`/etc/inet/ftpservers`

DESCRIPTION

The `ftpservers` file contains a list of virtual servers. Each virtual server listed has its own configuration file that completely replaces **`ftpaccess`**(4). The difference between a virtual server created through a “virtual “statement in **`ftpaccess`**(4) and one that is created by an entry in the `ftpservers` file is that the former shares the system’s default **`ftpaccess`**(4) file and configuration.

Each non-blank line in the `ftpservers` file defines one virtual server. The format of the line is:

```
<hostname> <config file>
```

The <hostname> is the fully qualified domain name of the virtual server, and it must be present in the `/etc/inet/hosts` file. In case of LDAP user connection, <hostname> must match the domain tree in the LDAP server. Moreover, <hostname> must be a fully qualified domain name. The <config file> is a replacement for the system default **`ftpaccess`**(4) configuration.

Anything following a hash mark is a comment in the `ftpservers` file. Single and double quotes and the backslash can be used as they are used in **`ftpaccess`**(4), to include blanks or special characters in either <hostname> or <config file>.

EXAMPLES

EXAMPLE 1 List of Virtual FTP Servers with Private Configuration Files

The following entries from the `ftpservers` file shows a list of virtual FTP servers and their private configuration files:

```
ftp.gadgets.com     /etc/inet/gadgets.conf
ftp.sprockets.com   /etc/inet/sprockets.conf
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpr
Interface Stability	Evolving

SEE ALSO

`in.ftpd`(1M), **`ftpaccess`**(4), **`attributes`**(5)

NOTES

A “virtual” statement in a virtual server’s private configuration file has no effect and is silently ignored. It is not possible to have virtual servers within virtual servers.

A readonly yes statement in the system default configuration file cannot be undone in a virtual server’s configuration file. The virtual server will also be readonly. The same applies to the `–r` option to **`in.ftpd`**(1M). A virtual server can only be made readonly by making the system default non-readonly and putting “readonly yes” in the virtual server’s configuration file.

NAME

xferlog – FTP server logfile

SYNOPSIS

`/usr/adm/xferlog`

DESCRIPTION

The `xferlog` file contains logging information from the FTP server daemon, **`in.ftpd`**(1M). Each server entry is composed of a single line of the following form, with all fields separated by spaces.

current-time transfer-time remote-host file- size filename
transfer-type special-action-flag directionaccess-mode username
service-name authentication-method authenticated-user-id

The fields are defined as follows:

current-time The current local time in the form "`DDD MMM dd hh:mm:ss YYYY`". Where `DDD` is the day of the week, `MMM` is the month, `dd` is the day of the month, `hh` is the hour, `mm` is the minutes, `ss` is the seconds, and `YYYY` is the year.

transfer-time The total time in seconds for the transfer.

remote-host The remote host name.

file- size The size of the transfered file in bytes.

filename The name of the transfered file.

transfer-type A single character indicating the type of transfer:

- **a** For an ascii transfer.
- **b** For a binary transfer.

special-action-flag One or more single character flags indicating any special action taken. These flags include:

- **C** File was compressed.
- **U** File was uncompressed.
- **T** File was a tar file.
- **–** No action was taken.

direction The direction of the transfer:

- **o** Outgoing.

i Incoming.

access-mode The method by which the user is logged in. Can be one of:

a For an anonymous guest user.

g For a passworded guest user. See the description of the guestgroup command in **ftpaccess**(4).

r For a local, authenticated (real) user.

username The local username, or if guest, the ID string given.

service-name The name of the service being invoked, usually FTP.

authentication-method Always zero (0).

0 None.

authenticated-user-id Always "*".

FILES /usr/adm/xferlog

ATTRIBUTES See **attributes**(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWftpr
Interface Stability	Evolving

SEE ALSO **in.ftpd**(1M), **attributes**(5), **ftpaccess**(4), **attributes**(5)

PART III Sun Internet Administrator

Sun™ Internet Administrator™ 1.1 man pages. These man pages are located in `/opt/SUNWixamc/man`.

CHAPTER 1

man Pages(1m): Maintenance Commands

NAME

mcIntro – Introduction to the Sun Internet Administrator software command-line utilities for Solaris ISP Server

DESCRIPTION

The man pages offer detailed instruction and examples on the options and subcommands for each utility. The command line utilities are available to initialize Sun Internet Administrator entries in Sun Directory Services, register the services and service hosts with Sun Internet Administrator, and create administrators for the registered services.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamm
Availability	SUNWispm
Interface Stability	Evolving

NOTES

mcadd(1m) Adds a specified registered service to Sun Internet Administrator. This command must be run on the host where Sun Internet Administrator is installed and you require root access to run this command. This command is located in `/opt/SUNWixamc/sbin`

mcaddadm(1m) Creates an administrator in Sun Internet Administrator and authorizes the administrator to services specified by you. This command must be run on the machine where Sun Internet Administrator is installed and you require root access to run this command. This command is located in `/opt/SUNWixamc/sbin`

mcadmpwd(1m) Sets the password for an existing administrator in Sun Internet Administrator. This command can be used to change an existing password. You require root access to run this command. This command is located in `/opt/SUNWixamc/sbin`

mcdsinit(1m) Makes initial service entries for Sun Internet Administrator in Sun™ Directory Services. You require root access to run this command. This command is located in `/opt/SUNWixamc/sbin`

mcdsclean(1m) Checks to see if registered services are still available for management from Sun Internet Administrator, and if not

found, removes the service entries in the directory services. This command must be run on the machine where Sun Internet Administrator is installed and you require root access to run this command. This command is located in `/opt/SUNWixamc/sbin`.

`mchelp(1m)` Displays the release version of Sun Internet Administrator installed on the system and all command line utilities associated with it. This command must be run on the machine where Sun Internet Administrator is installed. This command is located in `/opt/SUNWisp/SUNWixamr/`

`mchostls(1m)` Lists the package name and service name for each service installed and registered on the specified host. Use this utility to discover what services are available for management through Sun Internet Administrator. This command must be run on the machine where Sun Internet Administrator is installed and you require root access to run this command. This command is located in `/opt/SUNWixamc/sbin`.

`mcreg(1m)` Register a service application to make it available for management through Sun Internet Administrator. Records the service's user interface information for Sun Internet Administrator. This command must be run on the machine where Sun Internet Administrator is installed and you require root access to run this command. This command is located in `/opt/SUNWisp/SUNWixamr`

`mcrm(1m)` Remove a service from the services list in Sun Internet Administrator, so that the service is no longer centrally managed. This command must be run on the machine where Sun Internet Administrator is installed and you require root access to run this command. This command is located in `/opt/SUNWixamc/sbin`.

`mcrmadm(1m)` Removes an authorized administrator from the system to deny access to all services registered with Sun Internet Administrator or to only deny the administrator access to specific services. This command must be run on the machine where Sun Internet Administrator is installed and you require root access to run this command. This command is located in `/opt/SUNWixamc/sbin`

mcunreg(1m) Unregisters a service application to make it unavailable for management through Sun Internet Administrator. Removes the service's user interface information from Sun Internet Administrator. This command must be run on the machine where Sun Internet Administrator is installed and you require root access to run this command. This command is located in `/opt/SUNWisp/SUNWixamr`

NAME

mcadd – Add a registered (mcreg) service to Sun™ Internet Administrator™.

SYNOPSIS

mcadd –c *componentID* –v *componentVersion* –h *host*

DESCRIPTION

Adds the specified service to Sun Internet Administrator. Run mcreg before running mcadd and run mcadd on the machine where Sun Internet Administrator is installed. Root access is required to run this command.

OPTIONS

–c ***componentID***	Specifies the service to be managed, by a service-unique identifier. The package name is recommended because it is guaranteed to be unique.
–h ***host***	Specifies the host (by DNS name) where the service to be managed is installed.
–v ***componentVersion***	Specifies the release version of the service to be managed. Recommended format is major.minor.

EXAMPLES

EXAMPLE 1 Adding a Service

```
# /opt/SUNWixamc/sbin/mcadd -c SUNWftp -v 1.1 -h thurblig
```

This use of mcadd adds Sun™ Internet FTP Server™ version 1.1, installed on the machine thurblig, to Sun Internet Administrator.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamc
Interface Stability	Evolving

SEE ALSO

mcaddadm(1m), **mcadmpwd**(1m), **mcdsinit**(1m), **mcdsclean**(1m), **mchelp**(1m), **mchostls**(1m), **mcreg**(1m), **mcrm**(1m), **mcrmadm**(1m), **mcunreg**(1m)

NAME

`mcaddadm` – Add an administrator to Sun™ Internet Administrator™ authorized administrators list.

SYNOPSIS

mcaddadm {–n *name*} [–p *password*] –i *componentID-version*...

DESCRIPTION

Creates an administrator in Sun Internet Administrator with access to the version of services listed as the *componentID-version* option. To grant the administrator access to an additional service, run `mcrmadm`, then `mcaddadm`. Run `mcaddadm` on the machine where Sun Internet Administrator is installed. Root access is required to run this command.

OPTIONS

–i ***componentID-version*** — Specifies, by a service-specific identifier (package name) and version number of the service application, the Solaris ISP Server™ service for the administrator to manage. Enter a –i option for each separate service required.

Note - Run `mchostls` for a list of valid components that you can specify for the administrator to manage.

If this administrator will manage Sun Internet Administrator, enter `SUNWixamc` here. Do not list other packages, because access to Sun Internet Administratorauthorizes access to all services as well.

–n ***name*** — Specifies the name this administrator uses to log in to Sun Internet Administrator.

–p ***password*** — Specifies the password this administrator uses to log in to Sun Internet Administrator. Do not string this option with the command as password can be entered in non-printing mode. For security reasons, wait for the command to prompt for this password.

EXAMPLES

EXAMPLE 1 Adding an Administrator to Two Services

```
# /opt/SUNWixamc/sbin/mcaddadm -n april -i SUNWixftp-1.1 -i SUNWixnsa-2.0
# password:
```

In the foregoing, the administrator `april` is created. The command prompts for her password to authorize her to manage both Sun™ Internet FTP Server™ 1.1, and Sun™ Internet News Server™ 2.0 and the entry is not echoed on the screen.

EXAMPLE 2 Adding an Administrator of Sun Internet Administrator

```
# /opt/SUNWixamc/sbin/mcaddadm -n vickie -i SUNWixamc-1.1
# password:
```

In the foregoing, the administrator `vickie` is created. The utility prompts for her password to authorize her to manage Sun Internet Administrator 1.1 and the entry is not echoed on the screen. She can also administer any managed services.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamc
Interface Stability	Evolving

SEE ALSO

mcadd(1m), **mcadmpwd**(1m), **mcdsinit**(1m), **mcdsclean**(1m), **mchelp**(1m), **mchostls**(1m), **mcreg**(1m), **mcrm**(1m), **mcrmadm**(1m), **mcunreg**(1m).

NOTES

If you enter the password option on the command line, it is visible to anyone who can see your screen and to anyone issuing a `ps` command while `mcaddadm` is running. You can omit the –`p` option on the command line, avoiding this risk. When the command prompts you for the password, your entry is not echoed on the screen.

NAME

mcadmpwd – Set the password for an administrator in Sun™ Internet Administrator™.

SYNOPSIS

mcadmpwd –n *name* [–p *password*]

DESCRIPTION

Sets the password of an existing administrator. Run mcadmpwd on the machine where Sun Internet Administrator is installed. Root access is required to run this command.

OPTIONS

–n ***name*** Specifies the administrator, by login name.

–p ***password*** Specifies the new password for the administrator. Do not string this option with the command. For security reasons, wait for the command to prompt for the password.

EXAMPLES

EXAMPLE 1 Setting an Administrator's Password

```
# /opt/SUNWixamc/sbin/mcadmpwd -n April
# password:
# Re-enter password:
```

In the foregoing example, the administrator April is prompted for her new password playground and the password is not echoed on the screen.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamc
Interface Stability	Evolving

SEE ALSO

mcadd(1m), **mcaddadm**(1m), **mcdsinit**(1m), **mcdsclean**(1m), **mchelp**(1m), **mchostls**(1m), **mcreg**(1m), **mcrm**(1m), **mcrmadm**(1m), **mcunreg**(1m).

NOTES

If you enter the password option on the command line, it is visible to anyone who can see your screen and to anyone issuing a `ps` command while `mcadmpwd` is running. You can omit the `–p` options on the command line, avoiding this risk. The command then prompts you for this password, and your entry is not echoed on the screen.

NAME

mcdsclean – Checks to see if registered services are still installed and if not found, deletes the service entries in the directory services.

SYNOPSIS

mcdsclean –a

mcdsclean *hostname*

DESCRIPTION

Checks the hosts registered with Sun Internet Administrator to see if services are still available for management. If services are not found, removes the service entries from the directory services.

This command gets executed every time you register a service for management from Sun Internet Administrator GUI. You must run this command on Sun Internet Administrator host to check a specific host or all hosts registered for management from Sun Internet Administrator. You need root access to run this command.

OPTIONS

–a	Indicates that all hosts must be checked for services still available for management from Sun Internet Administrator.
hostname	Specifies the host that must be checked for services still available for management from Sun Internet Administrator. When specifying host to check, specify the same host name you specified when registering the host.

EXAMPLES

EXAMPLE 1 Check All Hosts

```
# /opt/SUNWixamc/sbin/mcdsclean –a
```

In the foregoing example, all hosts registered with Sun Internet Administrator are checked (using `mchostls –h`) to see if services are still available for management from Sun Internet Administrator. If any service is not found installed, the service entries are removed from the directory services.

EXAMPLE 2 Check A Host

```
# /opt/SUNWixamc/sbin/mcdsclean tulip
```

In the foregoing example, host tulip is checked (using `mchostls –h`) to see if services on host tulip, registered with Sun Internet Administrator, are still installed and available for management from Sun Internet Administrator. If registered services are not found, the service entries are deleted from the directory services.

ENVIRONMENT VARIABLES

See **`environ`**(5) for descriptions of the following environment variables that affect the execution of `command_name`: `NLSPATH`.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamc
Interface Stability	Evolving

SEE ALSO

`mcadd`(1m), **`mcaddadm`**(1m), **`mcadmpwd`**(1m), **`mcdsinit`**(1m) **`mchelp`**(1m), **`mcreg`**(1m), **`mcrm`**(1m), **`mcrmadm`**(1m), **`mcunreg`**(1m)

NAME

mcdsinit – Makes initial service entries in Sun™ Directory Services

SYNOPSIS

mcdsinit –d *DNofDSadmin* [–w *password*] –n *CconsoleAdmin* [–p *password*] [–q]

DESCRIPTION

Makes initial service entries for Sun™ Internet Administrator™ in Sun Directory Services. If Sun Directory Services is available, the Sun Internet Administrator administrator name and password, provided (by you), binds with Sun Directory Services using the directory services administrator distinguished name and password, and makes the initial service entries.

If Sun Directory Services is not available when installing Sun Internet Administrator, run this command on the machine where Sun Internet Administrator is installed to make the initial entries in Sun Directory Services. Root access is required to run this command.

OPTIONS

–d ***DNofDSadmin***	Specifies the distinguished name for binding to the Sun Directory Services as the directory services administrator.
h	Prints command usage message and exits.
–n ***ConsoleAdmin***	Specifies the name of the administrator of Sun Internet Administrator used to make the initial entries in the directory services.
–p ***password***	Indicates the password of the Sun Internet Administrator administrator used to make entries in the directory services. To ensure security for password, do not string this option with the command. Wait for the command to prompt you for the password.
–q	Suppresses output messages and the command is executed in a quiet mode. No error is reported.
–w ***password***	Indicates the password of Sun Directory Services administrator used for binding to the directory services. To ensure security for passwords, do not string this option with the command. Wait for the command to prompt you for the password.

EXAMPLES

EXAMPLE 1 Making Initial Entries

```
# /opt/SUNWixamc/sbin/mcdsinit -d cn=admin,o=sun,c=US -n april
DA administrator password:
```

(Continuation)

```
IA administrator password:
```

In the foregoing example,

- The directory services distinguished name and the Sun Internet Administrator administrator name are specified.
- The directory services and Sun Internet Administrator administrator passwords are omitted from the command line arguments. The utility prompts for these passwords and does not echo it on the command line, keeping the passwords secure.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamc
Interface Stability	Evolving

SEE ALSO

mcadd(1m), **mcaddadm**(1m), **mcadmpwdd**(1m), **mcdsclean**(1m), **mchelp**(1m), **mchostls**(1m), **mcreg**(1m), **mcrm**(1m), **mcrmadm**(1m), **mcunreg**(1m)

NOTES

If you enter the password option on the command line, it is visible to anyone who can see your screen and to anyone issuing a `ps` command while `mcdsinit` is running. You can omit the `–w` and `–p` options on the command line, avoiding these risks. The command then prompts you for these passwords, and your entry is not echoed on the screen.

NAME

`mchostls` – List all services on a given host.

SYNOPSIS

mchostls –h *host*

DESCRIPTION

Lists the package name and service name for each service installed and registered on the specified host. Use this utility to discover what services are available for management through Sun™ Internet Administrator™. Run `mchostls` on the machine where Sun Internet Administrator is installed. Root access is required to run this command.

OPTIONS

–h ***host*** Specifies the host whose installed services are to be displayed.

EXAMPLES

EXAMPLE 1 `mchostls` Output

Invoke `mchostls` by entering it at the command line:

```
# /opt/SUNWixamc/sbin/mchostls –h thurblig
```

The following is displayed on your screen:

```
ID                              Name
SUNWixnsa 1.1                   Sun(TM) News
SUNWixadm 1.1                   Sun(tm) FTP
```

The foregoing indicates that Sun Internet News Server and Sun Internet FTP Server are installed on thurblig and `mcreg` has been run for them, recording their management information.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamc
Interface Stability	Evolving

SEE ALSO

mcadd(1m), **mcaddadm**(1m), **mcadmpwd**(1m), **mcdsinit**(1m), **mcdsclean**(1m), **mchelp**(1m), **mcreg**(1m), **mcrm**(1m), **mcrmadm**(1m), **mcunreg**(1m)

NAME

`mcrm` – Delete a service from Sun™ Internet Administrator™.

SYNOPSIS

mcrm –c *componentID* –h *host* –v *componentVersion*

DESCRIPTION

Removes the specified service from Sun Internet Administrator, so that it is no longer centrally managed. Root access is required to run this command.

OPTIONS

–c ***componentID***	Specifies, by a service-unique identifier (package name), the service to be removed.
–h ***host***	Specifies the host where the service to be deleted is installed.
–v ***componentVersion***	Specifies the release version of the service to be managed.

EXAMPLES

EXAMPLE 1 Unmanaging a Service

```
# /opt/SUNWixamc/sbin/mcrm -c SUNWsamp -v 1.1 -h thurblig
```

In the foregoing, `SUNWsamp` version 1.1, installed on the machine thurblig, is deleted from Sun Internet Administrator. Even authorized administrators are no longer able to access the service's user interface through Sun Internet Administrator.

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamc
Interface Stability	Evolving

SEE ALSO

mcIntro(1m) **mcadd**(1m), **mcaddadm**(1m), **mcadmpwd**(1m), **mcdsinit**(1m), **mcdsclean**(1m), **mchelp**(1m), **mchostls**(1m), **mcreg**(1m), **mcrmadm**(1m), **mcunreg**(1m)

NOTES

Run `mcunreg` after running `mcrm` to completely disable the component for administration from Sun Internet Administrator. Running `mcrm` alone leaves the service available for registering with Sun Internet Administrator for administration.

NAME

`mcrmadm` – Delete an existing authorized administrator from Sun™ Internet Administrator™

SYNOPSIS

mcrmadm –`a`|–`i` *componentID-version* –`n` *name*

DESCRIPTION

Deletes an existing administrator registered with Sun Internet Administrator. If –`a` is entered, completely deletes the administrator from the system. If –`i` is entered, deletes the service specified by *componentID-version* from the administrator's authorized services list. The administrator remains in the system, but can no longer access the specified service. Multiple services can be deleted from the administrator's authorized service list by repeating the –`i` option for each service to be deleted. Root access is required to run this command.

OPTIONS

–`a`	Specifies the removal of the administrator, deleting access to all services.
–`i` ***componentID-version***	Specifies, by a service-specific identifier (package name) and version of service application, the service from which the administrator is removed.
–`n` ***name***	Specifies the administrator to remove, by login name.

EXAMPLES

EXAMPLE 1 Removing an administrator

```
# /opt/SUNWixamc/sbin/mcrmadm -a -n april
```

In the foregoing, `april` is removed from the system and cannot manage any services through Sun Internet Administrator.

```
# /opt/SUNWixamc/sbin/mcrmadm -i SUNWftp-1.1 -n april
```

In the foregoing, `april` no longer has access to Sun™ Internet FTP Server™ 1.1. She can still manage any other service for which she has authorization.

```
# /opt/SUNWixamc/sbin/mcrmadm -i SUNWftp-1.1 -i SUNWsns-1.1 -n april
```

In the foregoing, `april` no longer has access to Sun™ Internet FTP Server™ 1.1 and Sun™ Internet News Server™ 1.1. She can still manage any other service for which she has authorization.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixamc
Interface Stability	Evolving

SEE ALSO

mcIntro(1m) **mcadd**(1m), **mcaddadm**(1m), **mcadmpwd**(1m), **mcdsinit**(1m), **mcdsclean**(1m), **mchelp**(1m), **mchostls**(1m), **mcreg**(1m), **mcrm**(1m), **mcunreg**(1m)

PART IV Sun Internet News Server

Sun™ Internet News Server™ 1.1 man pages

CHAPTER 2

News Command Line Procedures

This section provides the procedures for the Sun™ Internet News Server™ command line tasks:.

This section assumes:

- Solaris ISP Services installation has been completed
- NEWS has been registered with the Sun Internet Admistrator (see the online help for the *Sun Internet Administrator Register Services Screen*)
- Sun Internet News Server is installed, and has been configured using the News Administration GUI.
- The directories `/opt/SUNWsns/sbin` and `/opt/SUNWsns/bin` are in your root $PATH.

2.1 Procedures

The following procedures explain Sun Internet News Server maintenance tasks.

2.1.1 Start/Stop Reader/Feeder Servers

The **snsnews**(1m) command is used to start and stop the News reader and feeder daemons, for example:

```
# /etc/init.d/snsnews start
# /etc/init.d/snsnews stop
```

2.1.2 Newsgroup Tasks

2.1.2.1 Expiration

Newsgroup article expiration is controlled by the configuration file `/var/news/config/expire.ctl` (see **`expire.ctl`**(4) for an extensive description). The file expire.ctl contains two types of entries; one for history retention, and one for newgroup article retention.

The format for newsgroup article retention is:

`newsgroup:newsgroup-type:mindays:maxdays:purgerafter`

For example, the entries

```
*:A:5:10:15
alt.binaries.pictures.animals.*:U:2:3:5
```

do two things:

- Set the default expiration parameters for all newsgroups to be “keep articles for a minimum of five days, expire articles after ten days, and purge articles after fifteen days.”
- Set the expiration parameters on `alt.binaries.pictures.animals.*`, an unmoderated group, to be “keep articles for a minimum of two days, expire articles after three days, and purge articles after five days.”

You should expire and purge articles more frequently in groups that are likely to use a lot of storage space, such as groups where binaries are posted.

After you make changes to `/var/news/config/expire.ctl`, you must stop then start the news server using `snsnews`.

2.1.2.2 Newsgroup Blocking

Blocking news groups from being fed to downstream servers is controlled by the configuration file `/var/news/config/newsfeeds` (see **`newsfeeds`**(4) for an extensive description).

The format for newsgroup blocking is:

`server-name:sendgroup,!blockgroup1,!blockgroup2...::`

For example, the entries

```
ME:*,!control,!junk,!local.*::
news.otherplace.com:*,!alt.binaries.*,!local.*::
```

do two things:

- Send all news groups except `control`, `junk`, and any group beginning `local.` to all servers by default.
- Send all groups except groups begining with `alt.binaries.` or `local.` to the server named `news.otherplace.com`.

After you make changes to `/var/news/config/newsfeeds`, you must stop then start the news server using `snsnews`.

CHAPTER 2

man Pages(1m): Maintenance Commands

NAME

newsIntro.1m – Introduction to the host configuration software command-line utilities for the Sun Internet News Server

DESCRIPTION

The man pages offer detailed instruction and examples on options and subcommands for each utility.

EXIT STATUS

Upon termination, each command returns the following exit values:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Evolving

SEE ALSO

newsIntro(4)

NOTES

archive(1m) archives Usenet articles. Files are copied to a directory within the archive directory, /var/news/storage/archive.

batcher(1m) article-batching backend for the feeder daemon **innd**(1m). Batcher read uses a list of files to prepare news batches for the specified host.

buffchan(1m) buffered file-writing backend for News. Buffchan is intended to be called by the feeder daemon as an exploder feed..

crosspost(1m) creates the links for cross-posted articles. Crosspost is designed to be used by the feeder daemon to create the links as the articles come in.

ctlindd(1m) controls the feeder daemon daemon innd(1m).

cvtbatch(1m) convert simple batchfiles that contain just the article name to INN batchfiles that contain additional information about each article.

expire(1m) Usenet article and history expiration program.

expireover(1m)	expires entries from the news overview database.
expirerm(1m)	remove expired articles.
fastrm(1m)	quickly removes a set of files.
filechan(1m)	file-writing backend for the feeder daemon. Filechan is intended to be called by the feeder daemon as a channel feed.
inncheck(1m)	checks and verifies inn configuration and database files.
innd(1m)	The Sun Internet News Feeder daemon, which handles all NNTP feeds. The innd program has been split out into the Sun Internet News Reader program **snsd**(1m) and the Sun Internet News Feeder program **innd**(1m) **Note -** The news Feeder daemon **innd**(1m) is evolving and should *not* be executed directly via the command line or in-house scripts.
innstat(1m)	prints a snapshot of the INN system. It displays the operating mode of the server, as well as disk usage and the status of all log and lock files.
innwatch(1m)	monitors innd. Innwatch is normally started by rc.news. Every (600) seconds it examines the load average, the number of free blocks, and the inodes on the spool partition as described by its control file, innwatch.ctl(4).
innxbatch(1m)	sends xbatched Usenet articles to a remote NNTP server.
innxmit(1m)	sends Usenet articles to a remote NNTP server.
isppammod(1m)	configures PAM for LDAP authentication for News service.
makeactive(1m)	recovers Usenet active file.

makehistory(1m)	recovers Usenet history database.
news-recovery(1m)	The news-recovery programs and man pages have been split out into separate programs and pages: **makeactive**(1m), **makehistory**(1m), and **newsrequeue**(1m).
news.daily(1m)	performs a number of important Usenet administrative functions. This includes producing a status report, removing old news articles, processing log files, rotating the archived log files, renumbering the active file, removing any old socket files found in the ''firewall'' directory, and collecting the output.
newslog(1m)	This program and man page have been split out to **scanlogs**(1m), **writelog**(1m), **nnstat**(1m), **tally.unwanted**(1m), and **tally.control**(1m).
newsrequeue(1m)	used to rewrite batchfiles after a system crash.
nntpget(1m)	gets Usenet articles from a remote NNTP server.
nntpsend(1m)	invokes **innxmit**(1m) to send Usenet articles to a remote NNTP site.
overchan(1m)	updates the news overview database as articles come in.
prunehistory(1m)	removes file names from Usenet history file.
rnews(1m)	reads messages typically queued by a UUCP newsfeed and sends them to the local feeder daemon.
scanlogs(1m)	summarizes the information recorded in the INN log files (see **newslog**(4). By default, it also rotates and cleans out the logs. It is normally invoked by the **news.daily**(1m) script.
snsd(1m)	The Sun News Server daemon. It listens on the NNTP port for reader connections, and handles all incoming NNTP connections.

snsnews(1m)	Start and stop the Sun Internet News feeder daemon **innd**(1m) and the Sun Internet News reader daemon **snsd**(1m). **Note -** The feeder daemon and reader daemon should be started only via **snsnews**(1m).
tally.control(1m)	keeps track of newsgroup creations and deletions, and updates the cumulative list of newsgroup creations and deletions, control.log.
tally.unwanted(1m)	keeps track of unwanted newsgroups, and updates the cumulative list of articles posted to unwanted newsgroups, `unwanted.log`.
tally.unwanted(1m)	keeps track of unwanted newsgroups, and updates the cumulative list of articles posted to unwanted newsgroups, `unwanted.log`.
writelog(1m)	used to write a log entry or send it as mail.

NAME

archive – archives Usenet articles

SYNOPSIS

archive [–a *archive*] [–f] [–i *index*] [–m] [–r] [*input*]

DESCRIPTION

Archive makes copies of files specified on its standard input. It is normally run either as a channel feed under the feeder daemon, or by a script before **expire**(1m) is run.

Archive reads the named *input* file, or standard input if no file is given. The input is taken as a set of lines. Blank lines and lines starting with a number sign (‘‘#’’) are ignored. All other lines should specify the name of a file to archive. If a filename is not an absolute pathname, it is taken to be relative to `/var/news/storage/articles`.

Files are copied to a directory within the archive directory, `/var/news/storage/archive`. The default is to create a hierarchy that mimics the input files; intermediate directories will be created as needed. For example, the input file `comp/sources/unix/2211` (article 2211 in the newsgroup comp.sources.unix) will be copied to `/var/news/storage/archive/comp/sources/unix/2211`.

OPTIONS

–a `archive`	Specifies the directory in which to archive instead of the default.
–f	Flatten all directory names, replacing the slashes with periods. In this case, the file would be copied to `/var/news/storage/archive/comp.sources.unix/2211`.
–i	Append one line to the specified *index* file for each article that it copies. This line will contain the destination name and the Message-ID and Subject headers.
–m	Files are copied by making a link. If that fails a new file is created. If the ‘‘–m’’ flag is used, then the file will be copied to the destination, and the input file will be replaced with a symbolic link pointing to the new file. The ‘‘–m’’ flag is ignored.
–r	By default, *archive* sets its standard error to `/var/news/logs/errlog`. To suppress this redirection, use the ‘‘–r’’ flag.

EXIT STATUS

0	success
1	failure due to one of the following reasons:

- usage error
- Can’t open one of the files for input
- Can’t open file for ouput
- Can’t cd to spool directory
- Can’t spool to file
- Can’t start spool
- Can’t write spool
- Can’t rename spool

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

ctlinnd(1m), newsfeeds(4), snsnews(1m)

NAME

batcher – article-batching backend for Sun Internet News Server

SYNOPSIS

batcher [–a *arts*] [–A *total_arts*] [–b *size*] [–B *total_size*] [–i *string*] [–N *num_batches*] [–p *process*] [–r] [–s *separator*] [–S *alt_spool*] [–v] *host* [*input*]

DESCRIPTION

Batcher read uses a list of files to prepare news batches for the specified *host*. It is normally invoked by a script run out of `cron`(1m) that uses *shlock*(1) to lock the host name, followed by a *ctlinnd*(1m) command to flush the batch file.

Batcher reads the named *input* file, or standard input if no file is given. Relative pathnames are interpreted from the */var/news/storage/out.going* directory. The input is taken as a set of lines. Blank lines and lines starting with a number sign (‘‘#’’) are ignored. All other lines should consist of one or two fields separated by a single space. The first field is the name of a file holding an article; if it is not an an absolute pathname it is taken relative to the news spool directory, */var/news/storage/articles*. The second field, if present, specifies the size of the article in bytes.

OPTIONS

–S Specifies an alternate spool directory to use if the article is not found which would normally be an NFS-mounted spool directory of a master server with longer expiration times.

–r By default, the program sets its standard error to */var/news/logs/errlog*. To suppress this redirection, use the ‘‘–r’’ flag.

–v Upon exit, *batcher* reports statistics via *syslog*(3). If the ‘‘–v’’ flag is used, they will also be printed on the standard output.

–b *Batcher* collects the text of the named articles into batches. To limit the size of each batch, use the ‘‘–b’’ flag. The default size is 60 kilobytes. Using ‘‘–b0’’ allows unlimited batch sizes.

–a Limits the number of articles in each batch. The default is no limit. A new batch will be started when either the byte count or number of articles written exceeds the specified limits.

–B Limits the total number of bytes written for all batches.

–A Limits the total number of articles that can be batched.

–N Limits the total number of batches that should be created.

In all three cases, the default is zero, which is taken to mean no limit.

`–i string` A batch starts with an identifying line to specify the unpacking method to be used on the receiving end. When the ‘‘–i’’ flag is used, the initial string, *string*, followed by a newline, will be output at the start of every batch. The default is to have no initial string.

`–s` Each article starts with a separator line to indicate the size of the article. To specify the separator use the ‘‘–s’’ flag. This is a *sprintf*(3) format string which can have a single ‘‘%ld’’ parameter which will be given the size of the article. If the separator is not empty, then the string and a newline will be output before every article. The default separator is ‘‘#! rnews %ld’’.

`–p` By default, batches are written to standard output, which is not useful when more than one output batch is created. Use the ‘‘–p’’ flag to specify the shell command that should be created (via *popen*(3)) whenever a new batch is started. The process is a *sprintf* format string which can have a single ‘‘%s’’ parameter which will be given the host name. A common value is:

```
( echo '#! cunbatch' ; exec compress ) | uux -- --r --z %s!rnews
```

EXIT STATUS

If the input is exhausted, *batcher* will exit with a zero status. If any of the limits specified with the ‘‘–B,’’ ‘‘–A,’’ or ‘‘–N’’ flags is reached, or if there is an error writing the batch, then *batcher* will try to spool the input, copying it to a file. If there was no input filename, the standard input will be copied to */var/news/storage/out.going/*host and the program will exit. If an input filename was given, a temporary file named *input*.bch (if *input* is an absolute pathname) or */var/news/storage/out.going/*input.bch (if the filename does not begin with a slash) is created. Once the input is copied, *batcher* will try to rename this temporary file to be the name of the input file, and then exit.

Upon receipt of an interrupt or termination signal, *batcher* will finish sending the current article, close the batch, and then rewrite the batch file according as described in the previous paragraph.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO ctlinnd(1m), newsfeeds(4), shlock(1).

NAME

buffchan – buffered file-writing backend for News

SYNOPSIS

buffchan [–b] [–c *lines*] [–C *seconds*] [–d *directory*] [–f *fields*] [–m *map*] [–p *pidfile*] [–l *lines*] [–L *seconds*] [–r] [–s *file_format*] [–u]

DESCRIPTION

Buffchan reads lines from standard input and copies certain fields in each line into files named by other fields within the line. *Buffchan* is intended to be called by the feeder daemon as an exploder feed.

OPTIONS

–b Once *buffchan* opens a file it keeps it open. The input must therefore never specify more files than the number of available descriptors can keep open. If the ‘‘–b’’ flag is used, the program will allocate a buffer and attach it to the file using *setbuf*(3).

–c If the ‘‘–c’’ flag is used with a number *n*, then *buffchan* will close and reopen a file after every *n* lines are written to a file.

–C Used to specify that all files should be closed and reopened every *n* seconds.

–d Used to specify a directory the program should change to before starting. If this flag is used, then the default for the ‘‘–s’’ flag is changed to be a simple ‘‘%s.’’

–f Buffchan input is interpreted as a set of lines. Each line contains a fixed number of initial fields, followed by a variable number of filename fields. All fields in a line are separated by white space. The default number of initial fields is one; the ‘‘–f’’ flag may be used to specify a different number of fields. See *filechan*(1m) for an example.

–p If the ‘‘–p’’ flag is used, the program will write a line containing its process ID (in text) to the specified file.

–l If the ‘‘–l’’ flag is used with a number *n*, then *buffchan* will call *fflush*(3) after every *n* lines are written to a file.

–L If the ‘‘–L’’ flag is used with a number *n*, then all files will be flushed every *n* seconds.

–r By default, the program sets its standard error to */var/news/logs/errlog*. To suppress this redirection, use the ‘‘–r’’ flag.

–s After the initial fields, each remaining field names a file to write. The ‘‘–s’’ flag may be used to specify a format string that maps the field to

a file name. This is a *sprintf*(3) format string which should have a single ‘‘%s’’ parameter which will be given the field. The default value is */var/news/storage/out.going/%*s. See the description of this flag in *filechan*(1m).

`–u` If the ‘‘–u’’ flag is used, the program will request unbuffered output.

Buffchan can be invoked as an exploder feed (see *newsfeeds*(4)). As such, if a line starts with an exclamation point it will be treated as a command. There are three commands, described below:

`flush` The ‘‘flush’’ command closes and reopens all open files. ‘‘flush xxx’’ which flushes only the specified site "xxx". These are analogous to the *ctlinnd*(1m) ‘‘flush’’ command, and can be achieved by doing a ‘‘send "flush xxx"’’ command. Applications can tell that the ‘‘flush’’ has completed by renaming the file before issuing the command; *buffchan* has completed the command when the original filename reappears.

Buffchan also changes the access permissions of the file from read-only for everyone to read-write for owner and group as it flushes or closes each output file. It will change the modes back to read-only if it reopens the same file.

`drop` The ‘‘drop’’ command is similar to the ‘‘flush’’ command except that any files are not reopened. If given an argument, then the specified site is dropped, otherwise all sites are dropped. (Note that the site will be restarted if the input stream mentions the site.) When a *ctlinnd* ‘‘drop site’’ command is sent, the feeder daemon will automatically forward the command to *buffchan* if the site is a funnel that feeds into this exploder. To drop all sites, use the *ctlinnd* ‘‘send buffchan-site drop’’ command.

`readmap` The map file (specified with the ‘‘–m’’ flag) is reloaded.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO snsnews(1m), ctlinnd(1m), filechan(1m), newsfeeds(4).

NAME

crosspost – creates the links for cross-posted articles

SYNOPSIS

crosspost [–D *dir*] [–s] [*file...*]

DESCRIPTION

Crosspost reads group and article number data from *files* or standard input if none are specified. (A single dash in the file list means to read standard input.) It uses this information to create the hard, or symbolic, links for cross-posted articles. *Crosspost* is designed to be used by InterNetNews to create the links as the articles come in. The feeder daemon normally creates the links but by having *crosspost* create the links, less time is spent waiting for disk IO.

Crosspost expects input with one line per article in the form:

group.name/123 group2.name/456 group3.name/789

with one line per article. Any dots in the input are translated into "/" to translate the news group into a pathname. The first field is assumed to be the name of an existing copy of the article. *Crosspost* will attempt to link all the subsequent entries to the first using hard links if possible or symbolic links if that fails.

By default, *crosspost* processes its input as an INN channel feed written as a ''WR'' entry in the *newsfeeds*(4) file, for example:

crosspost:*:Tc,Ap,WR:/usr/news/bin/crosspost

To process the history file and recreate all the links for all articles use:

awk <history -F' ' '(NF > 2){print $3}' | crosspost

where the -F is followed by a tab character.

The ''–D'' flag can be used to specify where the article spool is stored. The default directory is */var/news/storage/articles*.

By default *crosspost* will fsync(2) each article after updating the links. The ''–s'' flag can be used to prevent this.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO snsnews(1m), newsfeeds(4),

NAME

ctlinnd – controls the Sun Internet News Server feeder daemon

SYNOPSIS

ctlinnd [–h] [–s] [–t *timeout*] *command* [*argument...*]

DESCRIPTION

Ctlinnd sends a message to the control channel of the Sun Internet News Feeder daemon.

In the normal mode of behavior, the message is sent to the server, which then performs the requested action and sends back a reply with a text message and the exit code for *ctlinnd*. If the server successfully performed the command, *ctlinnd* will exit with a status of zero and print the reply on standard output. If the server could not perform the command (for example, it was told to remove a newsgroup that does not exist), it will direct *ctlinnd* to exit with a status of one. The ‘‘shutdown,’’ ‘‘xabort,’’ and ‘‘xexec’’ commands do not generate a reply; *ctlinnd* will always exit silently with a status of zero.

OPTIONS

–s No message will be printed if the command was successful.

–t Specifies how long to wait for the reply from the server. The timeout value specifies the number of seconds to wait. A value of zero waits forever, and a value less than zero indicates that no reply is needed. When waiting for a reply, *ctlinnd* will try every two minutes to see if the server is still running, so it is unlikely that ‘‘--t0’’ will hang. The default is ‘‘–t0.’’

–h To see a command summary, use the ‘‘–h’’ flag. If a command is included when *ctlinnd* is invoked with the ‘‘–h’’ flag, then only the usage for that command will be given.

If a large number of groups are going to be created or deleted at once, it may be more efficient to ‘‘pause’’ or ‘‘throttle’’ the server and edit the *active* file directly.

The complete list of commands follows. Note that all commands have a fixed number of arguments. If a parameter can be an empty string, then it is necessary to specify it as two adjacent quotes, like "".

addhist <***Message-ID***> ***arr exp post paths***

Add an entry to the history database. This directs the server to create a history line for *Message-ID*. The angle brackets are optional. *Arr*, *exp*, and *post* specify when the article arrived, what its expiration date is, and when it was posted. All three values are a number indicating the number of seconds since the epoch. If the article does not have an Expires header, then *exp* should be zero. *Paths* is the pathname within the news spool directory where the article is filed. If the article is cross-posted, then the names should

be separated by white space and the *paths* argument should be inside double quotes. If the server is paused or throttled, this command causes it to briefly open the history database.

`allow` ***reason***

Remote connections are allowed. The *reason* must be the same text given with an earlier ‘‘reject’’ command, or an empty string.

`begin` ***site***

Begins feeding *site*. This will cause the server to rescan the *newsfeeds*(4) file to find the specified site and set up a newsfeed for it. If the site already exists, a ‘‘drop’’ is done first. This command is forwarded.

`cancel` <***Message-ID***>

Removes the article with the specified Message-ID from the local system. This does *not* generate a cancel message. The angle brackets are optional. If the server is paused or throttled, this command causes it to briefly open the history database.

`changegroup` ***group rest***

The newsgroup *group* is changed so that its fourth field in the *active* file becomes the value specified by the *rest* parameter. This may be used to make an existing group moderated or unmoderated, for example.

`checkfile`

Checks the syntax of the *newsfeeds* file, and display a message if any errors are found. The details of the errors are reported to *syslog*(3).

`drop` ***site***

Flushes and drops *site* from the server’s list of active feeds. This command is forwarded.

`feedinfo` ***site***

Prints detailed information about the state of the feed to *site* or more brief status of all feeds if *site* is an empty string.

`tcl` ***flag***

Enables or disables Tcl news filtering. If *flag* starts with the letter ''y'' then filtering is enabled. If it starts with ''n'', then filtering is disabled.

`feedinfo` ***site***

Prints detailed information about the state of the feed to *site* or more brief status of all feeds if *site* is an empty string.

`flush` ***site***

Flushes the buffer for the specified site. The actions taken depend on the type of feed the site receives; see *newsfeeds*(4). This is useful when the site is fed by a file and batching is going to start. If *site* is an empty string, then all sites are flushed and the *active* file and history databases are also written out. This command is forwarded.

`flushlogs`

Closes the log and error log files and renames them to have a *.old* extension. The history database and *active* file are also written out.

`go` ***reason***

Re-opens the history database and starts accepting articles after a ''pause'' or ''throttle'' command. The *reason* must either be an empty string or match the text that was given in the earlier ''pause'' or ''throttle'' command. If a ''reject'' command was done, this will also do an ''allow'' command if the *reason* matches the text that was given in the ''reject.'' If a ''reserve'' command was done, this will also clear the reservation if the *reason* matches the text that was given in the ''reserve.'' Note that if only the history database has changed while the server is paused or throttled, it is not necessary to send it a ''reload'' command before sending it a ''go'' command. If the server throttled itself because it accumulated too many I/O errors, this command will reset the error count. If the server was not started

with the ‘‘`--ny`’’ flag, then this command also does a ‘‘readers’’ command with ‘‘yes’’ as the flag and *reason* as the text.

`hangup` ***channel***

Closes the socket on the specified incoming channel. This is useful when an incoming connection appears to be hung.

`help` ***[command]***

Prints a command summary for all commands, or just `command` if specified.

`logmode`

Causes the server to log its current operating mode to *syslog*.

`mode`

Prints the server’s operating mode as a multi-line summary of the parameters and operating state.

`name` ***nnn***

Prints the name of channel number *nnn* or of all channels if it is an empty string.

`newgroup` ***group rest creator***

Creates the specified newsgroup. The *rest* parameter should be the fourth field as described in *active*(4); if it is not an equal sign, only the first letter is used. The *creator* should be the name of the person creating the group. If the newsgroup already exists, this is equivalent to the ‘‘changegroup’’ command. This is the only command that has defaults. The *creator* can be omitted and will default to the empty string, and the *rest* parameter can be omitted and will default to ‘‘y’’. This command can be done while the server is paused or throttled; it will update its internal state when a ‘‘go’’ command is sent. This command updates the *active.times* (see *active*(4)) file.

`param` ***letter value***

Changes the command-line parameters of the server. The combination of defaults make it possible to use the text of the Control header directly. *Letter* is the feeder daemon command-line option to set, and *value* is the new value. For example, ‘‘i 5’’ directs the server to allow only five incoming connections. To enable or disable the action of the ‘‘–n’’ flag, use the letter ‘‘y’’ or ‘‘n’’, respectively, for the *value.*

`pause` ***reason***

Pauses the server so that no incoming articles are accepted. No existing connections are closed, but the history database is closed. This command should be used for short-term locks, such as when replacing the history files. If the server was not started with the ‘‘`--ny`’’ flag, then this command also does a ‘‘readers’’ command with ‘‘no’’ as the flag and *reason* as the text.

`reject` ***reason***

Remote connections (those that would not be handed off to *snsd*) are rejected, with *reason* given as the explanation.

`reload` ***what reason***

The server updates its in-memory copies of various configuration files. *What* identifies what should be reloaded. If it is an empty string or the word ‘‘all’’ then everything is reloaded; if it is the word ‘‘history’’ then the history database is closed and opened, if it is the word ‘‘hosts.nntp’’ then the **`hosts.nntp`**(4) file is reloaded; if it is the word ‘‘active’’ or ‘‘newsfeeds’’ then both the *active* and *newsfeeds* files are reloaded; if it is the word ‘‘overview.fmt’’ then the **`overview.fmt`**(4) file is reloaded. If it is the word ‘‘filter.tcl’’ then the *filter.tcl* file is reloaded. If a TCL procedure named ‘‘filter_before_reload’’ exists, it will be called prior to rereading *filter.tcl.* If a TCL procedure named ‘‘filter_after_reload’’ exists, it will be called after *filter.tcl* has been reloaded. Reloading the Tcl filter does not enable filtering if it is disabled; use *filter* to do this. The *reason* is reported to *syslog*. There is no way to reload the data **`inn.conf`**(4) file; the server currently only uses the ‘‘pathhost’’ parameter, so this restriction should not be a problem. The *startup.tcl* file cannot be reloaded.

`renumber` ***group***

Scans the spool directory for the specified newsgroup and update the low-water mark in the *active* file. If *group* is an empty string then all newsgroups are scanned.

`reserve` ***reason***

The next ''pause'' or ''throttle'' command must use *reason* as its text. This ''reservation'' is cleared by giving an empty string for the *reason*. This command is used by programs like **`expire`**(1m) that want to avoid running into other instances of each other.

`rmgroup` ***group***

Removes the specified newsgroup. This is done by editing the *active* file. The spool directory is not touched, and any articles in the group will be expired using the default expiration parameters. Unlike the ''newgroup'' command, this command does not update the *active.times* file.

`send` ***feed text...***

The specified *text* is sent as a control line to the exploder *feed*.

`shutdown` ***reason***

The server is shut down, with the specified reason recorded in the log and sent to all open connections. It is a good idea to send a ''throttle'' command first.

`signal` ***sig site***

Signal *sig* is sent to the specified *site,* which must be a channel or exploder feed. *Sig* can be a numeric signal number or the word ''hup,'' ''int,'' or ''term''; case is not significant.

`throttle` ***reason***

Input is throttled so that all existing connections are closed and new connections are rejected. The history database is closed. This should be used for long-term locks, such as when *expire* is being run. If the server was not

started with the ‘‘–ny’’ flag, then this command also does a ‘‘readers’’ command with ‘‘no’’ as the flag and *reason* as the text.

`xabort` ***reason***

The server logs the specified *reason* and then invokes the **abort**(3) routine.

`xexec` ***path***

The server gets ready to shut itself down, but instead of exiting it executes the specified *path* with all of its original arguments. If *path* is ‘‘innd’’ then `/opt/SUNWsns/lib/innd` is invoked; if it is ‘‘inndstart’’ then `/opt/SUNWsns/lib/inndstart` is invoked; if it is an empty string, it will invoke the appropriate program depending on whether or not it was started with the ‘‘–p’’ flag; any other value is an error.

In addition to being acted upon within the server, certain commands can be forwarded to the appropriate child process. If the site receiving the command is an exploder (such as **buffchan**(1m)) or it is a funnel that feeds into an exploder, then the command can be forwarded. In this case, the server will send a command line to the exploder that consists of the *ctlinnd* command name. If the site funnels into an exploder that has an asterisk (‘‘*’’) in its ‘‘W’’ flag (see **newsfeed**(4)) , then the site name will be appended to the command; otherwise no argument is appended.

BUGS

Ctlinnd is limited to server replies no larger than 4k.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Evolving

SEE ALSO

active(4), expire(1m), inn.conf(4), newsfeeds(4), overview.fmt(4), snsnews(1m)

NAME

cvtbatch – converts Usenet batch file to INN format

SYNOPSIS

cvtbatch [–w *items*]

DESCRIPTION

Cvtbatch reads standard input as a series of lines, converts each line, and writes it to standard output. It is used to convert simple batchfiles that contain just the article name to INN batchfiles that contain additional information about each article.

Each line is taken as the pathname to a Usenet article. If it is not an absolute pathname, it is taken relative to the spool directory, */var/news/storage/articles*. (Only the first word of each line is parsed; anything following white space is ignored.)

OPTIONS

–w Specifies how each output line should be written. The items for this flag should be chosen from the ‘‘W’’ flag items as specified in *newsfeeds*(4). They may be chosen from the following set:

```
b Size of article in bytes
f full pathname of article
m article message-id
n relative pathname of article
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

newsfeeds(4).

NAME

expire – Usenet article and history expiration program

SYNOPSIS

expire [–d *dir*] [–e] [–f *file*] [–g *file*] [–h *file*] [–i] [–l] [–n] [–p] [–q] [–r *reason*] [–s] [–t] [–v *level*] [–w *number*] [–x] [–z *file*] [*expire.ctl*]

DESCRIPTION

Expire scans the `history`(4) text file */var/news/state/history* and uses the information recorded in it to purge old news articles.

OPTIONS

`–d` Create the new history file and database in the specified directory, *dir*. This is useful when the file system does not have sufficient space to hold both the old and new history files. When this flag is used, *expire* leaves the server paused and creates a zero-length file named after the new history file, with an extension of ‘‘.done’’ to indicate that it has successfully completed the expiration. The calling script should install the new history file and un-pause the server. The ‘‘`–r`’’ flag should be used with this flag.

`–e` If the ‘‘–e’’ flag is used, then as soon as the first cross posting of the article expires, all copies of it are removed.

`–f` Specifies an alternate history file.

`–g` Append a one-line summary equivalent to the output of ‘‘–v1’’ and preceeded by the current time to the specified `file`.

`–h` Specifies an alternate input text history file. *Expire* uses the old database to determine the size of the new one.

`–i` Ignore the old database.

`–l` *Expire* normally just unlinks each file if it should be expired. If the ‘‘`–l`’’ flag is used, then all articles after the first one are treated as if they could be symbolic links to the first one. In this case, the first article will not be removed as long as any other cross-posts of the article remain.

`–n` If the feeder daemon is not running, use the ‘‘`–n`’’ flag and *expire* will not send the ‘‘pause’’ or ‘‘go’’ commands. (For more details on the commands, see *ctlinnd*(1m)). Note that *expire* only needs exclusive access for a very short time — long enough to see if any new articles arrived since it first hit the end of the file, and to rename the new files to the working files.

`–p` *Expire* makes its decisions on the time the article arrived, as found in the `history` file. Articles are therefore kept a little longer than with other expiration programs that base their decisions on the article's posting date. To use the article's posting date, use the ''–p'' flag.

`–q` *Expire* normally complains about articles that are posted to newsgroups not mentioned in the active file. To suppress this action, use the ''–q'' flag.

`–r` *Expire* normally sends a ''pause'' command to the local feeder daemon when it needs exclusive access to the history file, using the string ''Expiring'' as the reason. To give a different reason, use the ''–r'' flag. The process ID will be appended to the reason. When *expire* is finished and the new history file is ready, it sends a ''go'' command.

`–s` If the ''–s'' flag is used, then *expire* will print a summary when it exits showing the approximate number of kilobytes used by all deleted articles.

`–t` If the ''–t'' flag is used, then *expire* will generate a list of the files that should be removed on its standard output, and the new history file will be left in *history.n* and *history.n.dir* and *history.n.pag*. This flag is useful for debugging when used with the ''–n'' and ''–s'' flags. Note that if the ''–f'' flag is used, then the name specified with that flag will be used instead of `history`.

`–v` The ''–v'' flag is used to increase the verbosity of the program, generating messages to standard output. The *level* should be a number, where higher numbers result in more output. Level one will print totals of the various actions done (not valid if a new history file is not written), level two will print a report on each individual file, while level five results in more than one line of output for every line processed.

`–w` Use the ''–w'' flag to ''warp'' time so that *expire* thinks it is running at some time other then the current time. The value should be a signed floating point number of the number of days to use as the offset.

`–x` If the ''–x'' flag is used, then *expire* will not create any new history files. This is most useful when combined with the ''–n'', "–s'', and "–t'' flags to see how different expiration policies would change the amount of disk space used.

`–z` If the ‘‘`–z`’’ flag is used, then articles are not removed, but their names are appended to the specified `file`. See the description of *expirerm* in *news.daily*(1m).

If a filename is specified, it is taken as the control file and parsed according to the rules in *expire.ctl*(4). A single dash (‘‘–’’) may be used to read the file from standard input. If no file is specified, the file */var/news/config/expire.ctl* is read.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

ctlinnd(1m), expire.ctl(4), history(4), snsnews(1m)

NAME

expireover – expires entries from the news overview database

SYNOPSIS

expireover [–a] [–D *overviewdir*] [–f *file*] [–n] [–O *overview.fmt*] [–s] [–v] [–z] [*file...*]

DESCRIPTION

Expireover expires entries from the news overview database. It reads a list of pathnames (relative to the spool directory, */var/news/storage/articles*), from the specified *files* or standard input if none are specified. (A file name of ‘‘–’’ may be used to specify the standard input.) It then removes any mention of those articles from the appropriate overview database.

OPTIONS

–z If the ‘‘–z’’ flag is used, then the input is assumed to be sorted such that all entries for a newsgroup appear together so that it can be purged at once. This flag can be useful when used with the sorted output of *expire*(1m)’s ‘‘–z’’ flag.

–s If the ‘‘–s’’ flag is used, then *expireover* will read the spool directory for all groups mentioned in the *active*(4) file, and remove the overview entries of any articles that do not appear in the directory.

–f Specifies an alternate file. A name of ‘‘-’’ is taken to mean the standard input.

–a Reads the spool directory and adds any missing overview entries. It will create files if necessary. This can be used to initialize a database, or to sync up a overview database that may be lacking articles due to a crash. *Overchan* should be running to ensure that any incoming articles get included. Using this flag implies the ‘‘–s’’ flag; the ‘‘–f’’ flag may be used to add only a subset of the newsgroups.

–v List the entries that would be added or deleted.

–n Do not perform any real updates.

–D Specifies where the databases are stored. The default directory is */var/news/storage/over.view*.

–O Used to specify an alternate location for the *overview.fmt*(4) file; this is normally only useful for debugging.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO expire(1m), overview.fmt(4).

NAME

expirerm – removes articles that have been expired.

SYNOPSIS

expirerm *file*

DESCRIPTION

Expirerm is a script that removes a list of files. The specified `file` lists the files. It is sorted and then fed into a pipeline responsible for doing the removal, normally *fastrm*(1m). If there is a problem removing the files, then mail is sent to the news administrator. If there were no problems, then `file` is renamed to */var/news/logs/expire.list* where it is kept for safety until the next day's expiration.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

expire(1m), fastrm(1m)

NAME

fastrm – quickly removes a set of files

SYNOPSIS

fastrm [–d] [–e] [–u *N*] [–s *M*] [–c *I*] *base_directory*

DESCRIPTION

Fastrm reads a list of files, one per line, from its standard input and removes them. If a file is not an absolute pathname, it is taken relative to the directory specified on the command line. The *base_directory* parameter must be a simple absolute pathname, that is, it must not contain any ‘‘/./’’ or ‘‘/../’’ references.

Fastrm is designed to be faster than the typical ‘‘| xargs rm’’ pipeline. For example, *fastrm* will usually `chdir`(2) into a directory before removing files from it. If the input is sorted, this means that most files to be removed are simple names.

Fastrm assumes that its input is valid and that it is safe to just do an `unlink`(2) call for each item to be removed. As a safety measure, if *fastrm* is run by root it will first *stat*(2) the item to make sure that it is not a directory before unlinking it.

OPTIONS

–d If the ‘‘–d’’ flag is used then no files are removed. Instead a list of the files to be removed, in debug form, is printed on the standard output. Each line contains either the current directory of *fastrm* at the time it would do the unlink, and then the path name it would pass to `unlink`(2) as two fields separated by white space and a ‘‘/’’, or the absolute path name (a single field) of files it would unlink using the absolute path name.

–e If the ‘‘–e’’ flag is used, *fastrm* will treat an empty input file (`stdin`) as an error. This is most useful when *fastrm* is last in a pipeline after a preceding **`sort`**(1) as if the sort fails, there will usually be no output to become input of *fastrm*.

–u If the ‘‘–u’’ flag is used, then *fastrm* makes further assumptions about its work environment; in particular, that there are no symbolic links in the target tree. This flag also suggests that it is probably faster to reference the path ‘‘../../../’’ rather than start from the root and come down. (Note that this probably isn’t true on systems that have a namei cache, which usually holds everything except ‘‘..’’). The optional *N* is an integer that specifies the maximum number of ‘‘..’’ segments to use — paths that would use more than this use the absolute path name (from the root) instead. If the ‘‘–u’’ flag is given without a value, ‘‘–u1’’ is assumed.

–s If the ‘‘–s’’ flag is used, then *fastrm* will perform the unlinks from one directory, that is when a group of files in one directory appears in the

input consecutively in the order that the files appear in the directory from which they are to be removed. The intent of this flag is that on systems that have a per-process directory cache, finding files in the directory should be faster. It can have smaller benefits on other systems. The optional *M* is an integer that specifies the number of files that must be going to be removed from one directory before the files will be ordered. If the ''–s'' flag is given without a value, ''–s5'' is assumed. When the directory reordering is in use *fastrm* will avoid attempting to unlink files that it can't see in the directory, which can speed it appreciably when many of the file names have already been removed.

–c The ''–c'' flag may be given to instruct *fastrm* when it should chdir(2). If the number of files to be unlinked from a directory is at least *I* then *fastrm* will change directory to and unlink the files from the directory. Otherwise it will build a path relative to its current directory. If ''–c'' is given without the optional integer *I* then ''–c1'' is assumed, which will cause *fastrm* to always use chdir. If ''–c'' is not used at all, then ''–c3'' is assumed. Use ''–c0'' to prevent *fastrm* from ever using chdir(2).

–a –r The ''–a'' and ''–r'' options do nothing at all, except allow you to say ''fastrm –usa'' ''fastrm –ussr'' or ''fastrm –user''. These happen to often be convenient sets of options to use.

EXIT STATUS

Fastrm exits with a status of zero if there were no problems, or one if something went wrong. Attempting to remove a file that does not exist is not considered a problem. If the program exits with a non-zero status, it is probably a good idea to feed the list of files into an ''xargs rm'' pipeline.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Evolving

NAME

filechan – file-writing backend for InterNetNews

SYNOPSIS

filechan [–d *directory*] [–f *fields*] [–m *mapfile*] [–p *pidfile*]

DESCRIPTION

Filechan reads lines from standard input and copies certain fields in each line into files named by other fields within the line. *Filechan* is intended to be called by the feeder daemon as a channel feed. (It is not a full exploder and does not accept commands; see *newsfeeds*(4) for a description of the difference and *buffchan*(1m) for an exploder program.)

Filechan input is interpreted as a set of lines. Each line contains a fixed number of initial fields, followed by a variable number of filename fields. All fields in a line are separated by white space. The default number of initial fields is one.

For each line of input, *filechan* writes the initial fields, separated by white space and followed by a newline, to each of the files named in the filename fields. When writing to a file, *filechan* opens it in append mode and tries to lock it and change the ownership to the user and group who owns the directory where the file is being written.

OPTIONS

–f Specifies a different number of fields.

–d By default, *filechan* writes its arguments into the directory */var/news/storage/out.going*. The ‘‘–d’’ flag may be used to specify a directory the program should change to before starting.

–p If the ‘‘–p’’ flag is used, the program will write a line containing its process ID (in text) to the specified file.

If *filechan* is invoked with ‘‘–f 2’’ and given the following input:

```
news/software/b/132 <1643@munnari.oz.au> foo uunet
news/software/b/133 <102060@litchi.foo.com> uunet munnari
comp/sources/unix/2002 <999@news.foo.com> foo uunet munnari
```

Then the file *foo* will have these lines:

```
news/software/b/132 <1643@munnari.oz.au>
comp/sources/unix/2002 <999@news.foo.com>
```

The file *munnari* will have these lines:

```
news/software/b/133 <102060@litchi.foo.com>
comp/sources/unix/2002 <999@news.foo.com>
```

The file *uunet* will have these lines:

```
news/software/b/132 <1643@munnari.oz.au>
news/software/b/133 <102060@litchi.foo.com>
comp/sources/unix/2002 <999@news.foo.com>
```

Because the time window in which a file is open is very small, complicated flushing and locking protocols are not needed; a `mv`(1) followed by a `sleep`(1) for a couple of seconds is sufficient.

`-m` Specifies a map file by using the ‘‘`-m`’’ flag. Blank lines and lines starting with a number sign (‘‘#’’) are ignored. All other lines should have two host names separated by a colon. The first field is the name that may appear in the input stream; the second field names the file to be used when the name in the first field appears. For example, the following map file may be used to map the short names above to the full domain names:

```
# This is a comment
uunet:news.uu.net
foo:foo.com
munnari:munnari.oz.au
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

buffchan(1m), newsfeeds(4), snsnews(1m)

NAME

innccheck – checks feeder daemon configuration and database files.

SYNOPSIS

innccheck [–a] [–v] [–pedantic] [–f] [–perm] [–noperm][file=value | file]

DESCRIPTION

Innccheck examines various configuration files and databases and verifies things about them. Things verified depend on the file being checked, but generally are things like permissions, ownership, syntax errors in config files and so on.

Innccheck does not make changes to any files; it just reports what might be wrong. It is up to the operator to fix the problem.

The set of files checked may be restricted by using `file` or `file=value` arguments. For example, specifying `hosts.nntp` causes only the *hosts.nntp*(4) file to be checked. Using `hosts.nntp=/tmp/hosts.nntp.tst` on the command line will cause *innccheck* to only verify the hosts.nntp file, and it will perform the checks on the file /tmp/hosts.nntp file instead of the default one.

Valid values for `file` are:

```
active
control.ctl
expire.ctl
hosts.nntp
inn.conf
moderators
newsfeeds
overview.fmt
nnrp.access
nntpsend.ctl
passwd.nntp
```

OPTIONS

`–a` If any ‘‘`file`’’ value or ‘‘`file=value`’’ pairs are given, then normally only the files they refer to are checked. Use the ‘‘`–a`’’ flag to specify that *all* files should be checked regardless. In this case the form `file=value` will be the more useful.

`–v` Use the ‘‘`–v`’’ produce more verbose output.

`–pedantic` Use the ‘‘`–pedantic`” option to get reports on things that are not necessarily wrong, but may indicate a bad configuration — such as *inn.conf*(4) missing a key.

`–f` Prints the appropriate chown/chgrp/chmod command necessary to fix a problem that it reports. Any other output lines will be prefixed with a ‘‘#’’ character to generate valid

input for a shell. Note that the ‘‘–perm’’ flag must be used as well when using this flag.

–perm
Inncheck checks all files for permission problems. If the ‘‘–perm’’ flag is used, then *only* the files specified by the file or file=value command line arguments will be checked for problems other than permission problems.

–noperm
To avoid doing any checking of file permissions or ownership, use the ‘‘–noperm’’ option.

EXAMPLES

EXAMPLE 1 To have *inncheck* check all files for syntax and permission problems simply do:

```
inncheck
```

EXAMPLE 2 To have *inncheck* check all files for permission problems and to verify the syntax of the active and hosts.nntp files do:

```
inncheck -perm active hosts.nntp
```

EXAMPLE 3 To have *inncheck* check the test newsfeeds file in /var/tmp/newsfeeds.testing, do:

```
inncheck newsfeeds=/var/tmp/newsfeeds.testing
```

EXAMPLE 4 To have *inncheck* check all the files as it normally does, but to specify a different location for the newsfeeds file, do:

```
inncheck -a newsfeeds=/var/tmp/newsfeeds.testing
```

BUGS

If the ‘‘–f’’ and ‘‘–perm’’ options are used together, along with –a or some ‘‘file’’ or ‘‘file=value’’ arguments that refer to a file with a syntax problem, then the output will no longer be valid input for a shell.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO active(4), expire.ctl(4), history(4), hosts.nntp(4), inn.conf(4), newsfeeds(4), snsnews(1m)

NAME

innd – The Sun Internet News Server feeder daemon

SYNOPSIS

/opt/SUNWsns/lib/innd

DESCRIPTION

The Sun Internet News Feeder daemon is evolving and should not be executed directly.

Use **snsnews**(1M) to start and stop news servers.

SEE ALSO

ctlinnd(1m), snsd(1m), snsnews(1m)

NAME

innstat – prints snapshot of the INN system

SYNOPSIS

innstat

DESCRIPTION

The *innstat* script prints a snapshot of the INN system. It displays the operating mode of the server, as well as disk usage and the status of all log and lock files.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

snsnews(1m), news.daily(1m), newslog(4), snsd(1m)

NAME

innwatch – monitors the feeder daemon.

SYNOPSIS

innwatch [–`l` *log file*] [–`t` *seconds*]

DESCRIPTION

Innwatch is normally started by *rc.news*. Every *(600)* seconds it examines the load average, the number of free blocks, and the inodes on the spool partition as described by its control file, *innwatch.ctl*(4).

If the load gets too high or the disk gets too full, it throttles the news reader server. When the condition restores, it unblocks the news reader server. In addition, on each pass through the loop it will check the log file `/var/news/logs/news.crit` to see if it has been modified and sends mail to the news administrator if so.

Upon receipt of an interrupt signal (SIGINT), *innwatch* will report its status in the file */var/news/logs/innwatch.status*.

OPTIONS

–`l` Specifies a log file to watch, other than the default of *news.crit*.

–`t` Changes the period between check from the default.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

ctlinnd(1m), innwatch.ctl(4), shlock(1), snsnews(1m)

NAME

innxbatch – sends xbatched Usenet articles to a remote NNTP server

SYNOPSIS

innxbatch [–D] [–t *timeout*] [–T *timeout*] [–v] *host file...*

DESCRIPTION

Innxbatch connects to the NNTP server at the specified *host* and sends it the specified xbatch files, using the XBATCH extension to the NNTP protocol. It is normally invoked by a script run out of `cron`(1m) that uses *shlock*(1) to lock the host name, followed by a *ctlinnd*(1m) command to flush the batchfile.

Innxbatch normally blocks until the connection is made. To specify a timeout on how long to try to make the connection, use the ‘‘–t’’ flag. To specify the total amount of time that should be allowed for article transfers, use the ‘‘–T’’ flag. The default is to wait until an I/O error occurs, or all the articles have been transferred. If the ‘‘–T’’ flag is used, the time is checked just before an article is started; it will not abort a transfer that is in progress. Both values are measured in seconds.

Each file is removed after it has been successfully transferred.

If a communication error such as a `write`(2) failure, or an unexpected reply from the remote server occurs, *innxbatch* will stop sending and leave all remaining files untouched for later retry.

Upon exit, *innxbatch* reports transfer and CPU usage statistics via *syslog*(3). If the ‘‘–v’’ flag is used, they will also be printed on the standard output.

Use the ‘‘–D’’ flag to print debugging information on standard error. This will show the protocol transactions between *innxbatch* and the NNTP server on the remote host.

A sample *newsfeeds(4)* entry to produce appropriate xbatch files:

```
nase\
:*\
:Tc,Wnb\
:/var/news/storage/out.going$;/batcher \
  -p "(/var/news/storage/out.going$; > \
  /var/news/storage/out.going$;/nase.\$\$)" \
  nase.do.main
```

A sample script to invoke *innxbatch(1m)* is:

```
#!/bin/sh
## SH script to send xbatches for a site, wrapped around innxbatch
## Invocation:
##   sendxbatches.sh <sitename> <hostname> <xbatch file name> ...
if [ $# -le 3 ]
then
echo "usage: $0 <sitename> <hostname> <xbatch file name>"
exit 1
fi
## =()<. @<_PATH_SHELLVARS>@>()=
```

```
site="$1"; host="$2"; shift; shift

ctlinnd flush "$site" \
&& sleep 5 \
&& exec $NEWSBIN/innxbatch -v -D "$host" $*
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

innxmit(1m), nntpsend(1m), ctlinnd(1m), newsfeeds(4), shlock(1), snsnews(1m)

NAME

innxmit – sends Usenet articles to a remote NNTP server

SYNOPSIS

innxmit [–A *alt_spool*] [–a] [–c] [–d] [–l] [–M] [–r] [–s] [–t *timeout*] [–T *timeout*] [–p] [–S] [–P *portnum*] *host file*

DESCRIPTION

Innxmit connects to the NNTP server at the specified *host* and sends it the articles specified in the batch file named `file`. It is normally invoked by a script run out of `cron`(1m) that uses *shlock*(1) to lock the host name, followed by a *ctlinnd*(1m) command to flush the batch file.

If the `file` is not an absolute pathname, it is taken relative to the */var/news/storage/out.going* directory. It is normally written by specifying the ''Wnm'' flags in the *newsfeeds*(4) file. Each line in the batch file should be in one of the following formats:

```
filename Message-ID
filename
```

The *filename* field names the article to be sent. If it is not an absolute pathname it is taken relative to the news spool directory, */var/news/storage/articles*. If the *Message-ID* field is not specified, it will be obtained by scanning the article. The *filename* and *Message-Id* fields are separated by a space.

If a communication error such as a `write`(2) failure occurs, *innxmit* will stop sending and rewrite the batch file to contain the current article and any other unsent articles.

OPTIONS

`–t` *Innxmit* normally blocks until the connection is made. To specify a timeout on how long to try to make the connection, use the ''`–t`'' flag.

`–T` Specifies the total amount of time that should be allowed for article transfers. The default is to wait until an I/O error occurs, or all the articles have been transferred. If the ''`–T`'' flag is used, the time is checked just before an article is started; it will not abort a transfer that is in progress. Both values are measured in seconds.

`–P` Specifies a port number other than the default.

`–r` If the remote server sends an unexpected reply code, *innxmit* will requeue the article and proceed. Use the ''`–r`'' flag if the article should not be requeued.

`–v` Upon exit, *innxmit* reports transfer and CPU usage statistics via *syslog*(3). If the ‘‘`–v`’’ flag is used, they will also be printed on the standard output.

`–a` If all articles were sent successfully, *innxmit* will remove the batch file, otherwise it will rewrite it to contain the list of unsent articles. If no articles were sent or rejected, the file is left untouched. This can cause the batch file to grow excessively large if many articles have been expired and there are communication problems. To always rewrite the batch file, use the ‘‘`–a`’’ flag.

`–p` If the ‘‘`–p`’’ flag is given, then no connection is made and the batch file is purged of entries that refer to files that no longer exist. This implies the ‘‘`–a`’’ flag.

`–S` If the ‘‘`–S`’’ flag is given, then *innxmit* will offer articles to the specified host using the *‘‘xreplic’’* protocol. The ‘‘–S’’ flag implies ‘‘-s’’, since streaming is not supported in the xreplic protocol. To use this flag, the input file must contain the history data (commas are transliterated to spaces by the server). In order for this flag to be used, the input must contain the necessary history entries. This is usually done by setting up a ‘‘WnR’’ entry in the *newsfeeds* file.

`–d` Use the ‘‘`–d`’’ flag to print debugging information on standard error. This will show the protocol transactions between *innxmit* and the NNTP server on the remote host.

`–l` The ‘‘`–l`’’ flag is used to turn off logging of reasons the remote gives for rejecting an article.

`–M` If the ‘‘`–M`’’ flag is used, then *innxmit* will scan an article’s headers before sending it. If the article appears to be a MIME article that is not in seven-bit format, the article will be sent in ‘‘quoted-printable’’ form.

`–A` The ‘‘`–A`’’ flag may be used to specify an alternate spool directory to use if the article is not found; this would normally be an NFS-mounted spool directory of a master server with longer expiration times.

`–s` *Innxmit* will attempt to negotiate a streaming mode extension of the NNTP protocol with the server at connect time. If successful it will use a slightly different protocol that enhances throughput. If the server does not recognize the streaming mode negotiation *innxmit* will revert to normal NNTP transfer mode. Use the ‘‘`–s`’’ flag to disable the attempt to negotiate the streaming mode extension.

–c In streaming mode a check of each message ID is still made to avoid sending articles already on the server. The ‘‘–c’’ flag will, if streaming mode is supported, result in sending articles without checking. This results in slightly greater throughput and may be appropriate when it is known that the site could not already have the articles such as in the case of a "leaf" site.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

ctlinnd(1m), newsfeeds(4), shlock(1), snsnews(1m)

NAME

isppammod – configure PAM for LDAP authentication for News service

SYNOPSIS

isppammod –a *domain.name*

isppammod –d

isppammod –m *domain.name*

DESCRIPTION

The `isppammod` command provides an automated procedure for using PAM to authenticate News users through LDAP. The `isppammod` command creates a *domain.name* entry in *pam.conf* that specifies the use of */etc/opt/SUNWisp/lib/pam_ldap.so.1* for authentication. Note that *pam.conf* can contain only one *domain.name* for the news service.

OPTIONS

–a Add *domain.name* entry.

–d Delete the news domain entry.

–m Modify *domain.name* entry to reflect new *domain.name.*

EXIT STATUS

0 Successful completion.

>0 An error occured.

EXAMPLES

EXAMPLE 1 Add domain *allnews.com* to *pam.conf*

```
isppammod -a allnews.com
```

EXAMPLE 2 Delete the current domain from *pam.conf*

```
isppammod -d
```

EXAMPLE 3 Modify the existing *pam.conf* domain name entry to *main.news.com*

```
isppammod -m main.news.com
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO nnrp.access(4)

NAME

makeactive – recovers Usenet active file.

SYNOPSIS

makeactive [–m] [–o]

DESCRIPTION

Makeactive invokes `find`(1) to get a list of all directories in the news spool tree, `/var/news/storage/articles`. It discards directories named *lost+found* as well as those that have a period in them. It scans all other directories for all-numeric filenames and determines the highest and lowest number. The program's output is a set of **`active`**(4) file lines. Because there is no way to know if a group is moderated or disabled, the fourth field of all entries will be *y*. Also, mid-level directories that aren't newsgroups will also created as newsgroups with no entries (for example, there is a ''comp.sources.unix'' group, but no ''comp.sources'').

OPTIONS

`–m` Attempts to adjust the highest and lowest article numbers wherever possible. If articles are found in a newsgroup, the numbers will reflect what was found. If no articles are found in a newsgroup, the high number from the old file will be kept, and the low number will be set to one more then the high number. This flag may only be used if the ''`–o`'' flag is used.

`–o` Reads an existing *active* file for the list of group names and renumbers all groups. It will preserve the fourth field of the *active* file if one is present. This is analogous to the **`ctlinnd`**(1m) ''renumber'' command, except in this case the news feeder daemon should throttled using `ctlinnd throttle` or stopped using `snsnews stop`. Do not use this flag with output redirected to the standard active file.

EXIT STATUS

Makeactive exits with non-zero status if any problems occurred.

EXAMPLES

EXAMPLE 1 A typical way to use the program is with the following */bin/sh* commands:

```
ctlinnd throttle "Rebuilding active file"
TEMP=${TMPDIR-/var/tmp}/act$$
if [ --f /var/news/state/active; ] ; then
  if makeactive --o >${TEMP} ; then
    mv ${TEMP} /var/news/state/active;
  fi
else
  if makeactive >${TEMP} ; then
    # Edit to restore moderated
    # and aliased groups.
    ...
    mv ${TEMP} /var/news/state/active;
  fi
fi
ctlinnd reload active "New active file"
```

```
ctlinnd go ''
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

active(4), ctlinnd(1m), dbz(3), filechan(1m), history(4), newsfeeds(4), makehistory(1m), newsrequeue(1m), snsnews(1m)

NAME

makehistory – recovers Usenet history database.

SYNOPSIS

makehistory [–A *oldtmp*] [–a *active*] [–b] [–f *filename*] [–i] [–n] [–o] [–r] [–s *size*] [–T *tmpdir*] [–u[–v]]

DESCRIPTION

Makehistory rebuilds the **history**(4) text file and the associated **dbz**(3) database. The default name of the text file is `/var/news/state/history`; to specify a different name, use the ‘‘–f’’ flag. *Makehistory* scans the **active**(4) file to determine which newsgroup directories within the spool directory `/var/news/storage/articles` should be scanned. (If a group is removed, but its spool directory still exists, *makehistory* will ignore it.) The program reads each file found and writes a history line for it.

After the text file is written, *makehistory* will build the *dbz* database.

OPTIONS

–A Specifies the pathname to the directory that will be used by *makehistory* to store a copy of the history file as it’s being built. Existing data and valid history entries are appended to the history file.

–a Specifies the active file to use rather than the default one of `/var/news/state/active`.

–b Remove any articles that do not have valid Message-ID headers in them.

–f Specifies the database file name prefix, for example, specifying "-f file" results in the use of database files *file.dir* and *file.pag*. If the ‘‘–f’’ flag is not used, then a temporary link to the name *history.n* is made and the database files are written as *history.n.pag* and *history.n.dir*.

–o The link is not made and any existing history files are overwritten. If the old database exists, *makehistory* will use it to determine the size of the new database.

–i Ignore the old database. Using the ‘‘–o’’ flag implies the ‘‘–i’’ flag.

–s The program will also ignore any old database if the ‘‘–s’’ flag is used to specify the approximate number of entries in the new database. Accurately specifying the size is an optimization that will create a more efficient database. (The size should be the estimated eventual size of the file, typically the size of the old file.) For more information, see the discussion of *dbzfresh* and *dbzsize* in **dbz**(3).

–u Assumes that the feeder daemon is running (see **snsnews**(1m)), pauses the server while scanning, and then sends ‘‘addhist’’

commands (see **ctlinnd**(1m)) to the server for any article that is not found in the *dbz* database. The command ''makehistory –bu'' is useful after a system crash to delete any mangled articles and bring the article database back into a more consistent state.

`–v` If the ''`–v`'' flag is used with the ''`–u`'' flag, then *makehistory* will put a copy of all added lines on its standard output.

`–n` Scans the spool directory without rebuilding the *dbz* files. If used with ''`-u`'', the server will not be paused while scanning.

`–r` Builds the *dbz* files from an existing text file. The ''`–i`'' or ''`–s`'' flags can be useful if there are no valid *dbz* files to use.

`–T` *Makehistory* needs to create a temporary file that contains one line for each article it finds, which can become very large. This file is created in the */var/tmp* directory. The ''TMPDIR'' environment variable may be used to specify a different directory. Alternatively, the ''`–T`'' flag may be used to specify a temporary directory. In addition, the `sort`(1) that is invoked during the build writes large temporary files (often to `/var/tmp`; see your system man pages). If the ''`–T`'' flag is used, then the flag and its value will be passed to `sort`. On most systems this will change the temporary directory that `sort` uses. if used, this flag and its value will be passed on to the `sort`(1) command that is invoked during the build.

EXAMPLES

EXAMPLE 1 A typical way to use this program is with the following */bin/sh* commands:

```
ctlinnd throttle "Rebuilding history file"
cd /var/news/state
if makehistory --n --f history.n ; then
  :
else
  echo Error creating history file!
  exit 1
fi
# The following line can be used to retain expired history
# It is not necessary for the history file to be sorted.
# awk 'NF==2 { print; }' <history >>history.n
# View history file for mistakes.
if makehistory --r --s `wc --l <history` --f history.n; then
  mv history.n history
  mv history.n.dir history.dir
  mv history.n.pag history.pag
fi
ctlinnd go ''
```

BUGS AND LIMITATIONS

Makehistory does not handle symbolic links. If the news spool area is split across multiple partitions, the following commands should probably be run before the database is regenerated:

```
cd /var/news/storage/articles
find . -type l -name '[1-9]*' -print | xargs -t rm
```

Make sure to run the command on all the appropriate partitions!

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

active(4), ctlinnd(1m), dbz(3), filechan(1m), history(4), newsfeeds(4), makeactive(1m), newsrequeue(1m), snsnews(1m)

NAME

news.daily – does regular Usenet system administration

SYNOPSIS

news.daily [keyword...]

DESCRIPTION

News.daily performs a number of important Usenet administrative functions. This includes producing a status report, removing old news articles, processing log files, rotating the archived log files, renumbering the active file, removing any old socket files found in the ''firewall'' directory, and collecting the output. *This program should be run under the news administrator's id, not as root.*

By default, *news.daily* performs all of its functions and mails the output to the news administrator, *newsmaster*. By specifying ''keywords'' on the command line, it is possible to modify the functions performed, as well as change the arguments given to *expire*(1m) and *expireover*(1m).

News.daily should be run once a day, typically out of `cron`(1m). It may be run more often, but such invocations should at least use the ''notdaily'' keyword to prevent the log files from being processed and rotated too fast.

The *shlock*(1) program is used to prevent simultaneous executions.

KEYWORDS

The following keywords may be used:

delayrm

This keyword uses the ''–z'' flag when invoking *expire* and *expireover*. The names of articles to be removed are written to a temporary file, and then removed after expiration by calling *expirerm*(1m).

***expctl*=path**

Specifies the file to use as the *expire.ctl*(4) file for *expire*.

***expdir*=path**

By default, *expire* builds the new `history`(4) file and database in the same directory as the current files. Using this keyword specifies a different locale to build the new files (by passing the ''–d'' flag to *expire*), which will then be moved to the right location when finished.

nostat

Disables the status report generated by *innstat* (see *newslog*(1m)). Without this keyword, the status report is the first function performed, just prior to obtaining the *news.daily* lock.

notdaily

By default *news.daily* expects to be run only once a day. Use this keyword any extra times *news.daily* is run in the day and the normal logfile processing (and rotation) will not be done.

noexpire

By default, *expire* is invoked to remove old news articles. Using this keyword disables this function.

noexplog

Expire normally appends information to */var/news/logs/expire.log* (see *newslog*(4)). Using this keyword causes the *expire* output to be handled as part of *news.daily*'s output. It has no effect if the ''noexpire'' keyword is used.

***flags*='expire args'**

By default, *expire* is invoked with the an argument of ''–v1''. Using this keyword changes the arguments to those specified. Be careful to use quotes if multiple arguments are needed. This keyword has no effect if the ''noexpire'' keyword is used.

nologs

After expiration, *scanlogs*(1m) is invoked to process the log files. Using this keyword disables all log processing functions.

norotate

By default, log processing includes rotating and cleaning out log files. Using this keyword disables the rotating and cleaning aspect of the log processing: the logs files are only scanned for information and no contents are altered.

This keyword has no effect if the ''nologs'' keyword is used. The ''norotate'' keyword is passed on to *scanlogs* if it is invoked.

norenumber

Disables the *ctlinnd*(1m) renumber operation. Normally, the low-water mark for all newsgroups (see *active*(4)) is reset.

norm

By default, any socket *ctlinnd* socket that has not been modified for two days will be removed. Using this keyword disables this function.

nomail

News.daily normally sends a mail message containing the results to the administrator. Using this keyword causes this message to be sent to stdout and stderr instead. Normally, all utilities invoked by the script have their stdout and stderr redirected into a file. If the file is empty, no message is sent.

expireover

The *expireover* program is called after expiration to purge the overview databases.

***expireoverflags*='expireover args'**

If the ''expireover'' keyword is used, this keyword may be used to specify the flags to be passed to *expireover*. If the ''delayrm'' keyword is used, then the default value is ''–z'' and the list of deleted files; otherwise, the default value is ''–s''.

/full/path

The program specified by the given path is executed just before any expiration is done. A typical use is to specify an alternate expiration program and use the ''noexpire'' keyword. Multiple programs may be specified; they will be invoked in order.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

active(4), ctlinnd(1m), expire(1m), fastrm(1m), newslog(4), newslog(1m), innwatch.ctl(4), shlock(1).

NAME

scanlogs, writelog, innstat, tally.unwanted, tally.control: – programs to manipulate or summarize INN log files.

SYNOPSIS

none

DESCRIPTION

News log manipulation has been split into the `scanlogs`, `writelog`, `innstat`, `tally.control`, and `tally.unwanted` commands. Please refer to those man pages for more information.

SEE ALSO

innstat(1m), scanlogs(1m), tally.control(1m), tally.unwanted(1m), writelog(1m).

NAME

makeactive, makehistory, newsrequeue – tools to recover Usenet databases

SYNOPSIS

none

DESCRIPTION

This functionality has been split into the commands `makeactive`, `makehistory`, and `newsrequeue`. Please refer to the man pages for those commands for more information.

SEE ALSO

makehistory(1m), makeactive(1m), newsrequeue(1m)

NAME

newsrequeue – rewrites batchfiles.

SYNOPSIS

newsrequeue [–a *active*] [–h *history*] [–d *days*] [–l] [–n *newsfeeds*] [*input*]

DESCRIPTION

Newsrequeue is used to rewrite batchfiles after a system crash. It operates in two modes. In the first mode, it first reads the **active**(4) and **newsfeeds**(4) files to determine where the different newsgroups are to be distributed. It then opens the `history` database. Once the files are opened, *newsrequeue* reads from the specified *input* file, or standard input if no file is specified. Each line should have a single Message-ID, surrounded in angle brackets; any other text on the line is ignored. For example, the history file (or trailing subset of it) is acceptable input to the program operating in this mode.

Newsrequeue uses the first two fields of the *newsfeed* entry — the site name and the excludes field, and the patterns and distribs field. It ignores all flags in the third field except for the ''N'' field, and also ignores the fourth field altogether.

The output of *newsrequeue* consists of one line for each article that should be forwarded. Each such line contains the Message-ID, the filename, and the list of sites that should receive the article. The output is suitable for piping into **filechan**(1m).

The second mode is used if the ''–l'' flag is given. In this mode, it reads from the specified *input* file, or standard input if no file is specified. Each line should look like an *innd 1.5 Sec2* log entry. It parses entries for accepted articles, looks up the Message-ID in the history database to get the filename, and then scans the list of sites.

OPTIONS

–a To specify alternate locations for the active file, use the ''–a'' flag.

–n Use the ''–n'' flag to specify an alternate location for the **newsfeeds**(1m) file.

–h Use the ''–h'' flag to specify a different location for the history database,

–d If the ''–d'' flag is used, then only articles that were received within the specified number of *days* will be processed.

–l Reads **innd 1.5 Sec2** type log entries instead of a history-file like entries.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO active(4), ctlinnd(1m), dbz(3), filechan(1m), history(4), innd(1m), newsfeeds(4), makeactive(1m), makehistory(1m).

NAME

nntpget – gets Usenet articles from a remote NNTP server

SYNOPSIS

nntpget [–d *dist*] [–f *file*] [–n *newsgroups*] [–t *timestring*] [–o] [–u *file*] [–v] *host*

DESCRIPTION

Nntpget connects to the NNTP server at the specified *host* and retrieves articles from it. The Message-ID's of the desired articles are read from standard input. The articles are sent to standard output.

OPTIONS

–o May be used only if the command is executed on the host where the news feeder server is running. If this option is used, *nntpget* connects to the specified remote *host* to retrieve articles. Any article not present in the local `history` database is then fetched from the remote site and offered to the local server.

–v If the ''–v'' option is used with the ''–o'' option then the Message-ID of each article will be sent to standard output as it is processed.

–f The list of article Message-ID's is normally read from standard input. If the ''–f'' option is used, then a ''newnews'' command is used to retrieve all articles newer than the modification date of the specified `file`.

–u The ''–u'' option is the same as the "–f–" except that if the transfer succeeded, the file will be updated with a statistics line, modifying its timestamp so that it can be used in later invocations.

–t If the ''–t'' option is used, then the specified *timestring* is used as the time and date parameter to the ''newnews'' command.

–n If either the ''–t'' or ''–f'' options are used, then the ''–n'' option may be used to specify a newsgroup list. The default is ''*''.

–d The ''–d'' option may be used to specify a distribution list when using the ''–t'' or ''–f'' options. The default is no distribution list.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

snsnews(1m)

NAME

nntpsend – sends Usenet articles to remote site

SYNOPSIS

nntpsend [–a] [–c] [–d] [–D] [–p] [–r] [–S] [–l] [–n] [–t *timeout*] [–T *timelimit*] [–P *portnum*] [*sitenamefqdn*...]

DESCRIPTION

Nntpsend is a front-end that invokes **innxmit**(1m) to send Usenet articles to a remote NNTP site.

The sites to be fed may be specified by giving *sitename fqdn* pairs on the command line. If no such pairs are given, *nntpsend* defaults to the information given in the **nntpsend.ctl**(4) config file.

The *sitename* should be the name of the site as specified in the **newsfeeds**(4) file. The *fqdn* should be the hostname or IP address of the remote site.

An *innxmit* is launched for sites with queued news. All *innxmit* processes are spawned in the background, and the script waits for them all to finish before returning. Output is sent to the file `/var/news/logs/nntpsend.log`. In order to keep from overwhelming the local system, *nntpsend* waits five seconds before spawning each child.

Nntpsend expects that the batchfile for a site is named `/var/news/storage/out.going/sitename`.

When *sitename fqdn* pairs are given on the command line, any flags given on the command completely describe how **innxmit**(1m) operates. When no such pairs are given on the command line, then the information found in **nntpsend.ctl**(4) becomes the default flags for that site. Any flags given on the command line override the default flags for the site.

OPTIONS

`–c`

Disables message ID checking in streaming mode.

`–d –D`

The ‘‘–d’’ flag causes *nntpsend* to send output to stdout rather than the log file */var/news/logs/nntpsend.log*. The ‘‘–D’’ flag does the same, and it passes ‘‘–d’’ to all *innxmit* invocations which in turn causes *innxmit* to go into debug mode.

`–l`

If the ‘‘–l’’ (lazy) flag is specified, then the script will be more aggressive about deciding there is nothing to be done. This can be useful when using *nntpsend* as a backup for a site fed by *nntplink*.

–a –p –P –r –S –t –a –T

The ‘‘–a’’, ‘‘–p’’, ‘‘–P’’, ‘‘–r’’, ‘‘–S’’, ‘‘–t’’, and ‘‘–T’’ flags are passed on to the child *innxmit* program. See *innxmit*(1m) for more details. Note that if the ‘‘–p’’ flag is used then no connection is made and no articles are fed to the remote site. It is useful to have `cron`(1m) invoke *nntpsend* with this flag in case a site cannot be reached for an extended period of time.

EXAMPLES

EXAMPLE 1 `nntpsend` Example

With the following control file:

```
nsavax:erehwon.nsavax.gov::-S -t60
group70:group70.org::
xyz:xyz.com:4m-1m:-T1800 -t300
kremvax:kremvax.cis:2m:
```

The command:

```
% nntpsend
```

will result in the following:

```
Sitename Truncation Innxmit flags
nsavax (none) --a --S --t60
group70 (none) --a --t180
xyz 1m if >4m --a --T1800 --t300
kremvax 2m --a --t180
```

The command:

```
% nntpsend -d -T1200
```

will result in the following:

```
Sitename Truncation Innxmit flags
nsavax (none) --a --d --S --T1200 --t60
group70 (none) --a --d --T1200 --t180
xyz 1m if >4m --a --d --T1200 --t300
kremvax 2m --a --d --T1200 --t180
```

Remember that ‘‘–a’’ is always given, and ‘‘–t’’ defaults to 180.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

innxmit(1), newsfeeds(4), nntpsend.ctl(4)

NAME

overchan – updates the news overview database

SYNOPSIS

overchan [–D *dir*] [–c] [*file...*]

DESCRIPTION

Overchan reads article data from *files* or standard input if none are specified. (A single dash in the file list means to read standard input.) It uses this information to update the news overview database. *Overchan* is designed to be used by InterNetNews or the C News ''mkov'' packages to update the database as the articles come in. The database for each newsgroup is stored in a file named *.overview* in a newsgroup directory within the overview database tree.

Overchan locks the database file by locking an auxiliary file before appending the new data. To purge data after articles have been expired, see **expireover**(1m).

By default, *overchan* processes its input as an INN overview stream written as a ''WO'' entry in the **newsfeeds**(4) file, for example:

overview:*:Tc,WO:/usr/news/bin/overchan

This data consists of a line of text, separated into two parts by a tab. The first part is a list of all relative pathnames where the article has been written with a single space between entries. The second part is the data to be written into the overview file, except that the initial article number is omitted. The data in the overview files should be expired by running **expireover**(1m). This is normally done by adding the ''expireover'' flag to the **news.daily**(1m) invocation.

OPTIONS

–c Processes the output of the *mkov*(1m) program. This format is described in the ''nov'' distribution.

–D Specifies where the databases are stored. The default directory is `/var/news/storage/over.view`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

expireover(1m), news.daily(1m), newsfeeds(4), newsoverview(4), newsoverview(1m).

NAME

prunehistory – removes file names from Usenet history file

SYNOPSIS

prunehistory [–f *filename*] [–p] [*input*]

DESCRIPTION

Prunehistory modifies the **history**(4) text file to ‘‘remove’’ a set of filenames from it. The filenames are removed by overwriting them with spaces, so that the size and position of any following entries does not change.

Prunehistory reads the named *input* file, or standard input if no file is given. The input is taken as a set of lines. Blank lines and lines starting with a number sign (‘‘#’’) are ignored. All other lines are should consist of a Message-ID followed by zero or more filenames.

The Messge-ID is used as the **dbz**(3) key to get an offset into the text file. If no filenames are mentioned on the input line, then all filenames in the text are ‘‘removed.’’ If any filenames are mentioned, they are converted into the history file notation. If they appear in the line for the specified Message-ID then they are removed.

Since the news feeder daemon only appends to the text file, *prunehistory* does not need to have any interaction with it.

OPTIONS

–p *Prunehistory* will normally complain about lines that do not follow the correct format. If the ‘‘–p’’ flag is used, then the program will silently print any invalid lines on its standard output. (Blank lines and comment lines are also passed through.) This can be useful when *prunehistory* is used as a filter for other programs such as *reap*.

–f The default name of the history file is `/var/news/state/history`; to specify a different name, use the ‘‘–f’’ flag.

EXAMPLES

EXAMPLE 1 It is a good idea to delete purged entries and rebuild the *dbz* database periodically by using a script like the following:

```
ctlinnd throttle "Rebuilding history database"
cd /var/news/state
awk 'NF > 2 {
 printf "%s\t%s\t%s", $1, $2, $3;
 for (i = 4; i <= NF; i++)
  printf " %s", $i;
 print "\n";
}' <history >history.n
if makehistory --r --f history.n ; then
  mv history.n history
  mv history.n.pag history.pag
  mv history.n.dir history.dir
else
  echo 'Problem rebuilding history; old file not replaced'
fi
```

```
ctlinnd go "Rebuilding history database"
```

Note that this keeps no record of expired articles.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

dbz(3), history(4), snsnews(1m)

NAME

rnews – receive news from a UUCP connection

SYNOPSIS

rnews [–h *host*] [–v] [–U] [–N] [–S *master*] [*input*]

DESCRIPTION

Rnews reads messages typically queued by a UUCP newsfeed and sends them to the local InterNetNews server. The message is read from the specified input file, or standard input if no input is named.

When sent over UUCP, Usenet articles are typically joined in a single batch to reduce the UUCP overhead. Batches can also be compressed to reduce the communication time. If a message does not start with a number sign (‘‘#’’) and an exclamation point, then the entire input is taken as a single news article. If it does start with with those two characters, then the first line is read and interpreted as a batch command.

If the command is ‘‘#! rnews nnn’’ where *nnn* is a number, then the next *nnn* bytes (starting with the next line) are read as a news article.

If the command is ‘‘#! cunbatch’’ then the rest of input is fed to the **compress**(1) program with the ‘‘–d’’ flag to uncompress it, and the output of this pipe is read as *rnews*’s input. This is for historical compatibility — there is no program named *cunbatch*. A compressed batch will start with a ‘‘#! cunbatch’’ line, then contain a series of articles separated by ‘‘#! rnews nnn’’ lines.

If the command is any other word, then *rnews* will try to execute a program with that name in the directory `/usr/news/bin/rnews.libexec`. The batch will be fed into the program’s standard input, and the standard output will be read back as input into *rnews*.

If *rnews* detects any problems with an article such as a missing header, or an unintelligible reply from the server, it will save a copy of the article in the `/var/news/storage/in.coming/bad` directory.

OPTIONS

–S If the ‘‘–S’’ flag is used, then *rnews* connects to the specified host. If the flag is not used, it will try to connect to the server by opening a UNIX-domain stream connection. If that fails, it will try to open a TCP connection to the default remote server.

–U If the server is not available, the message is spooled into a new file created in the `/var/news/storage/in.coming` directory. The ‘‘–U’’ flag may be used to send all spooled messages to the server once it becomes available again, and can be invoked regularly by **cron**(1m).

–N Normally, if unpacking the input fails it is respooled to `/var/news/storage/in.coming` for another attempt later. If the

''–N'' flag is used, then no such respooling is done and rnews exits with status value ''9'' to indicate this.

–v If the ''–v'' flag is used, it prints a notice of all errors on the standard error, naming the input file (if known) and printing the first few characters of the input. Errors are always logged.

–h If the ''–h'' flag is given, or failing that, the enviroment variable *UU_MACHINE* is set, then *rnews* will log the Message-ID and host for each article offered to the server. Logging will only be done if the value is not an empty string.

BUGS

Rnews cannot process articles that have embedded \0's in them.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

snsnews(1m)

NAME

scanlogs – summarizes INN log files.

SYNOPSIS

scanlogs [norotate] [nonn]

DESCRIPTION

Scanlogs summarizes the information recorded in the INN log files (see **newslog**(4)) By default, it also rotates and cleans out the logs. It is normally invoked by the **news.daily**(1m) script.

KEYWORDS

The following keywords are accepted:

norotate Using this keyword disables the rotating and cleaning aspect of the log processing: the log files are only scanned for information and no contents are altered.

nonn Normally the *nn* log file is scanned and rotated. Using this keyword disables this function.

If *scanlogs* is invoked more than once a day, the ''norotate'' keyword should be used to prevent premature log cleaning.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

newslog(4), news.daily(1m), snsd(1m), snsnews(1m)

NAME

snsd – The Sun Internet News Reader daemon

SYNOPSIS

snsd [–p *port*] [–F *host=name*[,*port=name*]]

DESCRIPTION

The Sun Internet News Server daemon, `snsd`, handles all incoming NNTP connections. It listens on the NNTP port (see the "–p" option) for both reader and feeder connections. `snsd` handles all the reader connections directly. It dispatches all feeder connections to the feeder daemon (see **innd**(1m) and **snsnews**(1m)) which has been modified to receive such dispatches.

OPTIONS

–P Specifies the port number to listen on for NNTP connections. The default is the `services(4)`entry for `nntp` or port 119 if there is no such entry.

–F Specifies the remote host to be used as a posting host. This is relevant only when the Sun News Server is installed in the Reader-only configuration. The `–port=number` portion is optional. If it is not specified, the default is the `services(4)` entry for `nntp` or port 119 if there is no such entry.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Evolving

SEE ALSO

`hosts.nntp(4), sns.conf(4), syslog(1m), snsnews(1m)`

NAME

snsnews – Start or stop the Sun Internet News Server reader and feeder daemon(s)

SYNOPSIS

/etc/init.d/snsnews { start | stop }

DESCRIPTION

Starts or stops the news feeder daemon **innd**(1m) and the news reader daemon **snsd**(1m).

If the command is entered on a reader/feeder server, both daemons are started by `snsnews`. On a feeder server, `snsnews` starts only the feeder daemon `innd`; and on a reader server, `snsnews` starts only the reader daemon `snsd`.

Note - The feeder daemon and reader daemon should be started only via **snsnews**(1m).

OPERANDS

The following operands are supported:

start Start the news server(s)

stop stop the news server(s)

EXAMPLES

EXAMPLE 1

```
/etc/init.d/snsnews start
/etc/init.d/snsnews stop
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

NAME

tally.control – keeps track of newsgroup creation and deletion.

SYNOPSIS

tally.control

DESCRIPTION

tally.control is normally invoked by **scanlogs**(1m) It reads its standard input, which should be the `newsgroup.log` and `rmgroup.log` log files. It updates the cumulative list of newsgroup creations and deletions, `control.log`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

news.daily(1m), scanlogs(1m), tally.control(1m), tally.unwanted(1m), writelog(1m).

NAME

tally.unwanted – keeps track of unwanted newsgroups.

SYNOPSIS

tally.unwanted

DESCRIPTION

tally.unwanted is normally invoked by **scanlogs**(1m). It parses the article log file to update the cumulative list of articles posted to unwanted newsgroups, `/var/news/logs/unwanted.log`.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

news.daily(1m), newslog(4), snsd(1m), scanlogs(1m), snsnews(1m), tally.control(1m), tally.unwanted(1m), writelog(1m).

NAME

writelog – adds an entry to an INN log file.

SYNOPSIS

writelog name text...

DESCRIPTION

The *writelog* script is used to write a log entry or send it as mail. The *name* parameter specifies the name of the log file where the entry should be written. If it is the word ‘‘mail’’ then the entry is mailed to the news administrator, newsmaster. The data that is written or sent consists of the *text* given on the command line, followed by standard input indented by four spaces.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

innstat(1m) news.daily(1m), newslog(4), snsd(1m), scanlogs(1m), snsnews(1m)

CHAPTER 2

man Pages(4): File Formats

NAME

newsIntro.4 – Introduction to the host configuration files for the Sun™ Internet News Server™.

DESCRIPTION

The man pages offer detailed instruction and examples on keywords and parameters for each configuration file.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Evolving

SEE ALSO

newsIntro(1m)

NOTES

active(4) The file `/var/news/state/active` lists the newsgroups that the local site receives.

control.ctl(4) The file `/var/news/config/control.ctl` is used to determine what action is taken when a control message is received.

distrib.pats(4) The file `/var/news/config/distrib.pats` is used to determine the default value of the Distribution header.

expire.ctl(4) The file `/var/news/config/expire.ctl` is the default control file for the **expire**(1m) program, which reads it at startup.

history(4) The file `/var/news/state/history` keeps a record of all articles currently stored in the news system, as well as those that have been received but since expired.

hosts.nntp(4) The file `/var/news/config/hosts.nntp` is read by the feeder daemon to get the list of hosts that feed the local site Usenet news using the NNTP protocol.

inn.conf(4) The file `/var/news/config/inn.conf` specifies the configuration data for Sun Internet News Server programs.

innwatch.ctl(4)	The file /var/news/config/innwatch.ctl is used to determine what actions are taken during the periodic supervisions by **innwatch**(1m).
moderators(4)	The file /var/news/config/moderators contains the email addresses for moderated Usenet newsgroups.
newsfeeds(4)	The file /var/news/config/newsfeeds specifies how incoming articles should be distributed to other sites.
newslog(4)	Describes the Sun™ Internet News Server™ log files. Most log files created by the Sun™ Internet News Server™ programs reside in the /var/news/logs directory and have a ''.log'' extension. Several versions are usually kept with an additional extension such as ''.1'', ''.2'', etc. - the higher the number, the older the log. The older versions are compressed.
nnrp.access(4)	The file /var/news/rconfig/nnrp.access specifies the access control for those NNTP sites that are not handled by the news feeder daemon.
nntpsend.ctl(4)	The file /var/news/config/nntpsend.ctl specifies the default list of sites to be fed by nntpsend(1m).
overview.fmt(4)	The file /var/news/config/overview.fmt specifies the organization of the news overview database.
passwd.nntp(4)	The file /var/news/config/passwd.nntp contains host-name-password triplets for use when authenticating client programs to NNTP servers.
sns.conf(4)	The Sun Internet News Server configuration data file /var/news/rconfig/sns.conf specifies the maximum number of threads per NNTP handling process, and the maximum number of NNTP processes for the platform on which the server is running. sns.conf is updated using the News Administration GUI.

NAME

active, active.times – the list of active Usenet newsgroups

DESCRIPTION

The file `/var/news/state/active` lists the newsgroups that the local site receives. Each newsgroup should be listed only once. Each line specifies one group; their order in the file does not matter. Within each newsgroup, articles are assigned unique names, which are monotonically increasing numbers.

If an article is posted to newsgroups not mentioned in this file, those newsgroups are ignored. If no valid newsgroups are specified, the article is filed into the newsgroup ‘‘junk’’ and only propagated to sites that receive the ‘‘junk’’ newsgroup.

Each line consists of four fields specified by a space:

```
name himark lomark flags
```

The first field is the name of the newsgroup. Newsgroups that start with the three characters ‘‘to.’’ are treated specially; refer to the *INN 1.5* specifications. The second field is the highest article number that has been used in that newsgroup. The third field is the lowest article number in the group; this number is not guaranteed to be accurate, and should only be taken to be a hint. Note that because of article cancellations, there may be gaps in the numbering sequence. If the lowest article number is greater then the highest article number, then there are no articles in the newsgroup. In order to make it possible to update an entry in-place without rewriting the entire file, the second and third fields are padded out with leading zeros to make them a fixed width.

The fourth field can contain one of the following flags:

```
y    Local postings are allowed
n    No local postings are allowed, only remote ones
m    The group is moderated and all postings must be approved
j    Articles in this group are not kept, but only passed on
x    Articles cannot be posted to this newsgroup
=foo.bar Articles are locally filed into the ‘‘foo.bar’’ group
```

If a newsgroup has the ‘‘j’’ flag, then no articles will be filed into that newsgroup and local postings to that group should not be generated. If an article for such a newsgroup is received from a remote site, it will be filed into the ‘‘junk’’ newsgroup if it is not cross-posted. This is different from not having a newsgroup listed in the file because sites can subscribe to ‘‘j’’ newsgroups and the article will be propagated to them.

If the fourth field of a newsgroup starts with an equal sign, then the newsgroup is an alias. Articles can be posted to the group, but will be treated as if they were posted to the group named after the equal sign. The second and third fields are ignored. Note that the Newsgroup header is not modified (Alias groups are typically used during a transition, and are typically created with **ctlinnd**(1m). An alias newsgroup should not point to another alias.

The file `/var/news/state/active.times` provides a chronological record of when newsgroups are created. This file is normally updated by the feeder daemon (see **innd**(1m)) whenever a *ctlinnd* ''newgroup'' command is done. Each line consist of three fields:

```
name time creator
```

The first field is the name of the newsgroup. The second field is the time it was created, expressed as the number of seconds since the epoch — that is, a *time_t*; see **gettimeofday**(2) The third field is the electronic mail address of the person who created the group.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

ctlinnd(1m), snsnews(1m)

NAME

control.ctl – specifies handling of Usenet control messages

DESCRIPTION

The file `/var/news/config/control.ctl` is used to determine what action is taken when a control message is received. It is read by the *parsecontrol* script, which is called by all the control scripts.

The file consists of a series of lines; blank lines and lines beginning with a number sign (‘‘#’’) are ignored. All other lines consist of four fields separated by a colon:

message:from:newsgroups:action

The first field is the name of the message for which this line is valid. It should be either the name of the control message, or the word ‘‘all’’ to mean that it is valid for all messages.

The second field is a shell-style pattern that matches the email address of the person posting the message. (The poster’s address is first converted to lowercase.) The matching is done using the shell’s `case` statement; see **sh**(1) for details.

If the control message is ‘‘newgroup’’ or ‘‘rmgroup’’ then the third field specifies the shell-style pattern that must match the group being created or removed. If the control message is of a different type, then this field is ignored.

The fourth field specifies what action to take if this line is selected for the message. The following actions are understood:

`doit` The action requested by the control message should be performed. In most cases the control script will also send mail to newsmaster.

`doifarg` If the control message has an argument, this is treated as a ‘‘doit’’ action. If no argument was given, it is treated as a ‘‘mail’’ entry. This is used in ‘‘sendsys’’ entries script so that a site can request its own **newsfeeds**(4) entry by posting a ‘‘sendsys mysite’’ article. On the other hand, sendsys ‘‘bombs’’ ask that the entire *newsfeeds* file be sent to a forged reply-to address; by using ‘‘doifarg’’ such messages will not be processed automatically.

`doit=file` The action is performed, but a log entry is written to the specified log file, `file`. If `file` is the word ‘‘mail’’ then the record is mailed. A null string is equivalent to /dev/null. A pathname that starts with a slash is taken as the absolute filename to use as the log. All other pathnames are written to the `/var/news/logs/` file.*log*. The log is written by *writelog* (see **newslog**(1m))

`drop` No action is taken; the message is ignored.

`log` A one-line log notice is sent to standard error. The news feeder daemon normally directs this to the file `/var/news/logs/errlog`.

`log=`***file*** A log entry is written to the specified log file, `file`, which is interpreted as previously described.

`mail` A mail message is sent to the news administrator.

Lines are matched in order; the last match found in the file is the one that is used. For example, with the following three lines:

```
newgroup:*:*:drop
newgroup:tale@*.uu.net:comp.*|misc.*|news.*|sci.*|soc.*|talk.*:doit
newgroup:kre@munnari.oz.au:aus.*:mail
```

A newgroup coming from ''tale'' at a UUNET machine will be honored if it is in the mainstream Usenet hierarchy. If ''kre'' posts a newgroup message creating ''aus.foo'', then mail will be sent. All other newgroup messages are ignored.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

newsfeeds(4), scanlogs(1m), snsnews(1m)

NAME

distrib.pats – specifies default values for Usenet Distribution header

DESCRIPTION

The file `/var/news/config/distrib.pats` is used to determine the default value of the Distribution header. It consists of a series of lines; blank lines and lines beginning with a number sign (‘‘#’’) are ignored. All other lines consist of three fields separated by a colon:

```
weight:pattern:value
```

The first field is the weight to assign to this match. If a newsgroup matches multiple lines, the line with the heighest weight is used. This should be an arbitrary number greater than zero. Unlike other INN files, the order of lines in this file is not important.

The second field is the name of the newsgroup or a wild card pattern to specify a set of newsgroups. Multiple patterns are not allowed.

The third field is the value that should be used if this line is picked as the best match. It can be an empty string.

Programs that process a user’s posting should consult this file. The intent is that all newsgroups to which an article is posted be used to index into this file, and the value with the highest weight should be used as the value of the Distribution header, if none was specified.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

NAME

expire.ctl – the control file for Usenet article expiration

DESCRIPTION

The file `/var/news/config/expire.ctl` is the default control file for the **`expire`**(1m) program, which reads it at startup. Blank lines and lines beginning with a number sign (‘‘#’’) are ignored. All other lines should be in one of two formats.

The first format specifies how long to keep a record of fully expired articles. This is useful when a newsfeed intermittently offers older news that is not kept around very long. There should only be one line in this format, which looks like this:

/remember/:days

where *days* is a floating-point number that specifies the upper limit to remember a Message-ID, even if the article has already expired. (It does not affect article expirations.)

Most of the lines in the file will consist of five colon-separated fields, as follows:

```
pattern:modflag:keep:default:purge
```

The *pattern* field is a list of character string and/or wild card patterns, separated by commas. This field specifies the newsgroups to which the line is applied. Note that the file is interpreted in order, so that the last line that matches will be used. This means that general patterns (like a single asterisk to set the defaults) should appear before specific group specifications.

The *modflag* field can be used to further limit newsgroups to which the line applies, and should be chosen from the following set:

```
M Only moderated groups
U Only unmoderated groups
A All groups
```

The next three fields are used to determine how long an article should be kept. Each field should be either a number of days (fractions like ‘‘8.5’’ are allowed) or the word ‘‘never.’’ The most common use is to specify the default value for how long an article should be kept. The first and third fields *keep* and *purge* specify the boundaries within which an Expires header will be honored. They are ignored if an article has no Expires header. The fields are specified in the

file as ‘‘lower-bound default upper-bound,’’ and they are explained in this order. Since most articles do not have explicit expiration dates, however, the second field tends to be the most important one.

The *keep* field specifies how many days an article should be kept before it will be removed. No article in the newsgroup will be removed if it has been filed for less then *keep* days, regardless of any expiration date. If this field is the word ‘‘never’’, then an article cannot have been kept for enough days so it will never be expired.

The *default* field specifies how long to keep an article if no Expires header is present. If this field is the word ‘‘never’’ then articles without explicit expiration dates will never be expired.

The *purge* field specifies the upper bound on how long an article can be kept. No article will be kept longer then the number of days specified by this field. All articles will be removed after they have been kept for *purge* days. If *purge* is the word ‘‘never’’ then the article will never be deleted.

It is often useful to honor the expiration headers in articles, especially those in moderated groups. To do this, set *keep* to zero, *default* to whatever value you wish, and *purge* to never. To ignore any Expires header, set all three fields to the same value.

There must be exactly one line with a *pattern* of ‘‘*’’ and a *modflags* of ‘‘A’’. This matches all groups and is used to set the expiration default. It should be the first expiration line.

For example,

```
## How long to keep expired history
/remember/:5
## Most things stay for two weeks
*:A:14:14:14
## Believe expiration dates in moderated groups, up to six weeks
*:M:1:30:42
## Keep local stuff for a long time
foo.*:A:30:30:30
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO expire(1m)

NAME

history – the record of current and recently expired Usenet articles

DESCRIPTION

The file `/var/news/state/history` keeps a record of all articles currently stored in the news system, as well as those that have been received but since expired. In a typical production environment, this file will be many megabytes.

The file consists of text lines. Each line corresponds to one article. The file is normally kept sorted in the order in which articles are received, although this is not a requirement. The news feeder daemon appends a new line each time it files an article, and **`expire`**(1m) builds a new version of the file by removing old articles and purging old entries.

Each line consists of two or three fields separated by a tab:

```
<Message--ID>  \t  date
<Message--ID>  \t  date  \t  files
```

The *Message–ID* field is the value of the article's Message-ID header, including the angle brackets.

The `date` field consists of three subfields separated by a tilde. All subfields are the text representation of the number of seconds since the epoch — that is, a *time_t*; see **`gettimeofday`**(2) The first subfield is the article's arrival date. If copies of the article are still present then the second subfield is either the value of the article's Expires header, or a hyphen if no expiration date was specified. If an article has been expired then the second subfield will be a hyphen. The third subfield is the value of the article's Date header, recording when the article was posted.

The *files* field is a set of entries separated by one or more spaces. Each entry consists of the name of the newsgroup, a slash, and the article number. This field is empty if the article has been expired.

For example, an article cross-posted to comp.sources.unix and comp.sources.d that was posted on February 10, 1991 (and received three minutes later), with an expiration date of May 5, 1991, could have a history line (broken into two lines for display) like the following:

```
<312@litchi.foo.com> \t 666162000~673329600~666162180 \t
  comp.sources.unix/1104 comp.sources.d/7056
```

In addition to the text file, there is a **dbz**(3z) database associated with the file that uses the Message-ID field as a key to determine the offset in the text file where the associated line begins. For historical reasons, the key includes the trailing \0 byte (which is not stored in the text file).

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

dbz(3z), expire(1m), news-recovery(1m), snsnews(1m)

NAME

hosts.nntp, hosts.nntp.nolimit – list of hosts that feed NNTP news

DESCRIPTION

The file `/var/news/config/hosts.nntp` is read by the news feeder daemon to get the list of hosts that feed the local site Usenet news using the NNTP protocol. The server reads this file at start-up or when directed to by **`ctlinnd`**(1m).

When a host connects to the NNTP port of a system on which the news feeder daemon is running, the feeder daemon will do a check to see if the connecting host is named in this file. If the host is found, the daemon will allow it to provide its news feed.

When a host connects to the NNTP port of a system that is configured as a reader/feeder news server, the local server daemon will check to see if the host is listed in `hosts.nntp` as a feed provider. If so, the reader daemon will hand the feed over to the feeder daemon. If the host is not listed, the connection will be handled by the reader daemon.

Comments begin with a number sign (‘‘#’’) and continue through the end of the line. Blank lines and comments are also ignored. All other lines should consist of two or three fields separated by a colon.

The first field should be either an Internet address in dotted-quad format or a full domain name of a system. If a host’s entry has multiple addresses, all of them will be added to the access list. The second field, which may be blank, is the password the foreign host is required to use when first connecting. The third field, which may be omitted, is a list of newsgroups to which the host may send articles. This list is parsed as a **`newsfeeds`**(4) subscription list; groups not in the list are ignored.

For example:

```
## FOO has a password, UUNET and VIX do not.
## UUNET cannot post to local groups.
## These are comment lines.
news.foo.com:magic
uunet.uu.net::!foo.*
data.ramona.vix.com:
```

The first field may be suffixed by ‘‘/s’’ to indicate that streaming commands are specifically permitted to be used by this host. By default streaming commands are available to all hosts. If *any* entry in hosts.nntp has a ‘‘/s’’ suffix, then only those hosts with the ‘‘/s’’ suffix will be permitted to use streaming commands.

For example, with the following hosts.nntp file, only the host data.ramona.vix.com is allowed to use the streaming commands.

```
## As above, but
news.foo.com:magic
uunet.uu.net::!foo.*
data.ramona.vix.com/s:
```

Since the news feeder daemon is usually started at system boot time, the local nameserver may not be fully operational when the feeder daemon parses this file. As a work-around, a *ctlinnd* ''reload'' command can be performed after a delay of an hour or so. It is also possible to provide both a host's name and its dotted-quad address in the file.

If the file contains passwords, it should not be world-readable. The file `/var/news/config/hosts.nntp.nolimit`, if it exists is read whenever the ''hosts.nntp'' file is read. It has the same format, although only the first field is used. This can be used to allow local hosts or time-sensitive peers to connect regardless of the local conditions.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

ctlinnd(1m), snsd(1m,) snsnews(1m)

NAME

inn.conf – specifies the configuration data for Sun Internet News Server programs

DESCRIPTION

The file `/var/news/config/inn.conf` is used to determine various parameters. Blank lines and lines starting with a number sign (‘‘#’’) are ignored. All other lines specify parameters that may be read, and should be of the following form:

```
name : [optional white space] value
```

Everything after the white space and up to the end of the line is taken as the value; multi-word values should not be put in quotes. The case of names is significant — *server* is not the same as *Server* or *SERVER*.

Some parameters specified in the file may be overridden by environment variables, and some file parameters may be used to mask real data, such as when hiding a cluster of hosts behind a single electronic mail hostname. The current set of parameters is as follows:

fromhost The name of the host to use when building the From header line. The default is the fully-qualified domain name of the local host. The value of the FROMHOST environment variable, if it exists, overrides this.

moderatormailer The name of the default machine that contains forwarding aliases for all moderated groups. It is only used if the *moderators*(4) file doesn’t exist, or if the group is not matched by that file. The value is interpreted as a pattern match; see *moderators*(4).

organization Specifies what to put in the Organization header if it is blank. The value of the ORGANIZATION environment variable, if it exists, overrides this.

pathhost Specifies how to name the local site when building the Path header line. The default is the fully-qualified domain name of the local host.

server Specifies the name of the NNTP server to which an article should be posted. The value of the NNTPSERVER environment variable, if it exists, overrides this.

domain The domain name of the local host. It should not have a leading period, and it should not be a full host address, and is used only if the fully qualified domain name cannot be obtained. The check is very simple; if either routine returns a name with a period in it, then it is assumed to have the full domain name.

Three parameters are used only by *snsd* when accepting postings from clients:

mime-version If this parameter is present, then *snsd* will add the necessary MIME (Multipurpose Internet Mail Extensions) headers to all articles that do not have a Mime-Version header. This parameter specifies the MIME version, and should normally be ‘‘1.0’’.

mime-contenttype If MIME headers are being added, this parameter specifies the value of the Content-Type header. The default value is ‘‘text/plain; charset=US-ASCII.’’

mime-encoding If MIME headers are being added, this parameter specifies the value of the Content-Transfer-Encoding header. The default value is ‘‘7bit.’’

Note that this file can be identical on all machines in an organization.

EXAMPLES

```
fromhost: foo.com
moderatormailer: %s@uunet.uu.net
organization: Foo, Incorporated
#pathhost -- use FQDN.
server: news.foo.com
domain: foo.com
```

This file is intended to be fairly static; any changes made are typically not reflected until a program restarts.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO moderators(4).

NAME

innwatch.ctl – controls Usenet supervision by innwatch

DESCRIPTION

The file `/var/news/config/innwatch.ctl` is used to determine what actions are taken during the periodic supervisions by *innwatch.*

The file consists of a series of lines; blank lines and lines beginning with a number sign (‘‘#’’) are ignored. All other lines consist of seven fields, each preceded by a delimiting character:

:label:state:condition:test:limit:command:reason

The delimiter can be any one of several non-alphanumeric characters that does not appear elsewhere in the line; there is no way to quote it to include it in any of the fields. Any of ‘‘!’’, ‘‘,’’, ‘‘:’’, ‘‘@’’, ‘‘;’’, or ‘‘?’’ is a good choice. Each line can have a different delimiter; the first character on each line is the delimiter for that line. Whitespace surrounding delimiters, except before the first, is ignored, and does not form part of the fields, whitespace within fields is permitted. All delimiters must be present.

The first field is a label for the control line. It is used as an internal state indicator and in *ctlinnd* messages to control the server. If omitted, the line number is used.

The second field specifies when this control line should be used. It consists of a list of labels, and special indicators, separated by whitespace. If the current state matches against any of the labels in this field, this line will be used as described below. Values that can be used are:

- This line matches if the current state is the same as the label on this line, or if the current state is ‘‘run,’’ the initial state. This is also the default state if this field is empty.
- This line matches if the current state is ‘‘run.’’
- This line always matches.
- This line matches if the current state is the specified ‘‘label.’’
- This line matches if the current state is not the specified ‘‘label.’’

The third field specifies a shell command that is invoked if this line matches. Do not use any shell filename expansion characters such as ‘‘*’’, ‘‘?’’, or ‘‘[’’ (even quoted, they’re not likely to work as intended). If the command succeeds, as indicated by its exit status, it is expected to have printed a single integer to standard output. This gives the value of this control line, to be used below. If the command fails, the line is ignored. The command is executed with its current directory set to the news spool directory, `/var/news/storage/articles`.

The fourth field specifies the operator to use to test the value returned above. It should be one of the two letter numeric test operators defined in **test**(1) such as ‘‘eq’’, ‘‘lt’’ and the like. The leading dash (‘‘–’’) should not be included.

The fifth field specifies a constant with which to compare the value using the operator just defined. This is done by invoking the command

test value -operator constant

The line is said to ‘‘succeed’’ if it returns true.

The sixth field specifies what should be done if the line succeeds, and in some cases if it fails. Any of the following words may be used:

throttle Causes *innwatch* to throttle the server if this line succeeds. It also sets the state to the value of the line’s label. If the line fails, and the state was previously equal to the label on this line (that is, this line had previously succeeded), then a *go* command will be sent to the server, and *innwatch* will return to the ‘‘run’’ state. The ‘‘throttle’’ is only performed if the current state is ‘‘run’’ or a state other than the label of this line, regardless of whether the command succeeds.

pause Is identical to ‘‘throttle’’ except that the server is paused.

shutdown Sends a ‘‘shutdown’’ command to the server. For emergency use only.

flush Sends a ‘‘flush’’ command to the server.

go Causes *innwatch* to send a ‘‘go’’ command to the server and to set the state to ‘‘run.’’

exit Causes *innwatch* to exit.

skip The result of the control file is skipped for the current pass.

The last field specifies the reason that is used in those *ctlinnd* commands that require one. More strictly, it is part of the reason *innwatch* appends some information to it. In order to enable other sites to recognize the state of the local *innd* server, this field should usually be set to one of several standard values. Use ‘‘No space’’ if the server is rejecting articles because of a lack of file system resources. Use ‘‘loadav’’ if the server is rejecting articles because of a lack of CPU resources.

Once *innwatch* has taken some action as a consequence of its control line, it skips the rest of the control file for this pass. If the action was to restart the server, (that is, issue a ‘‘go’’ command), then the next pass will commence

almost immediately, so that *innwatch* can discover any other condition that may mean that the server should be suspended again.

EXAMPLES

EXAMPLE 1 innwatch example

```
 @@@df .|awk 'NR==2 {print $4}'@lt@10000@throttle@No space
@@@df -i .|awk 'NR==2 {print $4}'@lt@1000@throttle@No space (inodes)
```

The first line causes the server to be throttled if the free space drops below 10000 units (using whatever units `df` uses), and restarted again when free space increases above the threshold.

The second line does the same for inodes.

The next three lines act as a group and should appear in the following order. It is easier to explain them, however, if they are described from the last up.

```
!load!load hiload!loadavg!lt!5!go!
:hiload:+ load:loadavg:gt:8:throttle:loadav
/load/+/loadavg/ge/6/pause/loadav
```

The final line causes the server to be paused if *innwatch* is in the ‘‘run’’ state and the load average rises to, or above, six. The state is set to ‘‘load’’ when this happens. The previous line causes the server to be throttled when *innwatch* is in the ‘‘run’’ or ‘‘load’’ state, and the load average rises above eight. The state is set to ‘‘hiload’’ when this happens. Note that *innwatch* can switch the server from ‘‘paused’’ to ‘‘throttled’’ if the load average rises from below six to between six and seven, and then to above eight. The first line causes the server to be sent a ‘‘go’’ command if *innwatch* is in the ‘‘load’’ or ‘‘hiload’’ state, and the load average drops below five.

Note that all three lines assume a mythical command *loadavg* that is assumed to print the current load average as an integer. In more practical circumstances, a pipe of `uptime` into `awk` is more likely to be useful.

BUGS

This file must be tailored for each individual site, the sample supplied is truly no more than a sample. The file should be ordered so that the more common problems are tested first.

The ‘‘run’’ state is not actually identified by the label with that three-letter name, and using it will not work as expected.

Using an ‘‘unusual’’ character for the delimiter such as ‘‘(’’, ‘‘*’’, ‘‘&’’, ‘‘`’’, ‘‘á’’, and the like, is likely to lead to obscure and hard to locate bugs.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

ctlinnd(1m), news.daily(1m), snsnews(1m)

NAME

moderators – the mail addresses for moderated Usenet newsgroups

DESCRIPTION

The file `/var/news/config/moderators` is used to determine how to reach the moderator of a newsgroup. This is used when an unapproved local posting is made to a moderated newsgroup.

The file is read until a match is found. Blank lines and lines starting with a number sign (“#”) are ignored. All other lines should consist of two fields separated by a colon.

The first field is a wild card-style pattern. If it matches the name of the newsgroup, then the second field is taken to be a format string for **`sprintf`**(3) This string should have at most one *%s* parameter, which will be given the name of the newsgroup with periods transliterated to dashes.

Here is a sample file:

```
foo.important:announce-request@foo.com
foo.*:%s@mailer.foo.com
gnu.*:%s@prep.ai.mit.edu
*:%s@uunet.uu.net
```

Using the above file, postings to the moderated newsgroup in the left column will be sent to the address shown in the right column:

```
foo.important announce-request@foo.com
foo.x.announce foo-x-announce@mailer.foo.com
gnu.emacs.sources gnu-emacs-sources@prep.ai.mit.edu
comp.sources.unix comp-sources-unix@uunet.uu.net
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

sprintf(3), inn.conf(4)

NAME

newsadmconfig – defines the configuration for Sun Internet News Server administration

SYNOPSIS

`/etc/opt/SUNWixsna/admin/state/newsadmconfig`

DESCRIPTION

The `newsadmconfig` file stores configuration settings for the Sun™ Internet News Server™ administration server.

You should not edit this file directly. The settings in this file can be edited using Sun Internet News Server administration through the Sun Internet Administrator console.

EXTENDED DESCRIPTION

Each line in the file is in the form

keyword: value

Lines beginning with a hash mark (#) are ignored.

The following keywords are used to store the Sun Internet News Server administration configuration:

`administrator`	Defines the e-mail address of the news server administrator. Messages generated by the news server processes on this machine will be sent to the named address.
`IsSlave`	Defines whether this server is a slave or a peer of the server named in `slaveOf` if the local server is configured as a news feeder or reader/feeder server. Valid values are 0 if this server is a peer, and 1 if this server is a slave.
`slaveOf`	Defines the fully-qualified domain name of the host that feeds this news feeder or reader/feeder server. The host named here may be a master (if `IsSlave` is set to 1) or a peer.
`StorageLoc`	Defines the directory where articles will be stored. The default value is `/var/news/storage`.
`StateLoc`	Defines the directory where state information (such as the history and active files) is stored. The default value is `/var/news/state`.
`LogsLoc`	Defines the directory where news server logs are stored. The default value is `/var/news/logs`.

`ReaderRhost` — Defines the remote server from which this news server gets its news feed, if the local server is a news reader server.

`RstorageMount` — Defines the location where the remote feed server (`ReaderRhost`) storage directory will be mounted locally. The default value is `/var/news/storage`.

`RstateMount` — Defines the location where the remote feed server (`ReaderRhost`) state directory will be mounted locally. The default value is `/var/news/state`.

`RconfigMount` — Defines the location where the remote feed server (`ReaderRhost`) configuration directory will be mounted locally. The default value is `/var/news/config`.

`FeedConnections` — Defines whether this server will accept connections to feed news, if it is configured as a news feeder server. Valid values are O to deny connections or 1 to accept connections.

EXAMPLES

EXAMPLE 1

The following example shows a default configuration for a news reader/feeder server:

```
## newsadmconfig ##
administrator:  newsadmin@myISP.net
IsSlave:  0
slaveOf:  news1.myISP.net
StorageLoc:  /var/news/storage
StateLoc:  /var/news/state
LogsLoc:  /var/news/logs
ReaderRhost:
RstorageMount:
RstateMount:
RconfigMount:
FeedConnections:
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWixsna
Interface Stability	Evolving

NAME

newsfeeds – specifies where Usenet articles get sent

DESCRIPTION

The file `/var/news/config/newsfeeds` specifies how incoming articles should be distributed to other sites. It is parsed by the news feeder daemon when it starts up, or when directed to by **`ctlinnd`**(1m).

The file is interpreted as a set of lines according to the following rules. If a line ends with a backslash, then the backslash, the newline, and any white space at the start of the next line is deleted. This is repeated until the entire ‘‘logical’’ line is collected. If the logical line is blank, or starts with a number sign (‘‘#’’), it is ignored.

All other lines are interpreted as feed entries. An entry should consist of four colon-separated fields; two of the fields may have optional subfields, marked off by a slash. Fields or subfields that take multiple parameters should be separated by a comma. Extra white space can cause problems. Except for the site names, case is significant. The format of an entry is:

```
sitename[/exclude,exclude,...]\
 :pattern,pattern...[/distrib,distrib...]\
 :flag,flag...\
 :param
```

Each field is described below.

The *sitename* is the name of the site to which a news article can be sent. It is used for writing log entries and for determining if an article should be forwarded to a site. If *sitename* already appears in the article’s Path header, then the article will not be sent to the site. The name is usually whatever the remote site uses to identify itself in the Path line, but can be almost any word that makes sense; special local entries (such as archivers or gateways) should probably end with an exclamation point to make sure that they do not have the same name as any real site. For example, ‘‘gateway’’ is an obvious name for the local entry that forwards articles out to a mailing list. If a site with the name ‘‘gateway’’ posts an article, when the local site receives the article it will see the name in the Path and not send the article to its own ‘‘gateway’’ entry. See also the description of the ‘‘Ap’’ flag, below. If an entry has an exclusion subfield, then the article will not be sent to that site if any of the names specified as *exclude*s appear in the Path header. The same *sitename* can be used more than once. The appropriate action will be taken for each entry that should receive the article, regardless of the name, although this is recommended only for program feeds to avoid confusion. Case is not significant in site names.

The *patterns* specify which groups to send to the site and are interpreted to build a ‘‘subscription list’’ for the site. The default subscription is to get all groups. The patterns in the field are wild card-style patterns, and are matched in order against the list of newsgroups that the local site receives. If the first character of a pattern is an exclamation mark, then any groups matching the pattern are removed from the subscription, otherwise any matching groups are added. For example, to receive all ‘‘comp’’ groups, but only comp.sources.unix within the sources newsgroups, the following set of patterns can be used:

```
comp.*,!comp.sources.*,comp.sources.unix
```

Three things to note about this example are: The trailing ‘‘.*’’ is required. Second, the result of the last match is the most important. Third, ‘‘comp.sources.*’’ could be written as ‘‘comp.sources*’’ but this would not have the same effect if there were a ‘‘comp.sources-only’’ group.

There is also a way to subscribe to a newsgroup negatively. That is to say, do not send this group even if the article is cross-posted to a subscribed newsgroup. If the first character of a pattern is an atsign ‘‘@’’, it means that any article posted to a group matching the pattern will not be sent even though the article may be cross-posted to a group which is subscribed. The same rules of precedence apply in that the last match is the one which counts. For example, if you want to prevent all articles posted to any "alt.binaries.warez" group from being propagated even if it is cross-posted to another "alt" group or any other group for that matter, then the following set of patterns can be used:

```
alt.*,@alt.binaries.warez.*,misc.*
```

If you reverse the alt.* and alt.binaries.warez.* patterns, it would nullify the atsign because the result of the last match is the one that counts. Using the above example, if an article is posted to one or more of the alt.binaries.warez.* groups and is cross-posted to misc.test, then the article is not sent.

A subscription can be further modified by specifying ‘‘distributions’’ that the site should or should not receive. The default is to send all articles to all sites that subscribe to any of the groups where it has been posted , but if an article has a Distribution header and any *distrib*s are specified, then they are checked according to the following rules:

1. If the Distribution header matches any of the values in the subfield, then the article is sent.
2. If a distrib starts with an exclamation point, and it matches the Distribution header, then the article is not sent.
3. If the Distribution header does not match any distrib in the site's entry, and no negations were used, then the article is not sent.
4. If the Distribution header does not match any distrib in the site's entry, and any distrib started with an exclamation point, then the article is sent.

If an article has more than one distribution specified, then each one is according to the above rules. If any of the specified distributions indicates that the article should be sent, it is; if none do, it is not sent — the rules are used as a ''logical or.'' It is a mistake to have a single feed that specifies distributions that start with an exclamation point along with some that don't.

Distributions are text words, not patterns; entries like ''*'' or ''all'' have no special meaning.

The *flags* parameter specifies miscellaneous parameters. They may be specified in any order; flags that take values should have the value immediately after the flag letter with no white space. The valid flags are:

`<size` An article will only be sent to the site if it is less than `size` bytes long. The default is no limit.

`>size` An article will only be sent to the site if it is greater than `size` bytes long. The default is no limit.

A***checks*** An article will only be sent to the site if it meets the requirements specified in the *checks*, which should be chosen from the following set:

```
d Distribution header required
p Do not check Path header for the sitename before
  propagating (the exclusions are still checked).
```

B***high/low*** If a site is being fed by a file, channel, or exploder (see below), the server will normally start trying to write the information as soon as possible. Providing a buffer may give better system performance and help smooth out overall load if a large batch of news comes in. The value of this flag should be two numbers separated by a slash. The first specifies the point at which the server can start draining the feed's I/O buffer, and the second specifies when to stop writing and begin buffering again. The units are bytes. The default is to do no buffering, sending output as soon as it is possible to do so.

F*name* This flag specifies the name of the file that should be used if it is necessary to begin spooling for the site (see below). If *name* is not an absolute pathname, it is taken to be relative to `/var/news/storage/out.going`. Then, if the destination is a directory, the file *togo* in that directory will be used as filename.

G*count* If this flag is specified, an article will only be sent to the site if it is posted to no more than *count* newsgroups.

H*count* If this flag is specified, an article will only be sent to the site if it has *count* or fewer sites in its Path line. This flag should only be used as a rough guide because of the loose interpretation of the Path header; some sites put the poster's name in the header, and some sites that might logically be considered to be one hop become two because they put the posting workstation's name in the header. The default value for *count* is one.

`Isize` The flag specifies the size of the internal buffer for a file feed. If there are more file feeds then allowed by the system, they will be buffered internally in least-recently-used order. If the internal buffer grows bigger then `size` bytes, however, the data will be written out to the appropriate file. The default value is (16 * 1024) bytes.

N*modifiers* The newsgroups that a site receives are modified according to the *modifiers*, which should be chosen from the following set:

```
m Only moderated groups
u Only unmoderated groups
```

`Ssize` If the amount of data queued for the site gets to be larger than `size` bytes, then the server will switch to spooling, appending to a file specified by the ''F'' flag, or `/var/news/storage/out.going/` *sitename* if the ''F'' flag is not specified. Spooling usually happens only for channel or exploder feeds.

`Ttype` This flag specifies the type of feed for the site. *Type* should be a letter chosen from the following set:

```
c Channel
f File
l Log entry only
m Funnel (multiple entries feed into one)
p Program
x Exploder
```

Each feed is described below in the section on *feed types*. The default is *Tf*.

W*items* If a site is fed by file, channel, or exploder, this flag controls what information is written. If a site is fed by a program, only the asterisk (‘‘*’’) has any effect. The *items* should be chosen from the following set:

```
 b Size of the article in bytes
 f Article's full pathname
 g Newsgroup the article is in;
  if cross-posted, then the first of the groups this
  site gets
 m Article's Message-ID
 n Article's pathname relative to the spool directory
 p Time the article was posted as seconds since epoch.
 s Site that fed the article to the server;
  from the Path header
 s IP address of the site that sent the article
 t Time article was received as seconds since epoch
 * Names of the appropriate funnel entries;
  or all sites that get the article
 D Value of the Distribution header;
  ? if none present
 H All headers
 N Value of the Newsgroups header
 O Overview data
 R Information needed for replication
```

More than one letter can be used; the entries will be separated by a space, and written in the order in which they are specified. The default is *Wn*. The ‘‘H’’ and ‘‘O’’ items are intended for use by programs that create news overview databases. If ‘‘H’’ is present, then all of the article’s headers are written followed by a blank line. An Xref header (even if one does not appear in the filed article) and a Bytes header, specifying the article’s size, will also be part of the headers. If used, this should be the only item in the list; if preceeded by other items, however, a newline will be written before the headers. The ‘‘O’’ generates input to the **overchan**(1m) program. It, too, should be the only item in the list. The asterisk has special meaning. It expands to a space-separated list of all sites that received the current article. If the site is the target of a funnel however (that is, it is named by other sites which have a ‘‘Tm’’ flag), then the asterisk expands to the names of the funnel feeds that received the article. If the site is fed by a program, then an asterisk in the *param* field will be expanded into the list of funnel feeds that received

the article. A site fed by a program cannot get the site list unless it is the target of other ''Tm'' feeds.

The interpretation of the *param* field depends on the type of feed, and is explained in more detail below in the section on *feed types*. It can be omitted.

The site named *ME* is special. There should only be one such entry, and it should be the first entry in the file. If the *ME* entry has a subscription list, then that list is automatically prepended to the subscription list of all other entries. For example, ''*,!control,!junk,!foo.*'' can be used to set up the initial subscription list for all feeds so that local postings are not propagated unless ''foo.* explicitly appears in the site's subscription list. Note that most subscriptions should have ''!junk,!control'' in their pattern list. Unlike other news software, it does not affect what groups are received; that is done by the **active**(4)

If the *ME* entry has a distribution subfield, then only articles that match the distribution list are accepted; all other articles are rejected. A commercial news server, for example, might have ''/!local'' to reject local postings from other, misconfigured, sites.

FEED TYPES

The news feeder daemon provides four basic types of feeds: log, file, program, and channel. An exploder is a special type of channel. In addition, several entries can feed into the same feed; these are funnel feeds that refer to an entry that is one of the other types. Note that the term ''feed'' is technically a misnomer, since the server does not transfer articles, but reports that an article should be sent to the site.

The simplest feed is one that is fed by a log entry. Other than a mention in the news logfile, no data is ever written out. This is equivalent to a ''Tf'' entry writing to `/dev/null` except that no file is opened.

A site fed by a file is the simplest type of feed. When the site should receive an article, one line is written to the file named by the *param* field. If *param* is not an absolute pathname, it is taken to be relative to `/var/news/storage/out.going`. If empty, the filename defaults to `/var/news/storage/out.going/` sitename. This name should be unique.

When a site fed by a file is flushed (see **ctlinnd**(1m)), the following steps are performed. The script doing the flush should have first renamed the file. The server tries to write out any buffered data, and then closes the file. The renamed file is now available for use. The server will then reopen the original file, which will now get created.

A site fed by a program has a process spawned for every article that the site receives. The *param* field must be a **sprintf**(3) format string that may have a single *%s* parameter, which will be given a pathname for the article, relative to the news spool directory. The full path name may be obtained by prefixing the

%s in the *param* field by the news spool directory prefix. Standard input will be set to the article or `/dev/null` if the article cannot be opened for some reason. Standard output and error will be set to the error log. The process will run with the user and group ID of the `/var/news/logs` directory. The news feeder daemon will try to avoid spawning a shell if the command has no shell meta-characters; this feature can be defeated by appending a semicolon to the end of the command. The full pathname of the program to be run must be specified; for security, PATH is not searched.

If the entry is the target of a funnel, and if the ''W*'' flag is used, then a single asterisk may be used in the *param* field where it will be replaced by the names of the sites that fed into the funnel. If the entry is not a funnel, or if the ''W*'' flag is not used, then the asterisk has no special meaning.

Flushing a site fed by a program does not perform any action.

When a site is fed by a channel or exploder, the *param* field names the process to start. Again, the full pathname of the process must be given. When the site is to receive an article, the process receives a line on its standard input telling it about the article. Standard output and error, the user ID, and group ID of the all subprocess are set as for a program feed, above. If the process exits, it will be restarted. If the process cannot be started, the server will spool input to a file named `/var/news/storage/out.going/` sitename. It will then try to start the process some time later.

When a site fed by a channel or exploder is flushed, the server closes down its end of the pipe. Any pending data that has not been written will be spooled; see the description of the ''S'' flag, above. No signal is sent; it is up to the program to notice EOF on its standard input and exit. The server then starts a new process.

Exploders are a superset of channel feeds. In addition to channel behavior, exploders can be sent command lines. These lines start with an exclamation point, and their interpretation is up to the exploder. The following messages are generated automatically by the server:

```
newgroup group
rmgroup group
flush
flush site
```

These messages are sent when the **`ctlinnd`**(1m) command of the same name is received by the server. In addition, the ''send'' command can be used to send an arbitrary command line to the exploder child-process. The primary exploder is **`buffchan`**(1m).

Funnel feeds provide a way of merging several site entries into a single output stream. For a site feeding into a funnel, the *param* field names the actual entry that does the feeding.

For more details on setting up different types of news feeds, see the INN installation manual.

EXAMPLES

EXAMPLE 1

```
## Initial subscription list and our distributions.
ME:*,!junk,!foo.*/world,usa,na,ne,foo,ddn,gnu,inet::
## Feed all moderated source postings to an archiver
source-archive!:!*,*sources*,!*wanted*,!*.d\\
 :Tc,Wn:/opt/SUNWsns/bin/archive -f -i \\
 /usr/spool/news.archive/INDEX

## Watch for big postings
watcher!:*\\
 :Tc,Wbnm\\
 :exec awk '$1 > 1000000 { print "BIG", $2, $3 }' >/dev/console

## A UUCP feed, where we try to keep the "batching" between 4 and 1K.
ihnp4:/world,usa,na,ddn,gnu\\
 :Tf,Wnb,B4096/1024:

## Usenet as mail; note ! in funnel name to avoid Path conflicts.
## Can't use ! in "fred" since it would like look a UUCP address.
fred:!*,comp.sources.unix,comp.sources.bugs\\
 :Tm:mailer!
```

The last two sets of entries show how funnel feeds can be used. For example, the *nntpfanout* program would receive lines like the following on its standard input:

```
<123@litchi.foo.com> comp/sources/unix/888 nic.near.net uunet.uu.net
<124@litchi.foo.com> ne/general/1003 nic.near.net
```

EXAMPLE 2 Since the UUCP funnel is only destined for one site, the asterisk is not needed and entries like the following will be written into the file:

```
<qwe#37x@snark.uu.net> comp/society/folklore/3
<123@litchi.foo.com> comp/sources/unix/888
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO sprintf(3), active(4), buffchan(1m), ctlinnd(1m)

NAME

newslog – describes the Usenet log files

DESCRIPTION

Most log files created by Sun™ Internet News Server™ programs reside in the `/var/news/logs` directory and have a ‘‘.log’’ extension. Several versions are usually kept with an additional extension such as ‘‘.1’’, ‘‘.2’’, etc. — the higher the number, the older the log. The older versions are compressed.

The *scanlogs* script and related utilities (see **`newslog`**(1m) are responsible for rotating and compressing these files.

Some log files always have data, others only have data if there is a problem, and others are only created if a particular program is used or configuration parameter is set. The *innstat* script (see **`newslog`**(1m) monitors the size of all log files.

The following files will only accumulate data under the direction of **`control.ctl`**(4):

control.log miscctl.log newgroup.log rmgroup.log unwanted.log

In order to create these files, the ‘‘message’’ and ‘‘action’’ fields of *control.ctl* should be chosen from the following table:

```
Message Action Meaning
all log=miscctl Log all messages by default.
default log=miscctl Log unknown messages.
newgroup doit=newgroup Create group and log message.
newgroup log=newgroup Log message.
rmgroup doit=rmgroup Remove group and log message.
rmgroup log=rmgroup Log message.
‘‘other’’ doit=miscctl log and process the message.
‘‘other’’ log=miscctl Log message.
```

Here, ‘‘other’’ refers to any other control message such as:

checkgroups ihave sendme sendsys senduuname version

The following is a list of log files.

control.log This file maintains a count of the number of newgroup and rmgroup control messages seen for each newsgroup. The count is of the number of control messages with identical arguments, regardless of whether or not they were actually processed. All control arguments, including invalid ones, are counted. This file is updated by *tally.control*, which is

invoked by *scanlogs* if either the newgroup or rmgroup logs exist. This file is not rotated.

errlog This file contains the standard output and standard error of any program spawned by the news feeder daemon. The most common programs are the control-message handlers found in `/usr/news/bin/control`. This file should be empty. **`scanlogs`**(1m) will print the entire contents of this log file if it is non-empty.

expire.log By default, when **`news.daily`**(1m) is going to expire old news articles, it writes the date to this file, followed by any output from **`expire`**(1m) and the ending date. All lines but the first are indented four spaces.

miscctl.log When **`control.ctl`**(4) is configured as described above, all control messages except newgroup and rmgroup are appended to this file by **`writelog`**(1m). A summary line is generated describing the message and the action taken, followed by the article indented by four spaces, and a blank line.

newgroup.log When **`control.ctl`**(4) is configured as described above, all newgroup messages are appended to this file using the same format as for *miscctl.log*.

`news` logs articles received by the feeder daemon. **`scanlogs`**(1m) summarizes the rejected articles reported in this file.

news.crit All critical error messages issued by the feeder daemon are appended to this file. This log file should be empty. **`scanlogs`**(1m) will print the entire contents of this log file if it is non-empty. You should have the following line in your **`syslog.conf`**(4) file:

```
news.crit /var/news/logs/news.crit
```

news.err All major error messages issued by the feeder daemon are appended to this file via **`syslog`**(3). This log file should be empty. **`scanlogs`**(1m) will print the entire contents of this log file if it is non-empty. You should have the following line in your **`syslog.conf`**(4) file:

```
news.err /var/news/logs/news.err
```

news.notice All standard error messages and status messages issued by the feeder daemon are appended to this file via **syslog**(3). **scanlogs**(1m) uses the **awk**(1) script *innlog.awk* to summarize this file. You should have the following line in your syslog.conf file:

news.notice /var/news/logs/news.notice

nntpsend.log The **nntpsend**(1m) program appends all status messages to this file.

rmgroup.log When **control.ctl**(4) is configured as described above, all rmgroup messages are appended to this file using the same format as for *miscctl.log*.

unwanted.log maintains a count of the number of articles that were rejected because they were posted to newsgroups that do not exist at the local site. This file is updated by **tally.unwanted**(1m) and maintained in reverse numeric order (the most popular rejected group first). This file is not rotated.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

control.ctl(4), ctlinnd(1m), expire(1m), news.daily(1m), nntpsend(1m), newslog(1m), snsnews(1m)

NAME

nnrp.access – specifies access control for NNTP sites

DESCRIPTION

The file `/var/news/rconfig/nnrp.access` specifies the access control for those NNTP sites that are not handled by the the news feeder daemon. The **snsd**(1m) server reads it when first spawned by the feeder daemon. .

Comments begin with a number sign (‘‘#’’) and continue through the end of the line. Blank lines and comments are ignored. All other lines should consist of five fields separated by colons.

If you are using standard UNIX authentication:

```
hosts:perms:username:password:patterns
```

You may use the plus symbol (+) in the username field to mean “any valid UNIX user.” In this case, the password field is ignored, and the password supplied by the user is matched instead against a system lookup of the password for the supplied user name.

If you are using LDAP authentication, the *password* field is ignored, and the *username* field must contain `/pam/`:

```
hosts:perms:/pam/:password:patterns
```

The first field is a wild card-style pattern specifying the names or Internet address of a set of *hosts*. Before a match is checked, the client’s hostname (or its Internet address) is converted to lowercase. Each line is matched in turn, and the last successful match is taken as the correct one.

The second field is a set of letters specifying the permissions granted to the client. The *perms* should be chosen from the following set:

```
R The client can retrieve articles
P The client can post articles
```

The third and fourth fields specify the *username* and *password* that the client must use to authenticate itself before the server will accept any articles.

If you are using UNIX authentication, note that no authentication (other then a matching entry in this file) is required for newsreading. If they are empty, then no password is required. Whitespace in these fields will result in the client being unable to properly authenticate itself, and may be used to disable access. If you are using LDAP authentication, the *password* field is ignored.

The fifth field is a set of patterns identifying the newsgroups that the client is allowed to access. The patterns are interpreted in the same manner as the **newsfeeds**(4) file. The default, however, denies access to all groups.

The access file is normally used to provide host-level access control for reading and posting articles. There are times, however, when this is not sufficient and user-level access control is needed. Whenever an NNTP ''authinfo'' command is used, the **snsd**(1m) server re-reads this file and looks for a matching username and password. If the local newsreaders are modified to send the ''authinfo'' command, then all host entries can have no access and specific users can be granted the appropriate read and post access.

For example:

```
## host:perm:user:pass:groups
## Default is no access.
*:: -no- : -no- :!*
## FOO hosts have no password, can read anything.
*.foo.com:Read Post:::*
## A related workstation can't access FOO newsgroups.
lenox.foo.net:RP:martha:hiatt:*,!foo.*
## LDAP authentication is to be used for public access
public.foo.net:RP:/pam/::
```

If the file contains passwords, it should not be world-readable.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

newsfeeds(4), isppammod(1m), snsd(1m), snsnews(1m)

NAME

nntpsend.ctl – specifies the list of sites to feed via nntpsend

DESCRIPTION

The file `/var/news/config/nntpsend.ctl` specifies the default list of sites to be fed by **nntpsend**(1m).

Comments begin with a number sign (‘‘#’’) and continue through the end of the line. Blank lines and comments are ignored. All other lines should consist of four fields separated by a colon.

The first field is the name of the site as specified in the **newsfeeds**(4) file.

The second field should be the hostname or IP address of the remote site.

The third field, if non-empty, specifies the default tail truncation size of site’s batch file. If this field is empty, no truncation is performed. If this field is of the form ‘‘maxsize-truncsize’’ then it is passed as ‘‘–m maxsize –s truncsize’’, otherwise if it is of the form ‘‘truncsize’’ then it is passed as ‘‘–s truncsize’’.

The fourth field specifies some default flags passed to **innxmit**(1m) The flag ‘‘–a’’ is always given to *innxmit* and need not appear here. If no ‘‘–t timeout’’ flag is given in this field and on the **nntpsend**(1m) command line, ‘‘–t 180’’ will be given to *innxmit*.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

innxmit(1m), newsfeeds(4), nntpsend(1m), trunc(1).

NAME

overview.fmt – the format of the news overview database

DESCRIPTION

The file `/var/news/config/overview.fmt` specifies the organization of the news overview database. Blank lines and lines beginning with a number sign (‘‘#’’) are ignored. The order of lines in this file is important; it determines the order in which the fields will appear in the database.

Most lines will consist of an article header name, optionally followed by a colon. A trailing set of lines can have the word ‘‘full’’ appear after the colon; this indicates that the header should appear as well as its value.

If this file is changed, it is usually necessary to rebuild the existing overview database using **expireover**(1m) after removing all existing overview files.

The default file, shown below, is compatible with Geoff Collyer’s ‘‘nov’’ package:

```
Subject:
From:
Date:
Message-ID:
References:
Bytes:
Lines:
## Some newsreaders get better performance if Xref is present
#Xref:full
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

NAME

passwd.nntp – the passwords for connecting to remote NNTP servers

DESCRIPTION

The file `/var/news/config/passwd.nntp` contains host-name-password triplets for use when authenticating client programs to NNTP servers. This file is normally interpreted by the *NNTPsendpassword* routine. Blank lines and lines beginning with a number sign (‘‘#’’) are ignored. All other lines should consist of three or four fields separated by colons:

```
host:name:password
host:name:password:style
```

The first field is the name of a host, and is matched in a case-insensitive manner. The second field is a user name, and the third is a password. The optional fourth field specifies the type of authentication to use. The default is ‘‘authinfo’’ which means that NNTP ‘‘authinfo’’ commands are used to authenticate to the remote host. If either the username or password are empty, then the related command will not be sent. (The ‘‘authinfo’’ command is a common extension to RFC 977.)

For example:

```
## UUNET needs a password, MIT doesn't.
mit.edu:bbn::authinfo
uunet.uu.net:bbn:yoyoma:authinfo
```

This file should not be world-readable.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO

snsnews(1m)

NAME

sns.conf – Sun™ Internet News Server™ configuration data

DESCRIPTION

`/var/news/rconfig/sns.conf` enables you to specify the maximum number threads per NNTP handling process, and the maximum number of NNTP processes for the platform on which the server is running.

Options

numthreads The maximum number of threads per NNTP handling process. *value* must be a number greater than 0 or the keyword "`system`". If you specify keyword `system`, the server automatically optimizes the number of threads per process for the platform on which it is running.

numprocs The maximum number of NNTP handling processes. *value* must be a number greater than 0 or the keyword "`system`". If you specify keyword `system`, the server automatically optimizes the number of NNTP handling processes for the platform on which it is running.

EXAMPLES

EXAMPLE 1

```
##   After making any changes to this file, send a SIGHUP to the
##   main snsd process for it to recognize the changes.
##      kill -HUP ‘cat /var/news/rstate/snsd.pid‘
##
## Allow the server to optimize threads per process and
## processes per platform.
##
numthreads:       system
numprocs:        system
```

EXAMPLE 2

```
##   After making any changes to this file, send a SIGHUP to the
##   main snsd process for it to recognize the changes.
##      kill -HUP ‘cat /var/news/rstate/snsd.pid‘
##
## Allow no greater than 20 threads per process and
## 50 processes per platform.
##
numthreads:       20
numprocs:        50
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWsns
Interface Stability	Unstable

SEE ALSO `snsd(1)`

PART V Sun WebServer

Sun™ WebServer™ 2.1 man pages. These are installed in `/usr/share/man/`.

CHAPTER 2

man Pages(1m): Maintenance Commands

NAME

htIntro – introduction to the man pages for the Sun™ WebServer™ command-line utilities. The man pages offer detailed instructions and examples on options and subcommands for each utility.

DESCRIPTION

The command-line utilities are available to administer Sun WebServer.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhtman
Interface Stability	Evolving

FILES

The following files are used by the command-line utilities:

access.conf — Configures a web site's access control lists (ACLs) . Located at *site_path*/conf/access.conf.

access.conf — Configures the server administration ACLs. Located at /etc/http/access.conf.

content.conf — Defines the content variants, encoding types, and directory preferences for a web site. Located at *site_path*/conf/content.conf.

httpd-instances.httpd.conf — Tracks all Sun WebServer instances. When htserver creates a new server instance, an entry is added to this file. Located in /etc/http/.

***site_name*.site.conf** — Contains the web site servlet engine configuration if the servlet engine is not shared. Located at *site_path*/conf/*site_name*.site.conf.

***instance_name*.httpd.conf** — Defines the server instance's configuration. When hthost adds a new site, it creates an entry in httpd.conf to define the *site_path* and web site configuration file. Located in /etc/http/.

map.conf	Creates an alias to a path on the file system or a redirection to a remote URL from a Uniform Resource Identifier (URI) on the host. Located at *site_path*/conf/map.conf.
realms.conf	Defines realms of user and group information used by access control lists on a Sun WebServer web site. Located at *site_path*/conf/realms.conf.
servlets.properties	Defines each servlet that can be loaded by a web site. Located in *site_path*/conf/.

SEE ALSO

access.conf(4), **content.conf**(4), **httpd.conf**(4), **httpd.cgi.logs**(4), **httpd.event.logs**(4), **httpd-instances.httpd.conf**(4), **httpd.request.logs**(4), **httpd.servlet.logs**(4), **httpd.site.conf**(4), **map.conf**(4), **realms.conf**(4), **servlets.properties**(4)

NOTES

/usr/bin/htaccess	Adds or deletes ACLs for resources on a web site. An ACL applies to any token that can be a URI on that site, whether that URI is a directory, file, servlet, CGI, or alias to another resource.
/usr/bin/htcontent	Create or deletes information about the content of resources on a web site. For directories, you can set whether they are available to browse and specify the format for directory listings. For files, you can set preferences for HTTP 1.1 content negotiation. Preferences include character set, language, compression encoding, and media type.
/usr/bin/hthost	Associates a web site (or virtual host) with system resources, such as the server instance that hosts the site, a

	configuration directory, a configuration file, and a host name. Activates or shuts down web sites on a running server instance.
/usr/bin/htmap	Adds and deletes aliases from one URI to another resource on a web site. Redirects a token to nonfile resources such as servlets, CGI scripts, or Sun WebServer GUI.
/usr/bin/htpasswd	Changes passwords for users in `HTPASSWD` realms. It is provided as a tool that can be incorporated in CGI or other scripts to automate password maintenance. User must be created using `htrealm`. Once a user is created, any system user can run `htpasswd` to update passwords (as long as the realm administrator name and password are specified).
/usr/bin/htrealm	Creates, deletes, and lists realm definitions for use with ACLs. It can also be used to manage users and groups in `HTPASSWD` realms.
/usr/bin/htserver	Creates and maintains Sun WebServer server instances. Each server instance is a process associated with a configuration file, and each one hosts one or more web sites. `htserver` can start, stop, and restart server instances. It can also enable or disable server instances.
/usr/bin/htservlet	Configures the behavior of a servlet engine. Defines and modifies servlet engine runtime, security, and logging properties; adds or removes entries in the `servlets.properties` file, and loads, reloads, or unloads servlets in running servlet engines.

/usr/lib/httpd	Server instances can be started or stopped by using the `htserver` utility, through the Sun WebServer GUI, or executing this script. It is recommended that you use `htserver` or the Sun WebServer GUI.

NAME

/usr/bin/htaccess – Allows manipulation of Sun™ WebServer™ access control lists (ACLs) configuration.

SYNOPSIS

htaccess add [–a][–g *group_name*|–I *Internet_host*|–u *username*] –h *hostname* –i *instance* [–m *method*] [–r *realm_name*] [–s *scheme_name*] –U *URI_name* [– y|–n] [–z *admin*[–p]]

htaccess check [–a][–g *group_name*|–I *Internet_host*|–u *user_name*] –h *hostname* –i *instance* [–m *method*] –U *URI_name* [–z *admin*[–p]]

htaccess delete [–a][–g *group_name*|–I *Internet_host*|–u *user_name*] –h *hostname* –i *instance* [–m *method*] –U *URI_name* [–y|–n] [–z *admin*[–p]]

htaccess help

htaccess list –h *hostname* –i *instance* –U *URI_name* [–z *admin*[–p]]

htaccess version

DESCRIPTION

`htaccess` adds or deletes access control lists (ACLs) for resources on a web site. An ACL applies to any token that can be a Uniform Resource Identifier (URI) on that site, whether that URI is a directory, file, servlet, CGI, or alias to another resource.

ACLs can restrict access by the host name or IP address of a client, or by an authenticated user or group name. Users are authenticated against a named realm (see **`htrealm`**(1m)), which defines whether passwords are stored in a Sun WebServer specific file (HTPASSWD), the operating system (UNIXSYS), in the Solaris ISP Server™ environment, an LDAP directory of ISP subscribers (ISP), or in the Sun™ Internet Administrator ™ for Solaris ISP Server (ISPADMIN).

The `list` subcommand can be used to list the current ACLs on a URI.

The `check` subcommand can be used to check whether a user name and password can access a URI.

OPTIONS

Subcommands

The following subcommands are supported:

add	Adds a new ACL or permission to an existing ACL.
check	Checks if the specified access is allowed.
delete	Deletes an ACL or permission to an existing ACL.

help Displays help on usage.

list Lists all ACLs and their permissions for a given URL or specified host.

version Displays the version of `htaccess`.

Options

The following options are supported:

`–a` Specifies that the user or group is the administrator. Valid with the add and delete subcommands.

`–g` ***group_name*** Specifies the group in the realms to which the permissions apply. A group is a group of users defined in the realm. Use the wild card `\*` to indicate that the permission applies to any *group_name*. Valid with the add, delete, and check subcommands.

`–h` ***hostname*** Specifies the name of the virtual host containing the ACL. Valid with all subcommands.

`–I` ***Internet_host*** Specifies the IP or domain to which the permissions apply. *Internet_host* can be a fully qualified or partial domain name. If the domain name is partial, the permission applies to all hosts whose fully qualified names end with the domain. It can also be a fully qualified or partial IP address. If the IP address is partial, the permission applies to all hosts whose IP address begins with the *Internet_host*. Use the wild card `\*` to indicate that the permission applies to any *Internet_host*. Valid with the add, delete, and check subcommands.

`–i` ***instance*** Specifies the name of the httpd instance. Valid with all subcommands.

`–m` ***method*** Specifies the HTTP method name to which the permissions apply. The *method* directive is a list of `ALL`, `DELETE`, `GET`, `POST`, and `PUT`. Select ALL to permit all HTTP methods. Separate multiple methods with a space. The default is ALL. Valid with the add, delete, and check subcommands.

`–n` Denies access permission to the named user, group, or host. Valid with the add and delete subcommands.

`–p` Turns off prompting of password such that passwords are taken in from stdin and scripts may pipe (|) passwords. Valid with all subcommands.

`–r` ***realm_name*** Specifies the realm name. Valid with the add subcommand.

`–s` ***scheme_name*** Specifies the authentication scheme.

BASIC	The server expects user name and password information in base64 encoded text.
MD5	The server expects user name and a message digest of the password. The server must get the password in base64 encoded text locally, create a message digest, and compare it to the digest sent by the client. Valid only with `htpasswd`.
NONE	The server does not expect any authentication.

Valid only with the add subcommand.

`–U` ***URI_name*** Specifies URI name protected by the ACL. Valid with all subcommands.

`–u` ***username*** Specifies the user name to which the permission applies. A user is any user with a user name for which Sun WebServer retrieves the password from the realm name specified in the ACL. Use the wild card `\*` to indicate that the permission applies to any user. Valid with the add, check, and delete subcommands.

`–v` Specifies the verbose mode. Valid with all subcommands.

`–y` Allows the named user, group, or host to access the URI. Valid with the add and delete subcommands.

`–z` ***admin_name*** Specifies the name of the administrator. Valid with the add, check, and delete subcommands.

EXAMPLES

EXAMPLE 1

To protect the URL `http://www.A.com/project/` on the server instance “sws_server” using an HTPASSWD realm “Project” with user “user1”:

```
# htrealm add -i sws_server -h www.A.com -r Project \
-s HTPASSWD -d realms/Project
# htrealm add -i sws_server -h www.A.com -r Project -u user1
Setting password for the user user1.
Password:
Confirm Password:

# htaccess add -i sws_server -h www.A.com -U "/project" \
 -r Project -s BASIC -m GET -u '*' -y
```

EXAMPLE 2

To delegate access control management to the user web master in the realm `WebUsers`:

```
# htaccess add -i sws_server -h www.A.com -U / -r WebUsers -s MD5 -a \
 -u webmaster
```

EXAMPLE 3

To use `htaccess` as a user other than root:

```
% htaccess add -i sws_server -h www.A.com -U / -I .domain.A.com -m PUT -z admin
Enter password for admin:
```

EXAMPLE 4

To use `htaccess` as a user other than root and read the administration password from a file `/tmp/tp/admin.pwd`:

```
% htaccess list -i sws_server -h www.A.com -U / -z admin -p < /tmp/tp/admin.pwd
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by this utility:

site_path`/conf/access.conf`

Configures a web site's access control lists.

`/etc/http/access.conf`

Configures the server administration access control lists.

site_path`/conf/realms.conf`

Defines the realms used to define users for web site access control lists.

`/etc/http/realms.conf`

Defines the realms used to define users for server administration.

SEE ALSO

`access.conf`(4), **`htIntro`**(1m), **`htrealm`**(1m), **`realms.conf`**(4)

NOTES

If the command is run by the `root` user, then the user name and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

`MD5` authentication can only be used with `HTPASSWD` realms.

NAME

/usr/bin/htcontent – Administers HTTP/1.1 meta-data associated with resources in a document tree.

SYNOPSIS

htcontent add –h *hostname* –i *instance* –n *uri* [–p] [–u *administrator*]–O *directory_options*|–P *preference_options*|–V *variant_options*

htcontent delete –h *hostname* –i *instance* –n *uri* [–p] [–u *administrator*]–O *directory_options*|–P *preference_options*|–V *variant_options*

htcontent help

htcontent list –h *hostname* –i *instance* –n *uri* [–p] –u *administrator*[–O *directory_options*|–P *preference_options*|–V *variant_options*]

htcontent version

DESCRIPTION

`htcontent` creates or deletes information about the content of resources on a web site.

You can set whether directories can be browsed and also sets the format for directory listings. You can also create a list of default file names to search for in the directory.

For files, you can set preferences for HTTP 1.1 content negotiation. Preferences include character set, language, compression encoding, and media type.

You can set variant information for a Uniform Resource Identifier (URI). If a URI has a set of associated file variants, the server will select the most appropriate variant based on the client's preferences and the preference settings on each file variant.

See the Examples section for more detail.

OPTIONS

Subcommands

The following subcommands are supported:

add	Adds directories, options, preferences, or variants for a URI.
delete	Deletes directories, options, preferences, or variants for a given URI.
help	Displays help on usage.
list	Lists the configured directories, options, preferences, or variants for a URI.

version Displays the version of htcontent.

Options

The following options are supported:

–h ***hostname*** Specifies the virtual host name. Required with all subcommands.

–i ***instance*** Specifies the name of the httpd instance. Required with all subcommands.

–n ***uri*** Specifies the URI. The URI must already exist. Required with all subcommands.

–O ***directory_options*** Specifies options for directory listings and default files on directory options. Valid with all subcommands. This option requires a comma-separated list of parameters to specify directory settings. Lists should be specified in order of preference. The following are the valid directory options:

a ***all*** Deletes all preference or variants information. Valid with the delete subcommand.

d **[=*listing_type*]** Sets the method for displaying the contents of a directory when there is no file matching one of the default file names. Valid with the add and delete subcommands. The *listing_type* directive can be one of the following:

`fancy` Displays each directory as a hyperlink with the file size, the last modified time, and

an icon next to each entry to indicate the file type.

`off` Displays no directory contents.

`simple` Displays directory entries as plain text hyperlinks.

`f` **[=** ***file*****]** Specifies the file associated with the variant information (–V) or a list of default file names in a directory (–0). Use a colon (:) to separate items in a list. File names must be relative to the URI. Valid with the add and delete subcommands.

–P ***preference_options*** Sets the server's content negotiation preferences for the specified URI. Valid with all subcommands. This option requires a comma-separated list of parameters to specify preference settings. Lists should be specified in order of preference. The following are the valid preference options:

`a` ***all*** Deletes all preference or variants information. Valid with the delete subcommand.

`c` **[=** ***charset*****]** Specifies the character set of the data. A character set refers to a method used with one or more tables to convert a sequence of octets into a sequence of characters. The default *charset* for variants is

		ISO-8859–1. Valid with the add and delete subcommands.
	e **[=** ***encoding*****]**	Specifies the preferred encodings or the encoding type of a variant. Encoding refers only to methods of compression. For example, `gzip` or `compress` reveals which methods have been used to encode the file. For preferences, separate multiple encodings with a colon (:). Valid with the add and delete subcommands.
	l **[=** ***lang*****]**	Specifies the preferred languages of a variant. Languages are specified in the standard two-letter format. For preferences, separate multiple languages with a colon (:). Valid with the add and delete subcommands.
	t **[=** ***media_type*****]**	Specifies the preferred types of media of a variant. Media type is in standard MIME type format. For preferences, separate multiple media types with a colon (:). Valid with the add and delete subcommands.
–p		Disables password prompting. Passwords will be read from stdin. Valid with the add and delete subcommands.
–u ***administrator***		Specifies a user name in the web site's administration realm (`serverAdmin` by default), or in the realm specified by the ACL from the URI. Required with all subcommands.

–V ***variant_options*** Specifies variant suboptions. Valid with all subcommands. This option requires a comma-separated list of parameters to specify variant settings. Lists should be specified in order of preference. The following are the valid variant options:

a ***all***	Deletes all preference or variants information. Valid with the delete subcommand.
c **[=** ***charset*****]**	Specifies the character set of the data. A character set refers to a method used with one or more tables to convert a sequence of octets into a sequence of characters. The default *charset* for variants is ISO-8859–1. Valid with the add subcommand.
e **[=** ***encoding*****]**	Specifies the preferred encodings or the encoding type of a variant. Encoding refers only to methods of compression. For example, `gzip` or `compress` reveals which methods have been used to encode the file. For preferences, separate multiple encodings with a colon (:). Valid with the add subcommand.
f **[=** ***file*****]**	Specifies the file associated with the variant information (–V) or a list of default file names in a directory (–0). Use a colon (:) to separate items in a list. File names must be relative to the URI. Valid with the add and delete subcommands.
l **[=** ***lang*****]**	Specifies the preferred languages of a variant. Languages are specified in the

standard two-letter format. For preferences, separate multiple languages with a colon (:). Valid with the add subcommand.

`t` **[=** ***media_type*****]** Specifies the preferred types of media of a variant. Media type is in standard MIME type format. For preferences, separate multiple media types with a colon (:). Valid with the add subcommand.

EXAMPLES

EXAMPLE 1

This example displays a page named `home.html` in English, French, or German based on the client's preference. There are three files: `home.en.html`, `home.fr.html`, and `home.de.html`.

```
% htcontent add -i sws_server -h www.A.com -n home.html \\
   -u admin -V f=home.en.html,l=en
 Enter Password for admin:
% htcontent add -i sws_server -h www.A.com -n home.html \\
   -u admin -V f=home.fr.html,l=fr
 Enter Password for admin:

% htcontent add -i sws_server -h www.A.com -n home.html \\
   -u admin -V f=home.de.html,l=de
 Enter Password for admin:
```

CODE EXAMPLE 1

To view the variants associated with a URI, use `htcontent list`. After Example 1, you could verify the content settings:

```
# htcontent list -i sws_server -h www.A.com -n home.html \\
  -u admin -V
Enter Password for admin:
  home.de.html    lang = de
  home.fr.html    lang = fr
  home.en.html    lang = en
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by this utility:

site_path`/conf/content.conf` Defines the content variants, encoding types, and directory preferences for a web site.

SEE ALSO

`content.conf`(4), **`htIntro`**(1m)

NOTES

If the command is run by `root` user, then the user name and password of an administrator are not required.

Users other than `root` must use the `–u` option and pass the user name and password of a valid administrator to the command.

NAME

/usr/bin/hthost – Create, delete, and manage web sites on Sun™ WebServer™ `httpd` instances.

SYNOPSIS

hthost add [–f *site_config_file*] [–g *groupname*] –h *hostname* –i *instance* [–p] –s *site_path* [–u *username*] –z *admin*

hthost delete –i *instance* –h *hostname* [–p] –z *admin*

hthost disable –i *instance* –h *hostname* [–p] –z *admin*

hthost enable –i *instance* –h *hostname* [–p] –z *admin*

hthost help[[add]|[delete]|[disable]|[enable]|[list]|[version]]

hthost list [–i *instance*–h *hostname*] [–p] –z *admin*

hthost version

DESCRIPTION

`hthost` is used primarily to add and delete web sites. The `add` subcommand associates a web site (or virtual host) with system resources, such as the server instance that hosts the site, a configuration directory, a configuration file, and a host name.

The `enable` and `disable` subcommands are used to activate or shut down web sites on a running server instance. The enabled or disabled state is saved so that if the server instance is restarted, enabled sites are automatically reactivated.

Once a web site has been created on a server instance, you must edit its configuration file to further customize the site; see **`httpd.site.conf`**(4).

OPTIONS

Subcommands

The first argument to `hthost` must be one of the following subcommands:

add Adds a web site configuration for a host name to a server instance. The `add` subcommand creates a site path for configuration directories, creates a configuration file, and grants administrative rights to the given user name and optional group. The initial state of the site is enabled.

delete Deletes a web site configuration from a server instance. References to the site in the configuration files for the server instance are deleted, and the site is no longer available through `httpd`. The site's directories and files remain in place.

disable Disables a web site. The server instance that serves the site will not respond to requests for disabled sites.

enable Enables a web site, making it available through the server instance.

help Displays usage information for the command.

list Lists the sites supported by a server or properties of a specific site. If only an instance name is supplied, a list of all web sites on that server displays. If an instance name and a host name are supplied, then details about the web site for the host name are displayed.

version Displays the version of the `hthost` command.

Subcommand Options

The following options are supported.

–f *site_config_file* Specifies the location of the site configuration file relative to the *site_path* specified by –s. By default, site configuration is stored in the *site_path*/conf directory. Valid only with the add subcommand.

[–g *group_name*] Specifies a group in the server administration realm that has ownership rights on the new site. Valid only with the add subcommand.

–h *hostname* Specifies the host name of the site to which a subcommand applies. The host name is a token used to identify the site; no name service lookups are performed, for example, to expand a host name to a fully qualified domain name. Any form of the host name may be used for the add subcommand, but other commands must use the same form as that used when the site was added. Valid with all subcommands.

–i *instance* Specifies the server instance that hosts the web site. The *instance* name is defined uniquely for each server when it is created. Valid with all subcommands.

–p Specifies the administrative password. Valid only with all subcommands.

–s ***site_path***	Specifies the absolute path to the web site's directory tree. The *site_path* contains all configuration, access control, realm, and content directories and files for the site. Valid only with the `add` subcommand.
–u ***username***	Specifies a user in the server administration realm that has ownership rights on the new site. Valid only with the `add` subcommand.

EXAMPLES

EXAMPLE 1

A server administrator named `serverAdmin1` creates a site named www.A.com. The site will have an administrator named `user1` and have all configuration files and public documents in `/opt/WWW/A.com/`.

In order for `hthost` to successfully create a directory for the new site, you must have write permission to the directory under which the site configuration files will reside.

```
# hthost add -i sws_server -h www.A.com -u user1 \\
  -s /opt/WWW/A.com -f conf/A.com.httpd.conf -z serverAdmin1
Enter Password for serverAdmin1:
Creating site directory:
 /opt/WWW/A.com
Creating site configuration:
 /opt/WWW/A.com/conf/A.com.httpd.conf

# ls /opt/WWW/A.com
cgi-bin/        public/
conf/           servlets/
```

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by this utility:

site_path`/conf/`*site_name*`.site.conf`

The web site configuration file.

`/etc/http/`*instance_name*`.httpd.conf`

The server instance configuration file. When `hthost` adds a new site, it creates an entry in `httpd.conf` to defined the *site_path* and web site configuration file.

SEE ALSO

`htIntro`(1m), **`httpd.site.conf`**(4), **`httpd.conf`**(4)

NOTES

If the command is run by `root` user, then the user name and password of an administrator are not required.

Users other than the `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

/usr/bin/htmap – Manages maps at the site level.

SYNOPSIS

htmap add [–c *class*] –i *instance* –f *from* –h *hostname* –t *to* [–z *admin*[–p]]

htmap delete –i *instance* –f *from* –h *hostname* [–z *admin*[–p]]

htmap help

htmap list –i *instance* [–f *from*] –h *hostname* [–z *admin*[–p]]

htmap version

DESCRIPTION

The `htmap` command adds, deletes, and lists aliases from one Uniform Resource Identifier (URI) to another resource on a web site. By default it creates a reference from a URI token to a file or directory on disk.

`htmap` administers maps to establish an alias to another resource, make a resource outside of the `doc_root` accessible to a client, or partition the name space into various classes of resources such as CGI, imagemap, or servlet.

OPTIONS

The following subcommands are supported:

add Adds a new map.

delete Deletes an existing map.

help Displays help on usage.

list Lists all maps.

version Displays the version of `htmap` command.

The following options are supported:

–c ***class*** Specifies the class file for the map. Values can be one of the following (if no –c is specified, then the class defaults to NULL):

Note - *class_type* is not case sensitive.

ADMIN Treats the alias as a URL to access Sun WebServer and its GUI.

CGI Treats the aliased file or directory as a CGI resource (all files located here will be treated as executable scripts).

	DOOR	Treats the aliased file or directory as a resource door. Resource doors are multithreaded server daemons that run independently of the web server. With resource doors, Sun WebServer is able to pass incoming requests on to user-developed programs through the Solaris doors mechanism. For more information on Sun WebServer resource doors, refer to the “Site URL Aliases Screen” section in the online help.
	IMAP	Treats the alias as an imagemap resource.
	NULL	Treats the aliased directory in no special way.
	REMOTE	Treats the alias as a new URL, either on the local host or on another network location.
	SERVLET	Treats the aliased *resource_target* as a servlet or a chain of servlets.
	STATS	Treats the alias as an interface to server statistics.
–f ***from***		Indicates the URI token the web server will map. Any URI that begins with this token will be redirected to the resource defined by the map class and the –t to destination.
–h ***hostname***		Specifies the virtual host.
–i ***instance***		Specifies the name of the httpd instance. Valid with all subcommands.
–t ***to***		Defines the path name or URL to the actual resource. Valid with the add subcommand.

EXAMPLES

EXAMPLE 1

To create a URL `http://www.A.com/swshelp/` that references a directory outside of the `www.A.com` document root:

```
# htmap add -i sws_server -h www.A.com -f /swshelp/ \
  -t /usr/http/admin_server/public/admin/help/en/
```

CODE EXAMPLE 1

To create a URL `http://www.A.com/siteadmin/` that starts the Sun WebServer GUI for administration of the web site:

```
# htmap add -i sws_server -h www.A.com -f /siteadmin/ \
  -t /sws-administration -c ADMIN
```

CODE EXAMPLE 2

This example shows how to create an alias that accesses a servlet without using the standard servlet token (`/servlet/` by default). To redirect `http://www.A.com/calendar/` to a servlet chain that invokes a servlet named `login` and then a servlet named `calendar`:

```
# htmap -i sws_server -h www.A.com -f /calendar/ \
  -t login,calendar -c SERVLET
```

Note that `login` and `calendar` must be in the servlets path and have definitions in `servlets.properties`.

CODE EXAMPLE 3

To map a URL `http://www.A.com/doors/door-server/` to a resource door:

```
%htmap -i sws_server -h www.A.Com -f /doors/door-server/ \
-t /websites/www.A.com/doors/door-server -c DOOR -z admin
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by the command-line utilities:

site_path`/conf/map.conf` Creates an alias to a path on the file system or a redirection to a remote URL from a URI on the host.

SEE ALSO

`htIntro`(1m), **`map.conf`**(4)

NOTES

If the command is run by the `root` user, then the user name and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

/usr/bin/htpasswd – Administers passwords for the users in HTPASSWD realms.

SYNOPSIS

htpasswd help

htpasswd version

htpasswd [–i *instance_name*] [–h *hostname*] [–p] –r *realm_name* –u *user_name* [–v] [–z *admin_name*]

DESCRIPTION

htpasswd is a utility used only to change passwords for users in HTPASSWD realms. It is provided as a tool that can be incorporated in CGI or other scripts to automate password maintenance.

Users must be created using **htrealm**(1m). Once users are created, any system user can run htpasswd to update passwords (as long as the realm administrator name and password are specified).

OPTIONS

The following options are supported:

–i ***instance***	Specifies the name of the server instance. Valid with all subcommands.
–h ***hostname***	Specifies the name of the virtual host containing the realm. Valid with all subcommands.
–p	Turns off password prompting (for scripts). Valid with all subcommands.
–r ***realm_name***	Specifies the realm name. Valid with all subcommands.
–u ***username***	Specifies a user name whose password is to be set so that the user can have permission to modify realm data. Separate multiple user names with a white space. Valid with all subcommands.
–v	Displays verbose status messages.
–z ***admin_name***	Specifies the name of the administrator of the server, web site, or realm. If users omit this option, users will be prompted for the current password and then the new password. This allows users to change the password. Valid with all subcommands.

EXAMPLES

EXAMPLE 1

Any user can change passwords in an `HTPASSWD` realm if they have the user name and password of the realm administrator. If `realmadmin` is the realm administrator name, a user (or CGI script) can change the password for `user1`:

```
% htpasswd -i sws_server -h www.A.com -r Project \\
  -u user1 -z realmadmin
 Enter Password for realmadmin:
 Setting password for the user user1.
 Password:
 Confirm password:
```

EXAMPLE 2

User, for example, `user1` in the `WebUsers` realm, change their own passwords:

```
 % htpasswd -h www.A.com -i sws_server -r WebUsers -u user1
                          Password for user user1: ***
                          Changing password for the user user1
                          New Password: *****
                          Confirm Password: *****
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by the command-line utilities:

site_path`/conf/realms.conf` Defines realms of user and group information used by access control lists on a Sun WebServer web site.

SEE ALSO **realms.conf**(4), **htrealm**(1m)

NOTES

In order to access the global HTPASSWD realms (`/etc/http/realms/conf`), omit the –`i` and –`h` flags.

This command is installed with `setuid` to `adm` to permit end users invoking the command to have write access to the Sun WebServer configuration after performing necessary checks.

Superusers do not need to specify the –`z` flag, and are allowed to access any command-line utility without authentication.

If the command is run by the `root` user, then the user name and password of an administrator are not required.

Users other than `root` must use the –`z` option and pass the user name and password of a valid administrator to the command.

The server will check whether a user has been designated the realm administrator, site administrator, or server administrator (in this order), where the latter two are defined as those principals who have access to the pseudo-URI ("`/sws-administration`") at the site-level and global access control configuration (`/etc/http/access.conf`).

A server administrator has access to site administration and is able to manage site realms, ACLs, and content. However, site administrators can override this setting by delegating administrators in the administrator blocks in realms, ACLs, and content configuration.

Since the `HTPASSWD` users' file contains encoded passwords, it should be maintained securely.

NAME

/usr/bin/htrealm – Manages realms, users, and groups used to configure access control lists (ACLs) on a Sun™ WebServer™ web site.

SYNOPSIS

htrealm add [–h *hostname*–i *instance*] [–p] –r *realmname*{[–u *userid*[–A]]|[–s [HTPASSWD]|[ISP]|[ISPADMIN]|[UNIXSYS][–d *data_dir*]]|[–g *groupname*[–m *individuals*]]} [–v] –z *admin_name*

htrealm delete [–h *hostname*–i *instance*] [–p] –r *realmname*[[–u *userid*[–A]]|[–g *groupname*[[–A]|[–m *individuals*]]]] [–v] –z *admin_name*

htrealm help[[add]|[delete]|[list]]

htrealm list [–h *hostname*–i *instance*] [–r *realmname*{[–A]|[–u [*userid*]]|[–g [*groupname*]]}][–p] [–v] –z *admin_name*

htrealm version

DESCRIPTION

Realms in Sun WebServer define sets of protection spaces or authentication domains consisting of user names, groups, and passwords. Sun WebServer uses realm information to determine how a user is authenticated. For example, a UNIX-based realm stores user and password information as well as group information in appropriate files or tables if distributed NIS/NIS+ is used. For HTPASSWD realms, you can define your own set of users and groups in a realm. Regardless of how the realm information is stored and accessed, the access control settings require realms to protect resources.

Realms are also differentiated based on how they are used. Two different realms can have different names with the same underlying users and groups database. This gives additional flexibility in naming the authentication domains displayed in the browser.

Most browsers display the realm name in the prompt when a user name and password are required, so the realm name should indicate to users the purpose for password protection and which user name and password to use.

The `htrealm` command can be used to create, delete, and list realm definitions for use with ACLs. It can also be used to manage users and groups in `HTPASSWD` realms. `HTPASSWD` realms are Sun WebServer specific in that their data is stored in `user` and `group` files with Sun WebServer configuration.

Realms in the global `/etc/http/` directory are independent of any web site. These realm definitions are used only for server administration; the user names and passwords are used to log into the Sun WebServer GUI or to execute commands such as `htserver`. Only one such realm may be in use at any given time. The server administration realm must be defined in

`/etc/http/realms.conf` and used to protect the `/sws-administration` URI in `/etc/http/access.conf`.

OPTIONS

Subcommands

The following subcommands are supported:

add Adds a given realm, user, group, or member.

delete Deletes a given realm, user, group, or member.

help Displays help on usage.

list Lists all realms, users, groups, or members.

version Displays the version of `htrealm`.

Options

The following options are supported:

`–A` Indicates that the user or group specified with the `–u` or `–g` flags has administrative privileges of the realm. The administrators must already be valid principals within the realm. Valid with all subcommands (but used most frequently to add, delete, or list realm administrators).

`–d` ***data_dir*** Specifies a directory relative to the site path where the `users` and `groups` files for an `HTPASSWD` realm are stored. *data_dir* is required and valid only if `–s` is `HTPASSWD`; or if you are running on the Solaris ISP Server software, `–d` can also be used when `–s` is `ISPADMIN` to specify the ISP Component ID and version. The default is *site_path*/conf/realms/*realmname* when used with an `HTPASSWD` realm, and "`SUNWhttp-2.1`" when used with an `ISPADMIN` realm. Valid with the add subcommand.

`–g` ***groupname*** Specifies a set of users with permission to access the resources in the realm. Separate multiple group names with white space. Valid with all subcommands.

`–h` ***hostname*** Specifies the name of the virtual host containing the realm. Valid with all subcommands.

`–i` ***instance*** Specifies the name of the server instance. Valid with all subcommands.

–m *individuals* Specifies the individual members of the group. This is a comma-separated list. Valid with the add and delete subcommands.

–p Turns off the prompting for the password such that passwords are taken in from stdin, and scripts may pipe (|) passwords. Valid with all subcommands.

–r *realmname* Specifies the realm name. White spaces must be inside double quotes. Valid with all subcommands.

–s *source_name* Specifies the source of the realm (HTPASSWD, ISP, ISPADMIN, or UNIXSYS). Valid with the add subcommand.

- `HTPASSWD` Indicates that the user or group information is retrieved using the Sun WebServer users/group file format, and that user and group information will be maintained in the data directory named by `realm_dir`. The **`htrealm`**(1m) utility is used to create, delete, and list users and groups and modify passwords using `htpasswd`.
- `ISP` Indicates that the realm information is stored in the Solaris ISP Server shared directory service. Changes to user and group information cannot be made through Sun WebServer.
- `ISPADMIN` Indicates that the principals are Administrators in the Solaris ISP Server Sun™ Internet Administrator™. The `–d` flag takes the ISP-component ID and version (for example, “SUNWftp–2.0”).
- `UNIXSYS` Indicates that the operating system user and group definitions will be used to authenticate users in the realm. Changes to user and group information cannot be made through Sun WebServer.

–u *userid* Specifies the realm user with permission to modify realm data. Separate multiple user names with white space. Valid with all subcommands.

`–v`	Displays verbose status messages. Valid with all subcommands.
`–z` ***admin_name***	Specifies the name of the realm administrator. Valid with all subcommands.

EXAMPLES

EXAMPLE 1

To create a site-specific realm called `Subscribers` on the web site `www.A.com`, you create at least one user and one realm administrator:

```
# htrealm add -i sws_server -h www.A.com -r Subscribers \
  -s HTPASSWD
# htrealm add -i sws_server -h www.A.com -r Subscribers \
  -u user1
Setting password for the user user1.
Password:
Confirm Password:

# htrealm add -i sws_server -h www.A.com -r Subscribers \
  -u user1 -A
```

EXAMPLE 2

A nonroot user can add a realm if a valid user name and password from the `serverAdmin` realm are supplied:

```
% htrealm add -i sws_server -h www.A.com -r System \
  -s UNIXSYS -z admin
Enter Password for admin:
% htrealm list -i sws_server -h www.A.com -z admin
Enter Password for admin:
siteAdmin       HTPASSWD        -
System          UNIXSYS         -
```

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`>0`	An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by the command-line utilities:

site_path/conf/access.conf

Configures a web site's ACLs.

/etc/http/access.conf

Configures the server administration ACLs.

site_path/conf/realms.conf

Defines realms of user and group information used by access control lists on a Sun WebServer web site.

site_path/conf/realms/*HTPASSWD_realm_name*/users

Lists the users in the HTPASSWD realm.

Entries in this file have the form *username*:*password*.

site_path/conf/realms/*HTPASSWD_realm_name*/groups

Lists the groups in the HTPASSWD realm.

Entries in this file have the following form:

```
group <group_name> {
 member1
 member2
 member3
}
```

SEE ALSO

realms.conf(4), **htaccess**(1m), **access.conf**(4), **htpasswd**(1m)

NOTES

If the command is run by `root` user, then the user name and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

/usr/bin/htserver – Create, delete, and manage Sun™ WebServer™ `httpd` instances.

SYNOPSIS

htserver add *instance* [*conf_file*] [–v]

htserver delete *instance* [*instance*...] [–v]

htserver disable [[–a][*instance*]...] [–v]

htserver enable [[–a][*instance*]...] [–v]

htserver help *command*

htserver list [[–a][*instance*]...] [–v]

htserver query [[–a][*instance*]...]

htserver restart [[–a][*instance*]...]

htserver start [[–a][*instance*]...] [–v]

htserver stop [[–a][*instance*]...] [–v]

htserver update *instance* [*conf_file*] [–v]

DESCRIPTION

The `htserver` command creates and maintains the Sun WebServer server instances. Each server instance is a process associated with a configuration file, and each one hosts one or more web sites.

`htserver` can start, stop, and restart server instances. It can also enable or disable server instances. Each “enabled” server instance will be started when the machine reboots or when `htserver start` or `htserver restart` is run with no instance specified.

The `list` and `query` subcommands can be used to get information about what servers are running or enabled and what configuration files each instance uses.

Once a server instance is created, use **`hthost`**(1m) to add web sites. To modify the configuration, either use the Sun WebServer GUI (http://hostname:2380/admin/admin.html), or edit the server configuration file (see **`httpd.conf`**(4)).

The server instance named `admin` is the administration server. This server instance is created when Sun WebServer is installed and listens to port 2380 on all IP addresses. The `admin` server instance is used to access the Sun WebServer GUI.

OPTIONS

Subcommands

The first argument to `htserver` must be one of the following subcommands:

add Creates a reference to a new server instance so that it can be managed. An entry is added to the server list in **`httpd-instances.conf`**(4).

delete Deletes a server instance from `httpd-instances.conf`. This removes the server instance from Sun WebServer management, but does not delete the data or configuration files for the server or any of its sites.

disable Disables a server instance. Disabled instances can only be started by running `htserver start` *instance_name*, and explicitly specifying the instance name. the `disable` subcommand does not stop a running server instance.

enable Enables a server instance. Enabled instances will be started when the machine reboots or whenever `htserver start` or `htserver restart` is run with no instance named.

Solaris ISP Server has a background service that periodically checks the state of all enabled servers. If you have installed Sun WebServer as a part of the Solaris ISP Server, this service attempts to restart all enabled servers that are not running. If the restart fails due to an error in the server configuration, it disables the server and sends a message to the server administrator stating that the server has been disabled and will not be restarted automatically.

help Displays usage information for this command.

list Lists server instances and status information for all servers or each named instance.

query Displays detailed status and statistical information about a named server instance. Also displays host and port statistics when used with the `–v` option. Status can be one of the following:

Down All processes have stopped running. Occurs when the server has been stopped or during a small time frame before the server enters Initializing status.

Initializing Server is parsing configuration files and initializing internal data structures.

Running All listening ports are waiting for client connections. Occurs when the server has started or restarted successfully.

Restarting Server is destroying data structures, closing connections and listeners, and killing all running server processes. After this cleanup, the server status changes to Initializing.

Stopping Server is destroying data structures, closing connections and listeners, and killing all running server processes. Same as the Restarting state except that after the cleanup process is complete, the process dies rather than returning to Initializing status.

restart Restarts named server instances or all of the currently running server instances.

start Starts named server instances or all enabled server instances with the configuration files listed in `httpd-instances.conf`.

stop Stops named server instances or all running server instances.

update Updates a named instance with the named configuration file.

Subcommand Options

The following options are supported:

`–a` Indicates that the command applies to Sun WebServer. Sun WebServer is a special server instance that allows remote administration of servers and sites through the Sun WebServer GUI.

`–v` Runs the command in verbose mode with more descriptive messages output to the screen.

OPERANDS

The following operands are supported:

conf_file Specifies the name of a configuration file to use for a server instance. This operand is required as the last argument for the `add` and `update` subcommands.

instance Specifies a server by its instance name. Instance names are maintained in the `/etc/http/httpd-instances.conf`

file. A single instance name is required with the `add`, `update`, `delete`, and `query` subcommands. Other subcommands take an optional instance name or list of instance names.

EXAMPLES

EXAMPLE 1

To list all server instances (nonroot users must use `---z` and provide a user name and password from the `serverAdmin` realm):

```
% htserver list -z admin
Enter Password for admin:

Instance : sws_server
        Enabled : No
        pid : -
        Config file : / /etc/http/sws_server.httpd.conf

Instance : aws
        Enabled : Yes
        pid : 4018
        Config file :  /var/opt/SUNWixamc/awsconf/aws.conf

Instance : SUNWixmon
        Enabled : Yes
        pid : 4020
        Config file : /opt/SUNWixmon/sws/SUNWixmon.httpd.conf

Instance : admin
        Enabled : No
        pid : -
        Config file : / /usr/http/admin_server/conf/admin.httpd.conf
```

To create a new server named `Large_Sites` with default values for the configuration file (`/etc/http/Large_Sites.httpd.conf`), server root directory (`/var/http/Large_Sites`), and site directory (`/var/http/Large_Sites/websites/`):

```
# htserver add Large_Sites
Creating server configuration file:
 /etc/http/Large_Sites.httpd.conf
Creating server root directory:
/var/http/Large_Sites
Creating site directory:
/var/http/Large_Sites/websites/default_site
Creating site configuration:
```

(continued)

(Continuation)

/var/http/test_server/Large_Sites/websites/default_site/conf/default_site.site.conf

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

FILES

The following files are used by this utility:

`/etc/http/`*instance_name*`.httpd.conf` The server instance configuration file.

`/etc/http/httpd-instances.conf` Tracks all Sun WebServer instances. When `htserver` creates a new instance, an entry is added to this file.

SEE ALSO

`httpd.conf`(4), **`httpd-instances.conf`**(4), **`hthost`**(1m)

NOTES

If the command is run by the `root` user, then the user name and password of an administrator are not required.

Users other than `root` must use the `–z` option and pass the user name and password of a valid administrator to the command.

NAME

/usr/bin/htservlet – Configures the behavior of a servlet engine.

SYNOPSIS

htservlet add [–a *init_args*] [–b *codebase*] –c *classname* –h *hostname* –i *instance* –n *servlet_name* [–p] [–s] –u *username* [–v]

htservlet cookie [–C *cookie_comment*] [–D *cookie_domain*] [–h *hostname*] –i *instance* [–N *cookie_name*] [–p] [–Q *cookie_path*] [–S] –u *username* [–v] [–X *cookie_max_age*]

htservlet delete –h *hostname* –i *instance* –n *servlet_name* [–p] –u *username* [–v]

htservlet disable –g *option* [–h *hostname*] –i *instance* [–p] –u *username* [–v]

htservlet enable –g *option* [–h *hostname*] –i *instance* [–p] –u *username* [–v]

htservlet help

htservlet list –h *hostname* –i *instance* [–v]

htservlet load –h *hostname* –i *instance* –n *servlet_name* [–p] –u *username* [–v]

htservlet log [–f *prefix*] [–h *hostname*] –i *instance* [–m *max_num_files*] [–p] [–t *cycle_time*] –u *username* [–v] [–z *file_size*]

htservlet reload [–a *init_args*] –h *hostname* –i *instance* –n *servlet_name* [–p] –u *username* [–v]

htservlet query –h *hostname* –i *instance* [–v]

htservlet security [–h *hostname*] –i *instance* [–o *permission*] [–p] [–r *resource*] –u *username* [–v]

htservlet session [–h *hostname*] [–I *invalidation_time*] –i *instance* [–p] [–R *max_residents*] –u *username* [–v] [–W *swapdir*]

htservlet settings [–d *servlet_path*] [–h *hostname*] –i *instance* [–j *server_classpath*] [–P *properties_file*] [–p] –u *username* [–v]

htservlet single [–h *hostname*] –i *instance* [–L *init_pool_size*] [–M *max_pool_size*] [–p] –u *username* [–v]

htservlet unload –h *hostname* –i *instance* –n *servlet_name* [–p] –u *username* [–v]

htservlet version

DESCRIPTION

A servlet engine can be defined on any Sun WebServer server instance or any individual web site. The servlet engine runs a Java™ virtual machine that loads and executes servlets defined in its `servlets.properties` file.

The `htservlet` command defines and modifies servlet engine runtime, security, and logging properties; adds or removes entries in the `servlets.properties` file; and loads or unloads servlets in running servlet engines.

For subcommands where –h *hostname* is optional, omitting –h applies the subcommand to the shared servlet engine (server-wide setting).

OPTIONS

Subcommands

The following subcommands are supported:

Note - You must restart the server in order for the changes made to the configuration files to take effect.

---

add Adds a servlet to the `servlets.properties` file. Adding a servlet does not imply that the servlet is automatically loaded.

cookie Configures a default cookie for the servlet engine and writes changes to the `httpd.site.conf` file.

delete Deletes a servlet from the servlet engine and writes changes to the `servlets.properties` file.

disable Disables an option on the servlet engine and writes changes to the configuration file.

enable Enables an option on the servlet engine and writes changes to the configuration file.

help Displays help on usage.

list Lists all loaded servlets on the server.

load Loads a servlet from the `servlets.properties` file on a running server.

log Configures the log location and cycling parameters and writes changes to the configuration file.

query Returns current servlet engine settings on a running server.

reload Reloads a servlet on a running server.

security	Configures the security settings for the servlet engine and writes changes to the configuration file.
session	Configures the session management settings for the servlet engine and writes changes to the `httpd.site.conf` file.
settings	Specifies the settings on the servlet engine and writes changes to the configuration file.
single	Configures the pool size for servlets that implement the SingleThreadModel interface.
unload	Unloads a servlet from the running web server.
version	Displays the version of htservlet.

Options

The following options are supported:

–a ***init_args***	Specifies the optional initial arguments passed to the servlet. Used in the format *name=value* [, *name=value...*]. Valid with the add and reload subcommands.
–b ***codebase***	Specifies the URL of the servlet's code base. This URL can be pointing to a directory or a JAR file. Used only for remote servlets. Valid with the add subcommand.
–C ***cookie_comment***	Specifies a comment of the cookie carrying the session ID. Valid with the cookie subcommand.
–c ***classname***	Specifies the name of the servlet main class file. Valid with the add subcommand.
–D ***cookie_domain***	Specifies the domain where cookies with session IDs are valid. For example, if a cookie has a domain of "`www.A.com`", then only "`www.A.com`" will recognize it as a valid cookie. All other servers will reject this cookie. Valid with the cookie subcommand.
–d ***servlet_path***	Specifies the directories and JAR files for the servlet engine on the local machine. This is a colon-separated list. This option can only prepend the path specified to the original path in the file. Valid with the settings subcommand.

–f ***prefix*** The path name and the file prefix for servlet log files. As new log files are created, they use this prefix and a number suffix. Valid with the log subcommand.

–g ***option*** Specifies a servlet engine option to enable or disable. Valid options are:

`chain`	Enables or disables servlet chaining. Servlet chains are a sequence of servlets executed in the specified order to fulfill one single servlet request.
`cookie`	Enables or disables cookie support on this server. This is a server-wide setting.
`jvm`	Enables or disables the server to enable a JVM. This is a server-wide setting.
`log`	Enables or disables servlet error logging.
`persistence`	Enables or disables session persistence in the servlet engine. If session persistence is enabled, sessions are written out to disk on server shutdown, and recovered on startup. This is a server-wide setting.
`protocol`	Enables or disables rewriting of URL with session ID when a protocol switch is involved, for example, switching from "`http`" to "`https`" or vice-versa.
`remote`	Enables or disables loading servlets from remote sites in this servlet engine.
`se`	Enables or disables the servlet engine for the server process or

		web site. This is a server-wide setting.
	`session`	Enables or disables session support in all the servlet engines. If the session is supported, session swapping will be enabled by default.
	`share`	Allows all sites on the server to share a single servlet engine. This is a server-wide setting. If this option is disabled, each site is allowed to create its own servlet engine instance.
	`url`	Enables or disables URL rewriting for this servlet engine. If enabled, session IDs are appended to URLs by either the `encodedUrl()` or `encodeRedirectUrl()` method. This is a server-wide setting.
–h ***hostname***		Specifies the name of the virtual host. Valid with all subcommands.
–I ***invalidation_time***		Specifies the length of time (in minutes) a session is allowed to remain unused before becoming invalidated on this servlet engine. Valid with the session subcommand.
–i ***instance***		Specifies the name of the server instance. Valid with all subcommands.
–j ***server_classpath***		Specifies the class path for the Java virtual machine (JVM) which may include the location of `classes.zip` file of JDK, JSDK, and the servlet engine. This is a colon separated list (for example, *usr*/*lib*/*java*/ [:/*usr*/*java*/*lib*...]). This command only prepends to the existing path. Do not put the servlets directories in the server classpath. It is a server-level setting. Valid with the settings subcommand.

Note - Because running the command-line utility `htserver restart` or restarting the server from the Sun WebServer GUI does not restart the Java virtual machine, if you change `server_classpath`, you must kill and restart the Sun WebServer process in order for your changes to take effect.

–L *init_pool_size* Specifies the initial servlet pool size for any servlets implementing the SingleThreadModel interface. Valid with the single subcommand.

–M *max_pool_size* Specifies the maximum servlet pool size for any servlets implementing the SingleThreadModel interface. Valid with the single subcommand.

–m *max_num_files* Specifies the maximum number of log files. When the log suffix exceeds this number, the next log file is created which overwrites the first log file. The default number is 7 files. Valid with the log subcommand.

–N *cookie_name* Specifies the name of the cookie used to carry session IDs when cookies are enabled. Default is "`swssessionid`". Valid with the cookie subcommand.

–n *servlet_name* Specifies the name of the servlet. Valid with the add, delete, load, reload, and unload subcommands.

–o *permission* Specifies the access settings used in conjunction with the –r option. Valid with the security subcommand.

`all`	Allows local and remote servlets to access resources.
`local`	Allows access only to resources on the same host.
`none`	Allows no access.
`remote`	Allows access only to resources on servlets with a code base.

–P *properties_file*
Specifies the full path to the properties file for the servlet engine. Valid with the settings subcommand.

–p
Disables password prompting, and the password is piped (“|”) to the command. Valid with the add, delete, disable, enable, load, log, reload, unload, security, and settings subcommands.

–Q *cookie_path*
Specifies the path on the local host for which cookies with session IDs are valid. Pages outside of this path cannot read the cookie. This path is relative to the document root. Default is "/". Valid with the cookie subcommand.

–R *max_residents*
Specifies the maximum number of resident sessions in a servlet engine. If the maximum number has been reached, sessions are swapped out onto disk. Session swapping is enabled if sessions are enabled. Valid with the session subcommand.

–r *resource*
Specifies the resource settings used in conjunction with the –o option for access control. Valid with the security subcommand.

`file`	Sets access permissions for file resources such as read / write a file on local disk.
`link`	Sets access permissions for links to dynamic libraries.
`network`	Sets access permissions for network resources.
`security`	Sets access permissions for security resources such as `classLoaders`.
`system`	Sets access permissions for system resources such as **System.Exec ()**.

–S
Indicates that the session cookie will include the "secure" field. Valid with the cookie subcommand.

–s	Indicates that the servlet will be loaded at start-up. Valid with the add subcommand.
–t ***cycle_time***	The log cycle time measured in minutes. When the log cycle time exceeds this number, a new log file is created with an incremented suffix. The default time is 1440 minutes (1 day). Valid with the log subcommand.
–u ***username***	Specifies user name. Valid with the add, delete, disable, enable, load, log, reload, unload, security, and settings subcommands.
–v	Specifies verbose mode for more detailed messages. Valid with all subcommands.
–W ***swapdir***	Specifies the directory where sessions will be swapped during session persistence or when the number of resident sessions has exceeded the maximum. Valid with the session subcommand.
–X ***cookie_max_age***	Specifies the maximum age of a cookie before expiring. Valid with the cookie subcommand.
–z ***file_size***	Specifies the maximum log file size measured in bytes. When the log file size exceeds this number of bytes, a new log file is created with an incremented suffix. The default file size is 1048576 bytes (1 MB). Valid with the log subcommand.

EXAMPLES

EXAMPLE 1

To enable servlets on a server:

```
# htservlet enable -g jvm -i sws_server -u admin
```

EXAMPLE 2

To load a declared servlet on a server:

```
# htservlet load -i sws_server -h www.A.com -n foo -u http
```

EXAMPLE 3

To add servlets to be loaded at start-up:

```
# htservlet add -i sws_server -h www.A.com -n foo \
-c FooServlet -b http://x.eng/ -a counter=1 -s \
-u http
```

EXAMPLE 4

To add a servlet declaration (in verbose mode):

```
# htservlet add -i sws_server -h www.A.com -n foo \
-c FooServlet -b http://x.eng/ -a counter=1 -v \
-u http
```

EXAMPLE 5

To set the server classpath:

```
# htservlet settings -i sws_server -j /usr/jdk/lib/classes/zip:. \
-u http
```

EXAMPLE 6

To enable cookie support on a server:

```
# htservlet enable -g cookie -i sws_server -u admin
```

EXAMPLE 7

To set the cookie name for the default session identifier:

```
# htservlet cookie -i sws_server -h www.A.com \
-N MySessionId -u admin
```

EXAMPLE 8

To set the session swap directory:

```
# htservlet session -i sws_server -h www.A.com \
-W /tmp/sessionSwapDirectory -u admin
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhtsvl
Interface Stability	Evolving

FILES

The following files are used by this utility:

site_path`/conf/`*site_name*`.httpd.conf`

Contains the web site servlet engine configuration if the servlet engine is not shared.

`/etc/http/httpd.conf`

Contains the server instance servlet engine configuration if all web sites share the servlet engine.

site_path`/conf/servlets.properties`

Defines each servlet that can be loaded by a web site.

SEE ALSO

`httpd.conf`(4), **`httpd.site.conf`**(4), **`servlets.properties`**(4)

NOTES

If the command is run by the `root` user, then the user name and password of an administrator are not required.

Users other than `root` must use the `–u` option and pass the user name and password of a valid administrator to the command.

NAME

/usr/lib/httpd – Starts and stops servers.

SYNOPSIS

httpd help

httpd start

httpd stop

DESCRIPTION

Server instances can be started or stopped by using the `htserver` utility, through the Sun WebServer GUI, or by executing this script. It is recommended that you use `htserver` or the Sun WebServer GUI.

OPTIONS

The following subcommands are supported:

help Displays help on usage.

start Starts all “enabled” servers in `httpd-instances.conf`.

stop Stops all servers.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

SEE ALSO

`htserver`(1m)

NOTES

CHAPTER 2

man Pages(4): File Formats

NAME

htIntro – Introduction to the man pages of the Sun™ WebServer™ configuration and log files. The man pages offer detailed instruction and examples on syntax and directives for each file.

DESCRIPTION

The configuration and log files are available to configure and monitor Sun WebServer.

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhtman
Interface Stability	Evolving

FILES

The following files are used by the command-line utilities:

access.conf

Configures a web site's access control lists (ACLs) . Located at *site_path*/conf/access.conf.

access.conf

Configures the server administration ACLs. Located at /etc/http/access.conf.

content.conf

Defines the content variants, encoding types, and directory preferences for a web site. Located at *site_path*/conf/content.conf.

httpd-instances.httpd.conf

Tracks all Sun WebServer instances. When htserver creates a new server instance, an entry is added to this file. Located at /etc/http/httpd-instances.httpd.conf.

***site_name*.site.conf**

Contains the web site servlet engine configuration if the servlet engine is not shared. Located at *site_path*/conf/*site_name*.site.conf.

***instance_name*.httpd.conf**

Defines the server instance's configuration. When `hthost` adds a new site, it creates an entry in `httpd.conf` to define the *site_path* and web site configuration file. Located at `/etc/http/`*instance_name*`.httpd.conf`.

`map.conf`

Creates an alias to a path on the file system or a redirection to a remote URL from a Uniform Resource Identifier (URI) on the host. Located at *site_path*`/conf/map.conf`.

`realms.conf`

Defines realms of user and group information used by access control lists on a Sun WebServer web site. Located at *site_path*`/conf/realms.conf`.

`servlets.properties`

Defines each servlet that can be loaded by a web site. Located at *site_path*`/conf/servlets.properties`.

SEE ALSO

`htaccess`(1m), **`htcontent`**(1m), **`hthost`**(1m), **`htmap`**(1m), **`htpasswd`**(1m), **`htrealm`**(1m), **`htserver`**(1m), **`htservlet`**(1m), **`httpd`**(1m)

NOTES

`access.conf`

Defines ACLs for the content hosted by a web site. ACLs regulate access to resources on the site by defining which users, groups, and/or hosts have permissions to make HTTP GET, PUT, POST, and DELETE requests.

`content.conf`

Represents meta-data for directories, site preferences for languages, media and encoding types, and specifying variants for content.

`httpd-instances.httpd.conf`

Associates each unique `httpd` process (or server instance) with its configuration file, and defines whether or not a server instance is enabled.

`httpd.cgi.logs`

Logs errors generated by CGI scripts. Log files will be named *prefix.sequence*, where *sequence* is a cycling number.

`httpd-instances.httpd.conf`

The server instance configuration file contains directives that define the server's runtime behavior, the web sites it hosts, and the network connection end points it uses.

`httpd.event.logs`

The server events log files. The `httpd` server puts out error messages and warnings via `syslogd` to `/var/adm/messages` by default.

`httpd.request.logs`

Logs all incoming requests to a server activity log file. Sun WebServer generates log files using one of three log file formats configurable by the administrator.

`httpd.servlet.logs`

Logs errors generated by servlets. Log files will be named *prefix*.*sequence*, where *sequence* is a cycling number.

***site_name*`.site.conf`**

The sites instance configuration file contains directives that define the site's runtime behavior, and defines the identity and server resources used by a web site hosted by the server instance. This file defines properties such as the web site's canonical host name and aliases, the location of the configuration files, and the network connections available to the web site, and defines the servlet engine settings for the web site.

***instance_name*`.httpd.conf`**

The server instance's configuration file contains directives that define the server's runtime behavior, and defines the web sites hosted on the server. When `hthost` adds a new site, it creates an entry in `httpd.conf` to define the *site_path* and web site configuration file.

`map.conf`

A map directive allows you to redirect requests for a URL on a host to any other URL or to a different directory.

`realms.conf`

Defines realms of user and group information used by access control lists on a Sun WebServer web site. A realm defines a protection space, a domain of users and groups and their permissions.

`servlets.properties`

Defines the servlet properties file in a general Java properties file format. The file contains the name of each servlet and the initialization parameters of the servlet.

NAME

access.conf – Defines access control lists that regulate access to web site resources on a Sun ™ WebServer™ web site.

SYNOPSIS

site_path`/conf/access.conf` - Web site access control lists.

`/etc/http/access.conf` - Global access control lists for administration

DESCRIPTION

The `access.conf` file defines access control lists (ACLs) for the content (represented as URI) hosted by a web site. ACLs regulate access to resources on the site by defining which users, groups, and/or hosts have permission to make HTTP `GET`, `PUT`, `POST`, and `DELETE` requests. ACLs also determine which users, groups, and/or hosts have permission to modify the ACL itself.

ACL definitions may be created by **`htaccess`**(1m), by the Sun WebServer GUI, or by Apache Emulator for FrontPage publishing tools.

Note - If FrontPage publishing is active for a site, do not edit the `access.conf` file. FrontPage must be able to read and write the ACL information in a form that it can use, and manual edits may interfere with FrontPage's ability to manage ACL information.

Each ACL definition in the file consists of the following:

- Realm used to store information about valid users.
- Authentication scheme used to ask clients for password information.
- List of users and/or groups in the realm who have permission to change the ACL definition.
- HTTP method block or blocks defining the access permissions on the URI for the given methods.
- Optionally, URI of the resource using the ACL. If the URI is not specified, it applies to all the URIs under this site.
- Optionally, ACL definition blocks for URIs beneath the current one in the document root. These definitions assume the definition of the parent block; explicit directives change the definition in the child block and its children.

Syntax

The following syntax rules apply to the `access.conf` file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored.
- White space is ignored in directive definitions.

- A string with a space must be inside double quotes.
- Some directives accept a list of values. Separate multiple values by white space. If more than one line is required to list all values, escape all but the last new line with a backslash (\) at the end of the line.
- All directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, new lines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks that also use matched curly braces to contain a definition.
- ACL definitions (`url { }` blocks) may be nested within other definitions, as long as the nesting matches the actual URI hierarchy. You cannot nest `url /parent/subdir { ... url /parent { ... } }`. The nested URL is always treated as a path relative to the parent URL.
- The ACL definition on a given URI inherits directives from ACLs on parent URIs, even if the blocks are not nested.

Each ACL definition is in the following form:

```
url <URI> {
  [ realm <realm_name> ]
  [ authentication_scheme <auth_scheme> ]
  [ administrator {
     user <realm_user>[ <realm_user>]...
     [ group <realm_group>[ <realm_group>]... ]
  } ]
  [ method <method_list> {
     [ + | - group * | <realm_group>[ <realm_group>]... ]...
     [ + | - host * | <host>[ <host>]... ]...
     [ + | - user * | <realm_user>[ <realm_user>]... ]...
  } ]...
  [ url <URI> { <ACL> } ]...
}
```

The syntax and definition of each directive and block is explained in the following *Directives* section. Note that all directives are optional. Unless a directive is explicitly defined, its value is inherited from the ACL on the parent URI. If there is no value defined in the parent URI ACL (or any of its parents), the following defaults apply:

`realm`	There is no default. If no `realm` can be defined, then all `user` and `group` directives are ignored.
`authentication_scheme`	If you do not change the default of no `authentication_scheme`, then all `user` and `group` directives are ignored.
`method`	Defaults to `ALL`.

`group`	Defaults to `+ group *`, meaning any authenticated member of any group in the realm is allowed access unless another directive explicitly denies access.
`host`	Defaults to `+ host *`, meaning connections are allowed from any host name.
`user`	Defaults to `+ user *`, meaning any authenticated used defined in the realm is allowed access unless another directive explicitly denies access.
`administrator`	Defaults to either the site or the server administrator. A site administrator is the principal with access to the pseudo-URI "`/sws_administration`" in the site-level `access.conf`. Similarly, a server administrator is a principal with access to the pseudo-URI "`/sws_administration`" in the global `access.conf`.

Directives

The following keyword directives are valid in the `access.conf` file:

`administrator { `*admins*` }`

Names users and groups in the specified realm with permissions to change or delete the ACL through the Sun WebServer GUI or the **`htaccess`**(1m) utility. The directive *admins* may contain a user directive or optionally a group directive:

`user` *realm_user*`[ `*realm_user*`]...`

Defines ACL administrators. User names must be valid in the ACL's `realm`.

`group` *realm_group*`[ `*realm_group*`]...`

Defines groups whose members have permission to change or delete the ACL. Group names must be valid in the ACL's `realm`.

`authentication_scheme basic | md5 | none`

Defines the encoding of authentication information for the ACL where `basic` means user name and password information should be sent in BASE64 encoding over HTTP; `md5` means the server exchanges a message digest of certain header fields and the password is never sent over the wire; and `none` means that no authentication scheme will be used.

`method` *method*`[` *method*`]... { ` *permissions* `}`

Defines a block of *permissions* that regulate access to the named resource for the HTTP methods listed. The *method* can be `ALL` meaning any method, or it can be a list of one or more of: `DELETE`, `GET`, `POST`, or `PUT`. The *permissions* regulate access to the resource by realm user name, realm group name, or host name:

`+|- host * |` *pattern*

Allows (`+`) or denies (`-`) access to hosts matching the *pattern*. If *host* is an asterisk (`*`), the access permission serves as the default for all hosts.

If *pattern* consists only of numbers, it is considered an IP address pattern; otherwise, it is considered a host or domain name pattern. For IP addresses, the permission will be applied to any host whose IP address begins with *pattern*. For host or domain names, the permission will be applied to any host whose fully qualified domain name (FQDN) ends with *pattern*.

`+|- group * |` *group*`[` *group*`]...`

Allows (`+`) or denies (`-`) access to authenticated realm users who are members of one of the listed groups. If *group* is an asterisk (`*`), the access permission serves as the default for all groups.

`+|- user * |` *user*`[` *user*`]...`

Allows (`+`) or denies (`-`) access to authenticated realm users. If *user* is an asterisk (`*`), the access permission serves as the default for all users.

`realm` *realm_name*

Defines the realm to use as the source for user name, password, and group information for this ACL. The *realm_name* must exist in the web site's ***`realms.conf`***(4) file, or all user and group permissions will be ignored.

EXAMPLES

EXAMPLE 1

Access control for the Sun WebServer GUI server administration is defined in the global `/etc/http/access.conf` file. The special URI `/sws-administration` defines ACLs for administration:

```
url "/sws-administration" {
  realm              serverAdmin
  authentication_type basic
  + user  *
}
```

EXAMPLE 2

A web site ACL to restrict HTTP publishing to valid users:

```
url "/" {
   realm               publishing
   authentication_type basic
   method PUT DELETE {
      + user *
   }

}
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO

htaccess(1m), **htIntro**(4), **htrealm**(1m), **realms.conf**(4)

NOTES

The global `/etc/http/access.conf` is used by the Sun WebServer GUI. The command-line utilities protect server-wide administrative access using a pseudo-URI ("`/sws_administration`"). Similarly, the site-wide administrative access is protected by the server pseudo-URI at site-specific `access.conf`.

NAME

content.conf – Represents meta-data for directories, site preferences for languages, media and encoding types, and specifying variants for content.

SYNOPSIS

site_path`/conf/content.conf`

DESCRIPTION

The content definitions may be created by **`htcontent`**(1m).

For directories, you can set whether they can be browsed and also sets the format for directory listings. You can also create a list of default file names to search for in the directory.

For files, you can set preferences for HTTP 1.1 content negotiation. Preferences include character set, language, compression encoding, and media type.

You can set variant information for a Uniform Resource Identifier (URI). If a URI has a set of associated file variants, the server will select the most appropriate variant based on the client's preferences and the preference settings on each file variant.

Syntax

The following syntax rules apply to the `content.conf` file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored.
- White space is ignored in directive definitions.
- Some directives accept a list of values. Separate multiple values by white space. If more than one line is required to list all values, escape all but the last newline with a backslash (\) at the end of the line.
- Directives of the type name = value require whitespace around the = symbol.
- All directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, newlines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks. Directive blocks also use matched curly braces to contain a definition. The `url{}` blocks can be embedded inside other `url{}` blocks. Then the embedded `url{}` block inherits information from its parent block unless it explicitly redefines a directive, in which case the directives in the child block will override the directives in the parent.

Content settings take the following form:

```
url <URI> {
variant <variant_info>
 preferences {
```

(continued)

(Continuation)

```
  directory_listing {fancy | simple | off}
  default_files
  media_type
  language
  encoding
  charset
 }
 [url <file_uri> {
  filename <variant_info>
  preferences {
    media_type
    language
    encoding
    charset
  }
 }]...
}
```

The syntax and definition of each directive is explained in the following *Directives* section.

Directives

The following keyword directives are valid in the `content.conf` file:

`preferences` *preferences* Sets the server's content negotiation preferences for the specified URI. Preferences are listed in order of preference. Separate multiple preferences with a space. Valid preferences are:

`charset` *charset* [*charset*`...`]

Specifies the character set of the data. A character set refers to a method used with one or more tables to convert a sequence of octets into a sequence of characters. The default `charset` for variants is ISO-8859–1.

`default_file` *filename* [*filename*`...`]

Specifies in order of preference the default files Sun WebServer looks for in a directory. If no match is found in this list of file names, the directory contents are displayed. Multiple files must be separated by whitespace. A different set of default files can be named in any directory.

Use the null string, `""`, as the file name to indicate that there is no default file for the directory. If no file is named, the directory contents will always be listed.

`directory_listing` *method*

Sets the method for displaying the contents of a directory when there is no file matching one of the default file names. The listing type can be "`fancy`" to display each directory entry as a hyperlink with the file size, the last modified time, and an icon next to each entry to indicate the file type; or "`simple`" to display directory entries as plain text hyperlinks; or "`off`" to display no directory contents.

`encoding` *encoding* [*encoding*`...`]

Specifies the preferred encodings or the encoding type. Encoding refers only to methods of compression. For example, `gzip` or `compress` reveals which methods have been used to encode the file.

`language` *language* [*language*`...`]

Specifies in order of preference the preferred languages. Languages are specified in the standard two-letter format.

`media` *media_type* [*media_type*`...`]

Specifies the preferred types of media. Media type is in standard MIME type format.

`variant_info` *variants*

Sets the server's variants for resources. Separate multiple variants with a space. Valid variants are:

`charset` *charset* [*charset*`...`]

Specifies the character set of the data. A character set refers to a method used with one or more tables to convert a sequence of octets into a sequence of characters.

`encoding` *encoding* [*encoding*`...`]

Specifies the preferred encodings or the encoding type. Encoding refers only to methods of compression. For example, `gzip` or

`compress` reveals which methods have been used to encode the file.

`language` *language* [*language*...]

Specifies the preferred languages. Languages are specified in the standard two-letter format.

`media_type` *media_type* [*media_type*...]

Specifies the preferred types of media. Media type is in standard MIME type format.

EXAMPLES

EXAMPLE 1

To add an encoded French variant and a German variant for the URI `index.html`:

```
url /index.html {
    index.fr.html lang = fr enc = gzip char = iso-8809-1 type = text/html
    index.de.html lang = de type = text/html; level=3.0
}
```

Note that the `level=3.0` is part of the type definition; it indicates the HTML version level of the resource. If a browser uses the level in a request for an HTML document, and all other things are equal, the server will send the matching resource with the greatest HTML level acceptable to the client.

EXAMPLE 2

To set the default files in a directory to `index.html` or `index.shtml`:

```
url /pages/ {
   preferences {
      directory_listing off
      default_files index.html index.shtml
   }

   url index.html {
      index.ja.html lang = ja char = EUC_JP
      index.en.html lang = en char = iso-8809-1
      index.fr.html lang = fr char = iso-8809-1
   }

}
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO **htcontent**(1m), **htIntro**(4)

NAME

groups – Defines the groups in an HTPASSWD realm

SYNOPSIS

site_path/conf/realms/*HTPASSWD_realm*/groups

DESCRIPTION

The groups file defines the groups in an HTPASSWD realm.

You may create new groups in the realm by **htrealm**(1m) or by the GUI.

All members of a group must be valid users defined in **users**(4).

Syntax

The following syntax rules apply to the groups file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored
- White space is ignored in directive definitions.
- This directive accepts a list of values. Separate multiple users by white space. If more than one line is required to list all values, escape all but the last new line with a backslash (\) at the end of the line.
- Directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, newlines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks which also use matched curly braces to contain a definition.

Each group definition lists the member's user name separated by white space in the following form:

```
groupname {
 username1
 username2
}
```

EXAMPLES

EXAMPLE 1

Sample groups file:

```
# the groups file containing groups in an HTPASSWD realm
group administration {
 webmaster # group block listing members of the group
}
group group_two {
 user1
 user2
 user3
```

```
}
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO

htIntro(4), **htrealm**(1m), **realms.conf**(4), **users**(4)

NAME

httpd-instances.conf – List of all Sun™ WebServer™ server instances.

SYNOPSIS

`/etc/http/httpd-instances.conf`

DESCRIPTION

The `httpd-instances.conf` file associates each unique `httpd` process (or server instance) with its configuration file, and defines whether or not a server instance is enabled. A server instance is a named `httpd` process that services one or more web sites. At installation, there are two default instances: `sws_server` and `admin`, the server process which handles the Sun WebServer GUI. The `admin` instance is not created unless `SUNWhtadm` has been installed.

Entries in `httpd-instances.conf` may be modified by using the `htserver` utility, through the Sun WebServer GUI, or editing the file directly. It is recommended that you use `htserver` or the Sun™ WebServer™ GUI.

A new server instance is typically needed only when it is impractical to add web sites to an existing server, either because all existing instances already have large numbers of sites or because a new web site requires a configuration that no existing instance can support. Instances cannot share port numbers.

Syntax

The following syntax rules apply to the `httpd-instances.conf` file:

- The exclamation mark (!) is a comment character. All characters from a ! to the end of a line are ignored
- White space is ignored in directive definitions.
- Separate multiple values by white space or tab character.
- Each entry is written on one line with three fields. Additional fields on the same line are ignored.
- There is no syntax error checking for the first two fields (*instance-ID* and *configuration_file*). The third field can have any of the following values to indicate that the site is enabled: enabled, true, yes, 1. Any other value in this field would disable the site.

Each entry in `httpd-instances.conf` has the following format

<instance-ID> <configuration_file> <state>

The syntax and definition of each directive are explained in the following *Directives* section.

Directives

The following keyword directives are valid in the `httpd-instances.conf` file:

instance-ID

Shows the unique string used to identify the server instance. The *instance-ID* is used in the Sun WebServer GUI, command-line utilities, and (by default) to name the configuration file. The string may contain any alphanumeric characters, but may not contain spaces.

The *instance-ID* "`admin`" is reserved to name Sun WebServer. You should not delete or rename this server instance unless you want to disable web-based administration.

configuration_file

Shows the full path location of the configuration file for this server instance. By default, the configuration file will be `/etc/http/`*instance-ID*`.httpd.conf`, but any valid configuration file may be used.

state

Shows the state of the server for the purposes of starting all instances (such as when the machine boots). Valid values in this column are:

`enable` — Means the server will be started by default when the machine boots or when `htserver start` or `htserver restart` is run with no explicit instance name.

`disable` — Means the server will only be started if it is explicitly named. You must use `htserver start` *instance-ID* to start the instance.

EXAMPLES

EXAMPLE 1

Do not edit the `/etc/http/httpd-instances.conf` file. This example is shown for reference only:

```
Secure_Sites    /etc/http/secure.httpd.conf     disable
Small_Sites     /etc/http/small.httpd.conf      disable
Large_Sites     /etc/http/sws_server.httpd.conf enable
admin   /usr/http/admin_server/conf/admin.httpd.conf   disable
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

SEE ALSO

htIntro(4), **htserver**(1m)

NOTES

Do not edit `httpd-instances.conf` manually. Use **htserver**(1m) to add, delete, or modify instances.

NAME

httpd.conf – Server instance configuration file.

SYNOPSIS

`/etc/http/`*instance-ID*`.httpd.conf`

DESCRIPTION

The server instance configuration file contains directives that define the server's runtime behavior, the web sites it hosts, and the network connection end points it uses.

The directives are grouped in three major divisions. The valid directives for each division are explained in separate sections below. The major divisions are:

`server { }`	Defines the behavior of the server process. Includes cache settings, the user ID of the server process, servlet engine settings, and server-wide defaults for the web sites hosted by the instance.
`url` *hostname* `{ }`	Defines the identity and server resources used by a web site hosted by the server instance. There is one `url {}` block for each web site (or virtual host) handled by the instance, and it defines properties such as the web site's canonical host name and aliases, the location of the configuration file, and the network connections available to the web site.
`port` *number* `{ }`	Defines the settings for a network connection or connections. The server process will attempt to bind to each port on each IP address defined in its `port {}` blocks. If a port is in use, the server will log an error. The server will exit if it cannot bind to at least one port. Port directives include the IP addresses on which to use the port and the timeout period.

Syntax

The following syntax rules apply to the `httpd.conf` file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored
- White space is ignored in directive definitions.
- Some directives accept a list of values. Separate multiple values by white space. If more than one line is required to list all values, escape all but the last newline with a backslash (\) at the end of the line.
- Any value may optionally be enclosed in double quotes (").

- All directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, newlines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks that also use matched curly braces to contain a definition.

The overall format of the file is:

```
server {
   <server directives>
}
url  {
   site_path <absolute_pathname>
   site_conf <filename>
   <web site directives>
}
[ url <hostname> { <web site directives> } ]...
port <number> {
   <port directives>
}
[ port <number> { <port directives> } ]...
```

There must be only one `server {}` block. There must be at least one `url {}` block, and each block–except for the default site—must have a unique *hostname*, a `site_path` directory that exists, and a `site_conf` file that exists in the `site_path`. There must be at least one `port {}` block; duplicate port numbers are allowed as long as the IP address definitions are different.

See the "*Extended Description*" for the syntax of all valid directives in each division.

EXTENDED DESCRIPTION

`server {}` Block Directives

`access_enable yes | no`

Enables or disables the use of access control lists (ACLs) to control access to resources on the server. The default is `yes`.

`cache_enable yes | no`

Enables or disables server-side caching of documents. The default is `yes`. Caching can improve the performance by avoiding file system accesses for frequently requested static documents. Dynamic content, such as CGI output, is not cached.

`cache_large_file_cache_size` *MB*

Sets the size in MB of the server-side cache for large files. The large file cache caches files larger than 16 KB. The largest file size cached can be set with `cache_max_file_size`. The default value is 256 MB, and the maximum is only limited by virtual address space of the Sun™ WebServer™ process. You should rarely need to adjust this parameter.

`cache_max_file_size` *MB*

Sets the size of the largest file that will be cached in the large file cache. Documents that are larger than `cache_max_file_size` will never be cached. Caching works best by serving a large number of frequently requested documents. If the cache is filled with only a few very large documents, caching performance will be poor. The default `cache_max_file_size` is 1 (MB).

`cache_small_file_cache_size` *MB*

Sets the size in MB of the server-side cache for small files. The small file cache contains only documents smaller than 16 KB. Larger documents are stored in the large file cache or not cached at all. The default `cache_small_file_cache_size` is 8 (MB).

`cache_verification_time` *seconds*

Sets the number of seconds the server will wait before verifying the validity of a cached file. When Sun WebServer retrieves a file from the cache, if it has not been verified in `cache_verification_time` seconds, it is re-verified with the file on disk. If the dates are different, the file is removed from the cache and the actual file is retrieved. Cached files may be out of date if the actual file has changed. If your files only rarely change, you may want a higher value for `cache_verification_time` for improved performance. The default is 10 (seconds).

`cgi_error_log_cycle_time` *[[days,]hours:]minutes*

Sets the maximum age for CGI error log files. If a CGI error log is older than the number of `cgi_error_log_cycle_time` minutes, then a new request log file is started with an incremented sequence number. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. The log cycle time can be specified in days, hours, or minutes. For example, a log cycle time of 24 hours can be 1,0:0 (1 day), 24:0 (24 hours), or 1440 (1440 minutes). Default is 1 day.

`cgi_error_log_enable yes | no`

Enables or disables CGI script error logging. The default is no.

`cgi_error_log_max_files` *integer*

Sets the maximum number of CGI error log files that Sun WebServer will keep for this host. Log files end with a sequence number suffix which is incremented when a new log file is created. If the sequence goes beyond `cgi_error_log_max_files`, it is reset to 1 and the first CGI error log file is overwritten. This prevents the number of log files from growing without limit. Set `cgi_error_log_max_files` to "-1" to have no limit on the number of log files. Default is 7 files.

`cgi_error_log_max_file_size` *bytes*

Sets the maximum file size, in bytes, for CGI error log files for the current host. If a CGI error log exceeds `cgi_error_log_max_file_size`, a new log file is started with an incremented sequence number. Default is 1048576 bytes (1 MB).

`cgi_error_log_prefix` *prefix*

Sets the directory and log file name prefix for CGI script error logs for a given server. The string can have either an absolute or a relative path name followed by the prefix that will be used for this server's log files. Separate servers must have different prefix names so that there is no conflict in writing to the logs. Each server's log files will have a name in the form:

`<pathname>/prefix.<sequence>`

For example, `<server_root>/logs/error_log.2`. Sequence is incremented and a new file created whenever `cgi_error_log_cycle_time` or `cgi_error_log_max_file_size` is reached. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is "`logs/error_log`".

`comment "`*description*`"`

Simply a comment string to describe the current configuration file.

`cookie_enable yes | no`

Specifies whether the server sends cookies to the client. A cookie is an HTTP header that consists of a text-only string that gets entered into the memory of the client's browser. Session IDs are carried by the cookies in servlet sessions. Default is "`yes`".

`default_file` *file* [*file*`...`]

Lists in order of preference the name of the file Sun WebServer will look for in a directory when a URL request does not name a specific file. For example, if the URL request is for `http://hostname/`, Sun WebServer will look at the top directory of the host name's `doc_root` for the file specified as the *default_file*.

If `default_file` does not appear in the configuration file, then "`index.html`" is used. If `default_file` is set to an empty string (""), then no default file is used. If multiple files are specified, then the files are used in the specified order.

If no file matching the values for `default_file` is found, the directory contents will be listed, subject to the value of `directory_listing`.

The built in server-wide default is "`index.html`".

`directory_listing fancy | simple | off`

Specifies how the contents of directories will be listed if no file matching `default_file` is found in the directory. `directory_listing` is valid in `server {}` blocks as server-wide defaults or in `url {}` blocks for per-host settings.

`fancy`	Displays directory contents with each name as a hyperlink to the file, icons matching each file's type, and file size and date information. The icons used and the association of icons to file suffixes is configurable.
`simple`	Displays only each file name as a hyperlink to the file itself.
`off`	Disables displaying directory contents; an HTTP "404 Not Found" error is returned to the client instead.

The built in server-wide default is "`fancy`".

`error_document` *http_error_code_url*

Allows customized error messages to be returned to the client. Any valid URL, including CGI scripts, may be returned, so you have flexibility in what information you want to give clients when an error occurs. When an HTTP error code is returned, Sun WebServer will return an HTTP "302: Document Moved" status with a Location: header indicating the file to which the error has been remapped. Most clients will automatically fetch the URL named by the Location: header. You can redirect the following HTTP error codes:

- 400 - Bad Request (Remapped by default host only)
- 403 - Forbidden
- 404 - Not Found
- 412 - Precondition Failed
- 500 - Server Error
- 501 - Not Implemented
- 503 - Service Unavailable

The destination URL can be relative to the current or default host if it begins with a "/". Otherwise, an absolute URL must be specified. The following examples show a relative and absolute URL, respectively:

- error_document 503 "/cgi-bin/error.pl?503"
- error_document 500 "http://www2.A.com/mirror/"

Note - The path to which you remap 404 errors must be available in the document root. You can not redirect 404 errors to aliased directories. `error_document` is valid in `server {}` blocks as server-wide defaults or in `url {}` blocks for per-host settings.

`icon_add` *alt_text bitmap_URI file_type*[*file_type*]...

Allows association of a file extension with an icon file. The icon will be used to represent all files with the extension in `fancy` directory listings. `icon_add` can also change a default association or assign icons to file types not covered in the default set; for example, you may want to use your own icons to represent basic types. In addition to file suffixes, you can customize the icon used for “parent directory” (`../`) and “subdirectory” by specifying “UP” or “DIR” as the *alt_text*.

alt_text	Specifies a string that will be used instead of an icon in text-only browsers. For example, "`GIF`".
bitmap_URI	Specifies a URI path relative to the default host (in the `server {}` block) or the host named by the current `url {}` block. For example, "`/icons/binary.xbm`".
file_suffix(es) **|** ***content_type(s)***	Specifies a string of one or more file extensions or content type definitions that will use the icon in "`fancy`" directory listings.

In addition to file suffixes, you can customize the icon used for "parent directory" (`../`) and "subdirectory" by specifying "UP" or "DIR" as the *alt text*. `icon_add` is valid in `server {}` blocks as server-wide defaults or in the site configuration files for per-host settings.

For example,

```
icon_add "IMG" "/sws-icons/image.xbm" "gif jpeg xbm"
```

displays the icon in "`/sws-icons/image.xmb`" for "`gif`" "`jpeg`" and "`xbm`" files . In text-only browsers, the text "`IMG`" is displayed.

`icon_default` *bitmap_URI*

Sets the icon used for files with extensions that do not have a defined icon type. The path to the bitmap file must begin with a "`/`", and it is relative to the default host or the host defined by the current `url {}` block. `icon_default` is valid in `server {}` blocks as server-wide defaults or the site configuration files for site-level settings.

`lwp_threads_count` *integer*

Sets the total number of lightweight process (LWPs) threads in the kernel that Sun WebServer will attempt to use to map to user threads in its thread pool. By default, Sun WebServer uses 1 LWP per user thread (`lwp_threads_count` will equal `threads_n_active`).

Increasing this number may increase the actual thread concurrency of Sun WebServer, and it will also increase the share of system resources used by the httpd process.

`mime_file` *relative_path*

Contains the default MIME types definitions used for the server host. Default is `/etc/http/mime.types`.

If no MIME type for the file can be found, the `mime_default_type` is used. Entries in the `mime_file` have the form:

`<media type>/<media subtype> <file suffix(es)>`

For example: `text/html html htm`. The default server-wide `mime_file` is `/etc/http/mime.types`.

`mime_default_type` *type/subtype*

Sets the MIME type that will be used for files whose extension does not match any other MIME type. The default is "`text/html`". `mime_default_type` is valid in `server{}` blocks as server-wide defaults or in the site configuration files for per-host settings.

`se_session_enable yes | no`

Specifies whether sessions are supported for all servlet engines. A session is a series of requests from the same user that occur during a time period. If set to "`no`", servlet engines do not extract or insert session information into requests and no session swapping will be performed. If set to "`yes`", session information will be extracted and inserted as necessary, and session swapping is also performed when necessary. Default is "`yes`".

`se_session_persistence yes | no`

Specifies whether the server should keep session data persistent. If set to "`yes`", serializable data in sessions are swapped to disk before the server shuts down and revalidated from disk when the server restarts. If set to "`no`", the server removes swapped sessions each time it starts. Default is "`yes`".

`server_admin` *email_address*

Specifies the email address of the Sun WebServer administrator.

`server_classpath` *path*`[:`*path*`]...`

Specifies the Java™ classpath where the JDK, JSDK, and classes for the servlet engine are located. This is not the path for servlets to be loaded.

Changes to this parameter will not take effect until the server instance has been stopped completely; see the NOTES section for details.

`server_java_initial_heap_size` ***bytes***

Specifies how much memory is allocated for the heap when the Java virtual machine (JVM) starts. This directive has the same fuctionality as the command-line option –`ms` in Java. Default is 1048576 bytes (1MB). Changes to this parameter will not take effect until the server instance has been stopped completely; see the NOTES section for details.

`server_java_max_java_heap_size` ***bytes***

Specifies the maximum heap size for the interpreter. This directive has the same fuctionality as the command-line option –`mx` in Java. Default is 16777216 (16MB). Changes to this parameter will not take effect until the server instance has been stopped completely; see the NOTES section for details.

`server_java_max_stack_size` ***bytes***

Specifies the maximum stack size for Java code for each Java thread. This directive has the same fuctionality as the command-line option –`oss` in Java. Default is 409600 bytes (400KB). Changes to this parameter will not take effect until the server instance has been stopped completely; see the NOTESsection for details.

`server_java_max_native_stack_size` ***bytes***

Sets the maximum stack size for native code for each Java thread. This directive has the same fuctionality as the command-line option –`ss` in Java. Default is 131072 bytes (128KB). Changes to this parameter will not take effect until the server instance has been stopped completely; see the NOTESsection for details.

`server_java_properties` ***key*=*val*** **[key=val]...**

Specifies new properties for the system properties list in Java. This directive has the same fuctionality as the command-line option –`D` in Java. Changes to this parameter will not take effect until the server instance has been stopped completely; see the NOTES section for details.

`server_root` *absolute_path*

Specifies the path in which the server-wide scripts, icons, and base for sites hosted by the instance are stored. CGI and servlet error log file prefixes are relative to `server_root` when the log prefixes are specified without a leading "/".

`server_user` *username*

Sets the user name Sun WebServer will use after start up. The string must be a valid user name on the system. After httpd is started by root, it will change to the server_user user name. The default `server_user` is `root`. The `server_user` must be able to read and write the configuration file, any ACL files, read files in the `doc_root` directories, and be able to read and write files in any log file directories. You can set the `server_user` so the server does not run with root permissions. By running as a different user, there are fewer security risks because the server will be unable to change or serve to the clients sensitive files owned by root. You may want to create a user only to run Sun WebServer, and make sure that this user owns or has access to all of the necessary files and directories.

`servlet_engine { ` *parameters* `}`

Specifies parameters of the servlet engine:

`chaining_enable yes | no`

Enables or disables servlet chaining. This enables the servlet engine to run a sequence of servlets in a specified order to fulfill one single servlet request. Host administrators can specify a chain of servlets to be executed sequentially. Default is "`no`".

`cookie_comment` *comment*

Specifies the value of the comment field in cookies with session IDs. Default is "`Sun Web Server Session Tracking Cookie`".

`cookie_domain` *domain*

Specifies the domain where cookies with session IDs are valid. For example, if a cookie has a domain of "`www.A.com`", then only "`www.A.com`" will recognize it as a valid cookie. All other servers will reject this cookie.

`cookie_max_age` *seconds*

Specifies the value of the `max-age` field sent for cookies with session IDs. A cookie with `cookie_max_age` 0 expires immediately.

`cookie_name` *name*

Specifies the name of a cookie used to carry the session ID when cookies are enabled. Default is "`swssessionid`".

`cookie_path` *path*

Specifies the value of the path field sent for cookies with session IDs. This allows you to set the URL path where the cookie is valid. Pages outside of this path cannot read the cookie. This path is relative to `doc_root`. Default is "`/`".

`cookie_secure yes | no`

Specifies the value of the secure field sent for cookies with session IDs. This directive indicates whether a cookie should only be used under a secure server condition, such as SSL. Default is "`no`".

`dynamic_linking_enable all | local | remote | none`

Allows the specified servlets access to dynamic libraries. Default is "`none`".

`file_access_enable all | local | remote | none` *relative_path*

Allows specified servlets to have access to file resources, for example, read / write a file on local disk. Default is "`local`".

`network_access_enable all | local | remote | none`

Allows specified servlets to have access to network resources, for example, open a socket. Default is "`local`".

`properties_file` *path*

Path to the "`servlets.properties`" file. All loadable servlets are specified in this file. Each servlet engine instance can have a list of preloaded servlets that it wishes to load and initialize as soon as the server starts. This list is specified in a servlet properties file. Path can be either absolute or relative to *server_root*. Default is "`conf/servlets.properties`".

`reload_enable yes | no`

Allows or disallows the servlets to reload. If the servlet classfile changes, a servlet instance (reflecting the changes) can be reloaded. Reloading can be performed by either the server or the host administrator, depending on whether the hosts are sharing a servlet engine instance. Servlets do not reload automatically and will reload only when explicitly requested. Default is "`no`".

`remote_enable yes | no`

Enables or disables remote servlets. Enabling remote servlets allows the servlets from remote sites to be loaded by the server. The default is "`yes`".

`se_log_enable yes | no`

Enables or disables the servlet error logging. The default is “`no`”.

`se_log_cycle_time` *[[days,]hours:]minutes*

Sets the maximum age for servlet error log files for the current host. If a servlet error log is older than the number of `se_log_cycle_time` minutes, then a new servlet error log file is started with an incremented sequence number. If a log file contains no entries, then no new log file will be generated, regardless of how much time has passed. The log cycle time can be specified in days, hours, or minutes. For example, a log cycle time of 24 hours can be 1,0:0 (1 day), 24:0 (24 hours), or 1440 (1440 minutes). Default is 1 day.

`se_log_max_files` *integer*

Sets the maximum number of servlet error log files that Sun WebServer will keep for this host. Log files end with a sequence number suffix which is incremented when a new log file is created. If the sequence goes beyond `se_log_max_files`, it is reset to 1 and the first servlet error log file is overwritten. This prevents the number of log files from growing without limit. Set `se_log_max_files` to “-1” to have no limit on the number of log files. Default is 7 files.

`se_log_max_file_size` *bytes*

Sets the maximum file size, in bytes, for servlet error log files for the current host. If a servlet error log exceeds `se_log_max_size`, a new log file is started with an incremented sequence number. Default is 1048576 bytes (1 MB).

`se_log_prefix` *absolute_path/prefix*

Sets the directory and log file name prefix for servlet error logs for a given servlet engine. The string can have either an absolute or a relative path name followed by the prefix that will be used for this servlet engine’s servlet log files. Separate servlet engines must have different prefix names so that there is no conflict in writing to the logs. Each servlet engine’s servlet log files will have a name in the form:

`<pathname>/prefix.<sequence>`

For example, `<server_root>/logs/se_log.2`. Sequence is incremented and a new file created whenever `se_log_cycle_time` or `se_log_max_file_size` is reached. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is”`logs/se_log`”.

`security_access_enable all | local | remote | none` *absolute_path/prefix*

Allows the specified servlets to have access to security resources, for example, classLoaders. Default is "`local`".

`send_auth_hdrs yes | no`

Specifies whether to send authorization headers to servlets. Authorization headers hold information about client authentication such as encoded user name and password. Default is "`no`".

`session_invalidation_time` *[[days,]hours:]minutes*

Specifies the length of time that a sesssion is allowed to remain unused before it is invalidated and denied further access. Default is 30 minutes.

`session_max_residents` *integer*

Specifies the number of sessions allowed to remain in memory. If the number of sessions exceeds this number, then sessions are swapped out to disk (beginning with the least-recently used session) to reduce the number of resident sessions. Default is 4096.

`session_protocol_switch_rewriting yes | no`

Specifies whether session ID is added to URLs when URL dictates a switch from "http" to "https" or vice-versa. Used only in servlet URL rewriting. Default is "`no`".

`session_swap_directory` *path*

Specifies the directory path where the swapped sessions reside. The path can be either absolute or relative to `server_root`.

`singlethreadmodel_init_pool_size` *integer*

Specifies the initial number of instances of a single servlet to be spawned in the case of SingleThreadModel servlets. Default is 5.

`singlethreadmodel_max_pool_size` *integer*

Specifies the maximum number of instances of a single servlet to be spawned in the case of SingleThreadModel servlets. Default is 20.

`system_access_enable all | local | remote | none`

Allows or disallows the specified servlets to have access to system resources, for example, call `System.Exec()`. Default is "`local`".

`servlets_path` *path*

Specifies the absolute path to local directories and JAR files for all local servlets. This is a colon-separated list.

`server_servlets_enable yes | no`

Allows or disallows servlets to run in this server process. This option controls whether to start the JVM for this process. The default is “no”.

`site_restrictions { `*directives*` }`

Enables access to the following:

`cgi_superuser yes | no`

Allows or disallows `cgi_user` setting of any web site to be `root`. Default is “no”.

`cgi_user_unique yes | no`

Determines whether the `cgi_user` setting of any web site must be unique throughout the server. Default is “yes”.

`cgi_dns_enable yes | no`

Enables or disables the `REMOTE_HOST CGI` environment variable to be set and to be available to CGI scripts. `REMOTE_HOST` requires a DNS lookup of the IP address (`REMOTE_ADDR`) of the resource making the CGI request. Since DNS lookups can be resource consuming, allowing such DNS lookups can slow performance, especially on a server that uses extensive CGI. If you use `getRemoteHost()` or `getRemoteAddr()` or similar calls in your servlet programs requiring name resolution, `cgi_dns_enable` must be set to "`yes`" on both the server and the web site level. If you change this directive, you must restart the server in order for your change to take effect. The default is “no”.

`se_share yes | no`

Enables all web sites to share the servlet engine defined in the server block if set to “yes”. If set to “no”, there is no sharing and each web site can have its own servlet engine. This is a server-wide setting. The default is “no”.

`symlink_follow yes | no`

Follows or ignores symbolic links in the file system. Ignoring symbolic links may cause a performance loss as the file name and each directory in the path of a requested resource must be checked to make sure there are no symbolic links. Following symbolic links may be a security risk because a symbolic link can potentially point to a file that is outside of the `doc_root`. A symbolic link to a sensitive file (such as `/etc/passwd`) can only be made by someone with write access to the file, so the security risk is often small and easily managed by controlling who has access to the document root.

`symlink_follow` is valid in `server {}` blocks as server-wide defaults or in the site configuration files for per-host settings. Default is "`yes`".

`threads_n_active` *integer*

Specifies the maximum number of user threads Sun WebServer will have available in its thread pool. The number of threads will not grow beyond this number. Sun WebServer uses one thread per connection, releasing the thread to the thread pool when the request has completed. For keepalive connections, the thread is released to the thread pool after a request has completed, and a new thread is used if there is a new request on the connection. The number of threads sets an upper limit on the number of simultaneous connections Sun WebServer can handle. The default value is 128 threads.

`url_rewriting_enable yes | no`

Specifies whether the server uses rewritten URLs as a vehicle to carry the session ID of a servlet session. Rewritten URLs are URLs with session IDs embedded in them. The server also recognizes session IDs in the incoming URLs. Default is "`no`".

`user_doc_source UNIXSYS | ISP`

Specifies the source of user information for user document directories if `user_doc_enable` is "yes." In most cases, the only valid value is UNIXSYS, and users are defined through the operating system (for example, in `/etc/passwd` or NIS).

In the Solaris ISP Server™ software, if virtual FTP servers have been defined in Sun Directory Services, you may set this to `ISP`. User information will be taken from the directory server, and the value of a user's `ispContentDirectory` will be used.

`version` *version_string*

Describes the current version of Sun WebServer.

`url {}` Block Directives

`alias` *hostname*

Defines other names for the specified virtual host.

`conn_end_points [`*ip_address* `| *]:`*port* `[[`*ip_address* `| *]:`*port*`]...`

Determines the IP address and port numbers on which requests to this web site are accepted. The syntax of this directive is

`[<ip_address>]:port number`

If the `ip_address` refers to the set of all IP addresses for this web site, then for HTTP 1.1 virtual hosts, leaving the `ip_address` field blank means that it is available on all IP addresses.

This directive may not appear in the `url {}` block for the default virtual host. In consequence, the default host will receive all requests to unknown hosts and all HTTP 1.0 requests not addressed to any host that arrives on any port. The server rejects with error 400 HTTP 1.1 requests not addressed to any host.

`owner_group` *groupname*

Defines the UNIX group who owns the web site's content files.

`owner_user` *username*

Defines the UNIX user and group who own the web site's content files.

`site_config` *path*

Specifies the relative path to a web site's configuration files.

`site_enable yes | no`

Determines whether the site is currently enabled and accepting requests. `site_enable` is not an initialized setting. A site is enabled once it has been configured successfully. After successful configuration, use `hthost` to enable or disable the site.

`site_path` *directory_path*

Specifies the absolute path location of the web site.

`port {}` Block Directives

`expected_load low | medium | high`

Specifies the anticipated level of request traffic on this port. Setting this directive to "low" indicates that the number of incoming requests on this

port is expected to be low, so Sun WebServer dedicates fewer resources to handle the requests on this port. If set to "high", that the number of incoming requests on this port is expected to be high, so Sun WebServer allocates more resources to handle the requests on this port. The "medium" setting is appropriate for most situations. However, setting ports with high traffic (hundreds of requests per second) to "high" will improve the throughput of Sun WebServer, and setting ports with little traffic to "low" will improve overall resource allocation in Sun WebServer. The default is “medium”.

`ip_address` *`nnn.nnn.nnn.nnn`*

Indicates the IP address on the server that can receive requests on the current port. Use this if you do not wish to support all IP addresses on a port. You need to create a separate `port {}` block with the same port number for each specific ip_address you want to support. Use separate `port {}` blocks with unique `ip_address` settings to support IP-based virtual hosting. If this parameter is omitted, all IP addresses on the server will be supported on the port. The default is all IP addresses.

`keepalive_enable yes | no`

Allows or disables HTTP 1.0 keepalive connections on the current port. HTTP 1.1 connections always use keepalive, but HTTP 1.0 browsers can only establish a keepalive connection with Sun WebServer by sending a Connection: keepalive HTTP header (if `keepalive_enable` is set). Keepalive may improve performance since the connection is not destroyed and reestablished for each HTTP request. The default is "`yes`".

`request_timeout` *seconds*

Sets the maximum time, in seconds, that Sun WebServer will wait to fill an individual client request on the current port. The default is 180 seconds (three minutes).

`ssl_client_cert_required yes | no`

Determines whether the server will demand a certificate signed by a Certificate Authority (CA) known to the server when a client connects to the SSL port. The default is "no".

`ssl_enabled yes | no`

Enables or disables the Secure Sockets Layer (SSL). SSL encrypts and authenticates messages sent between a browser and Sun WebServer. Encryption using public key cryptography ensures the privacy of the messages sent between the client and Sun WebServer. Port 443 is the default SSL port and is recommended for easiest use by clients (no port will need to be specified in the `https` URL). The default is "no".

Note - To run SSL, you will need to set up a local root Certificate Authority (CA) with a Distinguished Name record, and generate a private/public key pair for the local root CA. The local root CA will be able to generate credentials and key pairs for every SSL-enabled host within your organization, whether it is a single host or a machine running Sun WebServer with hundreds of virtual hosts. The security tools use the Federated Naming System (FNS) to manage the naming context for users and hosts with certificates.

`ssl_ciphers` *cipher_string(s)*

Sets the cipher parameters used for SSL encryption. It may be one of the following:

Note - For domestic software, to ensure successful operation with various browsers, always include the strongest available cipher choice (`SSL_RSA_WITH_RC4_128_MD5`) in the `ssl_ciphers` attribute when you enable SSL on a port.

SSL_RSA_EXPORT_WITH_RC4_40_MD5

For 40–bit exportable ciphers. This is the default setting.

SSL_RSA_WITH_RC4_128_MD5

For 128–bit, North America only cipher (this requires separate SSL packages not available in the downloadable version of Sun WebServer).

EXAMPLES

EXAMPLE 1

A sample configuration file:

```
# Automatically generated.
#
#       SWS HTTP server configuration file
#
# file version number
version "SWS2.1"
server {
        comment                 "Sample Configuration"
```

```
        server_root             "/var/http/sample/"
        server_user             "root"
        cache_enable            "yes"
        cache_large_file_cache_size 256
        cache_max_file_size     1
        cache_small_file_cache_size 8
        cache_verification_time  10
        cgi_error_log_cycle_time 1,0:0
        cgi_error_log_enable    "no"
        cgi_error_log_max_files  7
        cgi_error_log_max_file_size 1048576
        cgi_error_log_prefix    "error_log"
        cookie_enable           "yes"
        lwp_threads_count       1
        threads_n_active        128
        access_enable           "yes"
        symlink_follow          "yes"
        user_doc_source         "ISP"
        directory_listing       "fancy"
        default_file            index.html
        mime_default_type       "text/html"
        mime_file               "/etc/http/mime.types"
        icon_default            "/sws-icons/unknown.xbm"

        site_restrictions {
                se_share                "no"
                cgi_superuser           "no"
                cgi_user_unique         "no"
                cgi_dns_enable          "no"
        }
        server_classpath        "/usr/lib/http/classes.zip:\\
       /usr/jsdk/lib/classes.zip:/usr/java/lib/classes.zip"
        server_servlets_enable  "yes"
        se_session_enable       "yes"
        se_session_persistence  "yes"
        url_rewriting_enable     "yes"
}
url {
        site_enable             "yes"
        site_path               "/var/http/sample/websites/default_sites"
        site_config             "conf/default_sites.site.conf"
        owner_user              "admin3"
}

port 80 {
        ip_address          129.146.146.146
        keepalive_enable        "yes"
        request_timeout         180
}
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO **htIntro**(4), **htserver**(1m), **hthost**(1m), **httpd.site.conf**(4)

NOTES All configuration parameters are editable through the Sun WebServer GUI. For administrators who use the command-line utilities or do not have access to the Sun WebServer GUI (for example, if the administrator is remote and does not have access through a firewall), there are some directives that can not be changed using the command-line utilities. These directives must be manually edited in the configuration files. In this case, the changes made directly in the configuration files may conflict with the edits from the GUI and the command-line utilities.

All non-servlet related directives, all `port{}` block directives, web site alias directives (in `host{}` block), and the `conn_end_points` directive (in `host{}` block) are not explicitly supported in the command-line utilities. If they are modified through an editor, then the synchronization of file writes are handled through UNIX (outside of the file locking mechanism used by both the administration and the command-line utilities). This could lead to file inconsistencies not encountered if one performed management exclusively through the Sun WebServer GUI.

Because running the command-line utility `htserver restart` or restarting the server from the Sun WebServer GUI does not restart the Java virtual machine, if you change the `server_java_*` or `server_classpath` directives, you must stop the server instance completely (`htserver stop`) the start it again for your changes to take effect.

When the CGI and servlet logs are enabled, they capture output from the standard error streams of CGI scripts and servlets in cycled log file sets that Sun WebServer manages. Enabling these logs can impair Sun WebServer's throughput because scripts and servlets often use the log files for debugging messages which can flood the log files; this file system traffic can degrade performance on a heavily loaded web site. However, to debug CGI scripts and servlets, or to record all the CGI and servlet output, you must enable the error log. You can minimize the disk space requirements by carefully choosing the cycling parameters from the server configuration screens in the Sun WebServer GUI.

NAME

httpd.event.logs – Description of the Events Log files.

SYNOPSIS

`/var/adm/messages`

DESCRIPTION

The httpd server puts out error messages and warnings via **`syslogd`**(1m) to `/var/adm/messages` by default. Use `/etc/syslog.conf` to change the default location.

Syntax

The definitions in the `httpd.event.logs` file have the following format:

```
<time> <host> sws.<instance>[pid]: \
[<version> <message_id> <seq_no> (<source>) <severity>]:  <message>
```

Errors sent to the screen before the daemon has been created or started are sent in console format. The format of the messages written to the console will be:

```
<message_id> <time> <severity>:  <message>
```

The following keyword directives are valid in the `httpd.event.logs` file:

time The date and time (in the format: MM DD hh:mm:ss) that the error occurred.

host The node name of the host (`uname –n`).

tag [pid] A tag and a PID form the framework for error messages from user space. The tag reveals which Sun™ WebServer™ daemon logged the message. The instance name spans multiple starts and stops of a single server. The tag contains the instance name of the server preceded by "`sws`". The PID contains the process ID of the process that generated the error message.

version The version of the error message format.

message_id The unique identifier for the error message with the form *modulename.id_number*. The *modulename* refers to a shared object or to a subsystem in the httpd daemon code. The *id_number* mirrors the message catalog number.

seq_no A sequence number determines the exact sequence of error messages in the log. This field is always "0" for Sun WebServer.

source This field contains the source of the error. The source is always “SW” for Sun WebServer.

severity The errors can be one of four severity levels:

CRITICAL Critical condition such as failure to daemonize.

ERROR Error conditions such as server being unable to bind to port. Error conditions cause a significant part of the server to fail.

WARNING Warning conditions such as being unable to write to a log file. Warning conditions do not prevent the server from running.

NOTICE Normal but significant conditions. These may require special handling.

message The actual text of the message.

EXAMPLES

EXAMPLE 1

Sample event log entries:

```
May 17 15:39:10 hostname sws.admin[14454]: [1 httpd.134 0 (SW) NOTICE]: \
Received a SIGHUP signal; restarting the server.
May 17 15:39:10 hostname sws.admin[14454]: [1 admin.195 0 (SW) NOTICE]: \
Running with SWS Configuration file \
"/usr/http/admin_server/conf/admin.httpd.conf".
May 18 15:35:33 hostname sws.Secure_Sites[15257]: [1 net.61 0 (SW) WARNING]: \
Network interface 129.146.146.25 is not configured on this host.

May 18 15:35:33 hostname sws.Secure_Sites[15257]: [1 net.183 0 (SW) ERR]: \
httpd cannot bind to any configured port

May 18 15:36:49 hostname sws.Large_Sites[15077]: \
[1 servlet.231 0 (SW) WARNING]: Servlets not allowed in this server.\
 The specified servlet-engine block in virtual host www.A.com will be ignored
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Evolving

SEE ALSO **htIntro**(4), **syslogd**(1m), **syslog.conf**(4)

NOTES To ensure that all Sun WebServer messages are logged by **syslogd**(1m), create an entry for `daemon.notice` messages in **syslog.conf**(4).

For example, to log Sun WebServer messages to `/var/adm/messages`, create the following entry:

```
daemon.notice    /var/adm/messages
```

Note - The white space between the message type and the file name consists only of tabs.

This entry would cause all messages with a severity of “notice” or greater generated by any daemon process to be logged to `/var/adm/messages`.

NAME

httpd.request.logs – Description of the Sun™ WebServer™ request log files.

SYNOPSIS

site_path/log-prefix

DESCRIPTION

The `httpd.request.logs` file logs all incoming requests to a server request log file. Sun WebServer generates log files using one of three log file formats configurable by the administrator: the Common Log Format (CLF), the Extended Common Log Format (ECLF) also known as the Combined Log Format, or the Extended Log Format (ELF). The location of a request log file is set by the log-prefix directive in the *site_path/conf/httpd.site.conf* or in the Sun WebServer GUI.

The most recent log file can be accessed through the log prefix. If the default prefix `http_log` located at *site_path*/http_log has the ELF log type, then apart from having log files with names /*site_path*/http_log.elf.1, you will also have a symbolic link called *site_path*/http_log.elf, which points to the latest log file that the server is writing.

Note - If you are running on the Solaris ISP Server™ software and you are also running the network cache accelerator (NCA), check the NCA log file (specified with the `logd_path_name` directive in `/etc/ncalog.conf`) for additional log entries. Sun WebServer does not record all requests in the site request log on ports where NCA is running. Also, note that cache misses may be logged in both the Sun WebServer site request log as well as the NCA log, and some events such as requests for the site on ports not running NCA may be logged only in the Sun WebServer site request log.

Common Log Format (CLF)

The CLF format is used by most HTTP servers and analysis tools.

To get CLF logs, set `log_type` to `''clf''` in the web site configuration file.

The entries in the Common Log File (CLF) logs have the following format:

```
host rfc931 authuser [DD/MM/YYYY:hh:mm:ss] "request" ddd bbbb
```

The following list explains the log entry fields::

host	The DNS name or the IP number of the remote client.
rfc931	The information returned on this client for this request, otherwise (-).
authuser	The user ID sent for authentication, otherwise (-).

DD/MM/YYYY:hh:mm:ss	The date and time of the request.
request	The first line of the HTTP request as sent by the client.
ddd	The HTTP status code returned by the server, if not available (-).
bbbb	The number of bytes sent, not including the HTTP header, if not available (-).

Extended Common Log Format (ECLF)

An extended common log format file is a variant of the common log format file with two additional fields at the end of the line, the referrer and the user agent fields.

To get ECLF logs, set `log_type` to `''eclf''` in the web site configuration file.

The entries in the ECLF log files have the following format:

```
<host> rfc931 authuser [DD/MM/YYYY:hh:mm:ss GMT_offset] \
"request" ddd bbbb "referrer" "user_agent"
```

The following list explains the log entry fields:

host	The DNS name or the IP number of the remote client.
rfc931	The information returned on this client by `identd` for this request, otherwise (-).
authuser	The user ID sent for authentication, otherwise “-”.
DD/MM/YYYY:hh:mm:ss	The date and time of the request.
GMT_offset	The difference between the local time and Greenwich Mean Time (GMT).
request	The first line of the HTTP request as sent by the client.
ddd	The status code returned by the server, if not available (-).
bbbb	The number of bytes sent, not including the HTTP header, if not available (-).

referrer The URL the client was on before requesting your URL, if not available, (-).

useragent The software the client claims to be using, if not available (-).

Extended Log Format (ELF)

The Extended Log File (ELF) format is a flexible format for recording HTTP requests, which is particularly suited for log analysis tools. ELF records more information than the CLF format. It contains a sequence of lines containing ASCII characters delimited by a new line. Lines that start with # are comment directives.

To get ELF logs, set `log_type` to `''elf''` in the web site configuration file.

The entries in the ELF log files have the following format:

date time cs-method cs-uri sc-status time-taken bytes cs-ip cs-host

The following list explains the log entry fields:

date The date in YYYY-MM-DD format.

time The time the request was accepted by the server in the 24–hour format in the machine's time zone.

cs-method The client-to-server HTTP method.

cs-uri The client-to-server requested URI.

sc-status The server-to-client HTTP status code in the response.

time-taken The time taken to process the request.

bytes The number of bytes sent.

cs-ip The client-to-server IP address and port of the client.

cs-host The DNS name of the remote client, if not available "-".

EXAMPLES

EXAMPLE 1

Extended log format (elf):

```
#Version: 1.0
#Software: Sun_WebServer/2.1
#Start-Date: 1998-05-18 17:57:20
```

```
#Fields: date time cs-method cs-uri sc-status time-taken bytes \
cs-ip cs-host cs-referer cs-agent
1998-05-18 17:57:01 GET / 200 0 848 129.146.114.74:49028 -

1998-05-18 17:57:13 GET /swshelp/ht_helpTOC.doc.html \
304 0 142 129.146.114.74:49028 -

1998-05-18 17:57:20 GET /swshelp/ht_server_cache_settings.html \
304 0 142 129.146.114.74:49028 -

1998-05-18 17:57:21 GET /swshelp/frame.ht_server_cache_settings1.html \
304 1 142 129.146.114.74:49032 -

1998-05-18 17:57:21 GET /swshelp/SWSbanner.gif \
304 0 142 129.146.114.74:49032 -
```

EXAMPLE 2

Common log format (clf):

```
129.146.114.74 - - [18/May/1998:17:47:58 -0700] \
"GET /monhelp/SISP_Banner.html HTTP/1.0" 200 558
129.146.114.74 - - [18/May/1998:17:48:11 -0700] \
"GET /swshelp/ht_server_cache_settings.html HTTP/1.0" 200 379

129.146.114.74 - - [18/May/1998:17:48:12 -0700] \
"GET /swshelp/ht_server_cache_settings1.html HTTP/1.0" 200 2645

129.146.114.74 - - [18/May/1998:17:48:12 -0700] \
"GET /swshelp/SWS_Banner.html HTTP/1.0" 200 330

129.146.114.74 - - [18/May/1998:17:48:25 -0700] \
"GET /swshelp/ht_server_web_sites_list.html HTTP/1.0" 200 379

129.146.114.74 - - [18/May/1998:17:49:09 -0700] \
"GET /mchelp/awsTOC.doc.html HTTP/1.0" 200 5953

129.146.114.74 - - [18/May/1998:17:48:12 -0700] \
"GET /swshelp/SWSbanner.gif HTTP/1.0" 200 16712
```

EXAMPLE 3

Extended common log format (eclf):

```
129.146.114.74 - - [18/May/1998:18:02:56 -0700] \
"GET /swshelp/ht_server_web_sites_list.html HTTP/1.0" 304 142 \
"http://isp-doc1/swshelp/ht_helpTOC.doc.html" \
"Mozilla/4.02 [en] (X11; U; SunOS 5.6 sun4m)"
129.146.114.74 - - [18/May/1998:18:03:03 -0700] \
"GET /swshelp/ht_server_iplist2.html HTTP/1.0" 200 2377 \
"http://isp-doc1/swshelp/ht_helpTOC.doc.html" \
"Mozilla/4.02 [en] (X11; U; SunOS 5.6 sun4m)"

129.146.114.74 - - [18/May/1998:18:03:05 -0700] \
"GET /swshelp/ht_server_iplist3.html HTTP/1.0" 200 341 \
"http://isp-doc1/swshelp/ht_helpTOC.doc.html" \
```

```
"Mozilla/4.02 [en] (X11; U; SunOS 5.6 sun4m)"

129.146.114.74 - - [18/May/1998:18:03:02 -0700] \
"GET /swshelp/frame.ht_server_iplist2.html HTTP/1.0" 200 249 \
"http://isp-doc1/swshelp/ht_helpTOC.doc.html" \
"Mozilla/4.02 [en] (X11; U; SunOS 5.6 sun4m)"
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttp
Interface Stability	Stable

SEE ALSO

htIntro(4), **httpd.site.conf**(4)

NAME

httpd.servlet.logs and httpd.cgi.logs – The log files for servlet requests and CGI script errors.

DESCRIPTION

The `httpd.servlet.logs` log errors are generated by servlets. You can specify the location of the log files with `htservlet` or through the Sun™ WebServer™ GUI. See **`htservlet`**(1m) for details on configuring log settings.

The `httpd.cgi.logs` log errors are generated by CGI scripts. You can specify the location of the log files through the Sun WebServer GUI.

Log files will be named *prefix.sequence*, where *sequence* is a cycling number. Each entry is a Common Log File (CLF) format header indicating the request that launched the servlet or CGI, followed by all error messages generated during the request.

The servlet error logs will have the following format:

{client_name [DD/MM/YYYY:hh:mm:ss] "request" "vhostname" "script_file_name"}
error_messages

where *error_messages* is the error output for the servlet.

The CGI error logs will have the following format:

{client_name [DD/MM/YYYY:hh:mm:ss] "request" "vhostname" "script_file_name" [block_number]}
error_messages

where *error_messages* is the error output for the CGI script, and *block_number* identifies the block number of the CGI script. The final block from a script also has a keyword “FINAL” at the end of the header. Since Sun WebServer uses constant buffer sizes and many CGI scripts are using the same error log file, it is possible that error messages from a given script are logged into more than one non-adjacent block. If Sun WebServer could log all the error messages from the script into just one block, then it doesn’t print either the block number or the keyword `FINAL`.

The most recent log file can be accessed through the log prefix. If the *prefix* is `/var/http/logs/se_log`, then apart from having log files with names `/var/http/logs/se_log.1`, you will also have a symbolic link called `/var/http/logs/se_log`, which points to the latest log file that the server is writing.

See **`httpd.request.logs`**(4) for a description of the CLF.

EXAMPLES

EXAMPLE 1

Sample CGI error logs:

```
{129.146.115.80 [14/Mar/1998:11:03:23 -0800] \
"GET /cgi-bin/login HTTP/1.0" www.A.com /tmp/tp/root1/cgi-bin/login} \
Invalid number of parameters
{129.146.115.80 [14/Mar/1998:12:03:29 -0800] \
"GET /cgi-bin/sendfile HTTP/1.0" www.A.com /tmp/tp/root1/cgi-bin/sendfile 1}\
Unable to open file: /var/http/server/websites/public/profiles/file.1

Output not generated.

{129.146.115.80 [14/Mar/1998:13:03:35 -0800] \
"GET /cgi-bin/login HTTP/1.0" www.A.com /tmp/tp/root1/cgi-bin/login 2} \
Authentication failed for user "user1"
```

EXAMPLE 2

Sample servlet error logs:

```
{120.120.120.120 [07/Nov/1998:15:51:52 -0800] \
"GET /servlet/security HTTP/1.0" www.A.com /servlet/security}
cannot exec: exec
{120.120.120.120 [07/Nov/1998:15:51:54 -0800] \
"GET /servlet/network HTTP/1.0" www.A.com /servlet/network}
cannot open url connection: connect

{120.120.120.120 [07/Nov/1998:15:51:56 -0800] \
"GET /servlet/remote HTTP/1.0" www.A.com /servlet/remote}

com.sun.sws.se.ServletSecurityException: file.read
        at com.sun.sws.se.ServletSecurity.checkRead(ServletSecurity.java:473)
        at com.sun.sws.se.ServletSecurity.checkRead(ServletSecurity.java:427)
        at java.io.FileInputStream.(FileInputStream.java)
        at java.io.FileReader.(FileReader.java)
        at RemoteServlet.service(RemoteServlet.java:22)
        at com.sun.sws.se.ServletHandler.servletService(ServletHandler.java:191)
        at com.sun.sws.se.ServletHandler.handle_servlet_service\
(ServletHandler.java:258)

{- [07/Nov/1998:15:51:57 -0800] "- - -" - -}
SimpleServlet: init
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhtsvl
Interface Stability	Evolving

SEE ALSO **htIntro**(4), **httpd.conf**(4), **httpd.site.conf**(4), **httpd.request.logs**(4)

NAME

httpd.site.conf – Site instance configuration file.

SYNOPSIS

site-path`/conf/`*site_name*`.site.conf`

DESCRIPTION

The site instance configuration file contains directives that define the site's runtime behavior, and defines the identity and server resources used by a web site hosted by the server instance. This file defines properties such as the web site's canonical host name and aliases, the location of the configuration file, and the network connections available to the web site. It also defines the servlet engine settings for the web site. The location of `httpd.site.conf` is determined by the site_config directive in `httpd.conf`.

Syntax

The following syntax rules apply to the `httpd.site.conf` file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored
- White space is ignored in directive definitions.
- Some directives accept a list of values. Separate multiple values by white space. If more than one line is required to list all values, escape all but the last newline with a backslash (\) at the end of the line.
- Any value may optionally be enclosed in double quotes (").
- All directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, newlines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks that also use matched curly braces to contain a definition.

The overall format of the file is a list of directives.

See the "*Extended Description*" for the syntax of all valid directives in each division.

EXTENDED DESCRIPTION

Directives

`access_file`

Specifies the path in which the server will look for ACLs for each web site hosted by the server instance. Path can be either absolute or relative to *site_path*. The default is "`conf/access.conf`".

`afp_enable yes | no`

Enables Microsoft FrontPage support.

`cache_control public | private | no_cache`

Sets the caching policy used by intermediate caching proxies. The "`public`" setting allows the cache to be shared with other proxies. The "`private`" setting allows the files to be cached, but the cache can not be shared with other proxies. The "`no_cache`" setting does not allow the cache to be shared with other proxies. The default is "`public`."

`cgi_dns_enable yes | no`

Enables or disables the `REMOTE_HOST CGI` environment variable to be set and to be available to CGI scripts. `REMOTE_HOST` requires a DNS lookup of the IP address (`REMOTE_ADDR`) of the resource making the CGI request. Since DNS lookups can be resource consuming, allowing such DNS lookups can slow performance, especially on a server that uses extensive CGI. If you use `getRemoteHost()` or `getRemoteAddr()` or similar calls in your servlet programs requiring name resolution, `cgi_dns_enable` must be set to "`yes`" on both the server and the web site level. If you change this directive, you must restart the server in order for your change to take effect. The default is "`no`".

`cgi_enable yes | no`

Enables or disables the authority to execute CGI scripts for the host defined by the `url {}` block. If enabled, resources in the CGI directory can be accessed through URLs of the form `http://hostname/cgi-bin/` and executed as CGI scripts. If enabled and `cgi_suffix_enable` is also enabled, then any file with a `.cgi` extension can be executed as a CGI script. Individual sites can override this server setting. The default is "`no`".

`cgi_error_log_cycle_time` *[[days,]hours:]minutes*

Sets the maximum age, in minutes, for CGI error log files for the current host. If a CGI error log is older than the number of `cgi_error_log_cycle_time` minutes, then a new request log file is started with an incremented sequence number. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. The log cycle time can be specified in days, hours, or minutes. For example, a log cycle time of 24 hours can be 1,0:0 (1 day), 24:0 (24 hours), or 1440 (1440 minutes). Default is 1 day.

`cgi_error_log_enable yes | no`

Enables or disables CGI script error logging. The default is `no`.

`cgi_error_log_max_files` *integer*

Sets the maximum number of CGI error log files that Sun™ WebServer™ will keep for this host. Log files end with a sequence number suffix which is incremented when a new log file is created. If the sequence goes beyond `cgi_error_log_max_files`, it is reset to 1 and the first CGI error log file is overwritten. This prevents the number of log files from growing without limit. Set `cgi_error_log_max_files` to "-1" to have no limit on the number of log files. Default is 7 files.

`cgi_error_log_max_file_size` *bytes*

Sets the maximum file size, in bytes, for CGI error log files for the current host. If a CGI error log exceeds `cgi_error_log_max_file_size`, a new log file is started with an incremented sequence number. Default is 1048576 bytes (1 MB).

`cgi_error_log_prefix` *path/prefix*

Sets the directory and log file name prefix for CGI script error logs for a given host. The string can have either an absolute or a relative path followed by the prefix that will be used for this host's log files. Separate virtual hosts must have different prefix names so that there is no conflict in writing to the logs. Each host's log files will have a name in the form:

`<pathname>/prefix.<sequence>`

For example, `<site_path>/logs/error_log.2`. Sequence is incremented and a new file created whenever `cgi_error_log_cycle_time` or `cgi_error_log_max_file_size` is reached. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is "`logs/error_log`".

`cgi_suffix_enable yes | no`

Allows or disallows any file with a `.cgi` extension to be executed as a CGI script. This can allow users to create CGI without having access to `/cgi-bin`, and the Sun WebServer administrator has no control over what CGI the server is executing. This is a potential security risk, so consider which `.cgi` files may be put on your system before enabling this feature. Enabling both `user_doc_enable` and `cgi_suffix_enable` is generally risky since it allows any user to execute arbitrary `.cgi` files through Sun WebServer. Default is "`no`".

`cgi_user` *username*

Sets the user name that will be used to run CGI scripts. By setting the `cgi_user`, you can control which files and services CGI scripts can affect. The `cgi_user` must have execute permission for scripts in a host's `/cgi-bin` directory. Default is "`nobody`".

`content_file` *path*

Specifies the path to the file containing the content for meta data. Path can be either absolute or relative to *site_path*. Default is "`conf/content.conf`".

`content_digest yes | no`

Allows or disallows MD5 checksum to be sent with the response entity. Default is "`no`".

`default_file` *file* [*file*]`...`

Sets the name of the file Sun WebServer will look for in a directory when a URL request does not name a specific file. For example, if the URL request is for `http://hostname/`, Sun WebServer will look at the top directory of the host name's `doc_root` for a file specified as the `default_file`.

If `default_file` does not appear in the configuration file, then "`index.html`" is used. If `default_file` is set to an empty string (""), then no default file is used. If multiple files are specified, then the files are used in the specified order.

If no file matching the values for `default_file` is found, the directory contents will be listed, subject to the value of `directory_listing`.

The built in server-wide default is "`index.html`". The listings are in order of preference.

`directory_listing fancy | simple | off`

Specifies how the contents of directories will be listed if no file matching `default_file` is found in the directory.

`fancy`	Displays directory contents with each name as a hyperlink to the file, icons matching each file's type, and file size and date information. The icons used and the association of icons to file suffixes is configurable.

`simple`	Displays only each file name as a hyperlink to the file itself.
`off`	Disables displaying directory contents; a HTTP "404 Not Found" error is returned to the client instead.

The built-in server-wide default is “`fancy`”.

`doc_root` *relative_path*

Sets the top-level directory available to the host for serving web documents. Except for special cases (such as `/cgi-bin` and users’ personal directories), Sun WebServer will only have access to the file system at the `doc_root` and its subdirectories. This must be set for every host, but it does not have to be unique. You should create a new `doc_root` for your site.

`error_document` *<http error code< <url>*

Allows customized error messages to be returned to the client. Any valid URL, including CGI scripts, may be returned, so you have flexibility in what information you want to give clients when an error occurs. When an HTTP error code is returned, Sun WebServer will return an HTTP "302: Document Moved" header with a Location: header indicating the file to which the error has been remapped. Most clients will automatically fetch the URL named by the Location: header. You can redirect the following HTTP error codes:

- 400 - Bad Request (Remapped by default host only)
- 403 - Forbidden
- 404 - Not Found
- 412 - Precondition Failed
- 500 - Server Error
- 501 - Not Implemented
- 503 - Service Unavailable

The destination URL can be relative to the current or default host if it begins with a (/). Otherwise, an absolute URL must be specified. The following examples show a relative and absolute URL, respectively:

- error_document 503 "/cgi-bin/error.pl?503"
- error_document 500 "http://www2.A.com/mirror/"

Note - The path to which you remap 404 errors must be available in the document root. You can not redirect 404 errors to aliased directories.

`icon_add` *alt_text bitmap_URI file_type*`[` *file_type*`]...`

Allows association of a file extension with an icon file. The icon will be used to represent all files with the extension in `fancy` directory listings. `icon_add` can also change a default association or assign icons to file types not covered in the default set; for example, you may want to use your own icons to represent basic types.

alt text	Specifies a string that will be used instead of an icon in text-only browsers. For example, "`GIF`".
URI path to bitmap	Specifies a URI path relative to the default host (in the `server {}` block) or the host named by the current `url {}` block. For example, "`/icons/binary.xbm`".
file suffix(es) **|** ***content type(s)***	Specifies a string of one or more file extensions or content type definitions that will use the icon in "`fancy`" directory listings.

In addition to file suffixes, you can customize the icon used for "parent directory" (`../`) and "subdirectory" by specifying "UP" or "DIR" as the *alt text*.

For example,

```
icon_add "IMG" "/sws-icons/image.xbm" "gif jpeg xbm"
```

displays the icon in "`/sws-icons/image.xmb`" for "`gif`" "`jpeg`" and "`xbm`" files . In text-only browsers, the text "`IMG`" is displayed.

`icon_default` *bitmap_URI*

Sets the icon used for files with extensions that do not have a defined icon type. The path to the bit map file must begin with a (`/`), and it is relative to the default host or the host defined by the current `url {}` block.

`log_cycle_time` *[[days,]hours:]minutes*

Sets the maximum age for request log files for the current host. If a request log is older than the number of `log_cycle_time` minutes, then a new request log file is started with an incremented sequence number. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. The log cycle time can be specified in days, hours, or minutes. For example, a log cycle time of 24 hours can be 1,0:0 (1 day), 24:0 (24 hours), or 1440 (1440 minutes). Default is 1 day.

`log_max_files` *integer*

Sets the maximum number of request log files that Sun WebServer will keep for this host. Log files end with a sequence number suffix which is incremented when a new log file is created. If the sequence goes beyond `log_max_files`, it is reset to 1 and the first request log file is overwritten. This prevents the number of log files from growing without limit. Set `log_max_files` to "-1" to have no limit on the number of log files. Default is 7 files.

`log_max_size` *bytes*

Sets the maximum file size, in bytes, for request log files for the current host. If a request log exceeds `log_max_size`, a new log file is started with an incremented sequence number. Default is 1048576 bytes (1 MB).

`log_prefix` *path/prefix*

Sets the directory and log file name prefix for request logs for a given host. The string can have either an absolute or a relative path followed by the prefix that will be used for this host's log files. Separate virtual hosts must have different prefix names so that there is no conflict in writing to the logs. Each host's log files will have a name in the form:

`<pathname>/prefix.<log_type>.<sequence>`

For example, `<site_path>/logs/http_log.elf.2`. Sequence is incremented and a new file created whenever `log_cycle_time` or `log_max_size` is reached. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is "`logs/http_log`".

`log_translate_ip_address yes | no`

Determines whether a given virtual host logs its IP address or DNS name into its log files. Default is "`no`" and the IP address is logged.

`log_type none | clf | elf | eclf`

Sets the format for a host's request logs. Valid parameters are:

`none`	Disallows logging performed for this host.
`clf`	Logs in common log file format, a format widely used by servers and analysis tools.
`elf`	Logs in extended log file format.
`eclf`	Logs in extended common log file format; provides more fields than common log format.

If `log_type` is not "`none`", then `log_prefix` must be set. Default is "`elf`".

`map_file` *filename*

Specifies the location of the web site's map file. Path can be either absolute or relative to *site_path*. For information on maps, see the man page for `map.conf(4)`Default is "`conf/map.conf`".

`mime_default_type` *type/subtype*

Sets the MIME type that will be used for files whose extension do not match any other MIME type. The default is "`text/html`".

`mime_file` *relative_path*

Contains the default MIME types definitions used for the server host. Path is relative to `site_path`. Default is "`mime.types`".

If no MIME type for the file can be found, the `mime_default_type` is used. Entries in the mime_file have the form:

`<media type>/<media subtype><file suffix(es)>`

For example: text/html html htm The default server-wide `mime_file` is `/etc/http/mime.types`.

`publish_enable yes | no`

Allows or disallows HTTP methods `PUT` and `DELETE`. The default is "`no`".

If `publish_enable` is set to "`yes`," Sun WebServer implements HTTP `PUT` and `DELETE` requests by redirecting the requests to the executable script

located in *server_root*/cgi-bin/publish_script. This CGI script provides the publishing support for the server. If you create your own publish_script, install it in this directory to replace the implementation provided. The original C source code is provided in /usr/http/src/put/putscript.c so that you can modify the code to suit your individual needs. In the sample server instance sws_server, the server root is /var/http/sws_server/, so the script is located in /var/http/sws_server/cgi-bin/publish_script.

Note - If you enable publishing, you may want to use access control to restrict the users who can use PUT and DELETE methods. Otherwise, any user with access to the web site can modify it using PUT and DELETE. Because Sun WebServer handles many requests simultaneously, it is possible that a PUT or DELETE request is processed in a time interval that overlaps with a GET request for the same file. In some cases, the results of such a GET request may be truncated or contain no data. Subsequent requests (after the PUT and DELETE request has completed) will behave as expected.

---

realm_file *path*

Specifies path to the file containing realm information. Path can be either absolute or relative to *site_path*. Default is "conf/realms.conf".

se_enable yes | no

Enables or disables the servlet engine. The default is "no".

servlet_token *URI path*

Specifies the token which characterizes the request as a servlet request. Default is "/servlet".

ssi_enable yes | no

Enables or disables the ability to use server-side include statements in .shtml files. May be set differently for each virtual host on the server. Default is "no".

ssi_exec yes | no

Enables or disables the ability to use #exec server-side include directive. #exec will execute shell commands or CGI scripts, which may pose security risks and/or slow performance. Default is "no".

`ssi_suffix` *string*

Specifies SSI file suffix. Default is “`.shtml`”.

`ssi_xbithack full | on | off`

Allows any file in the document root with its executable bit set to be treated as a server-parsed HTML file. `ssi_suffix` can be "`full`" or "`on`" only if `ssi_enable` is "`yes`". If "`full`", Sun WebServer treats any file with the user or group executable bit set as server-parsed HTML, and it sends the file’s Last-Modified time in the HTTP header if the user executable bit is set. If "`on`", Sun WebServer only treats files with the user’s executable bit set as server-parsed HTML. Default is “`off`”.

Note - If `cgi_suffix_enable` is also enabled, Sun WebServer will check to see if an executable file is a CGI script before parsing it for server-side includes.

`symlink_follow yes | no`

Follows or ignores symbolic links in the file system. Ignoring symbolic links may cause a performance loss as the file name and each directory in the path of a requested resource must be checked to make sure there are no symbolic links. Following symbolic links may be a security risk because a symbolic link can potentially point to a file that is outside of the `doc_root`. A symbolic link to a sensitive file (such as `/etc/passwd`) can only be made by someone with write access to the file, so the security risk is often small and easily managed by controlling who has access to the document root. Default is “`yes`”.

`user_doc_enable yes | no`

Allows or disallows UNIX users to create personal HTML directories in their home directories. Default is “`no`”.

See the “NOTES” section for information on user directories in a Solaris ISP Server environment.

`user_doc_root` *relative_path*

Sets the directory name that users can create in their home directories for personal HTML files. When a request comes in for `//server/~username/`,

Sun WebServer will look for files in `~username/<user_doc_root>/`. Enables access. Default is "`public_html`".

See the "NOTES" section for information on user directories in a Solaris ISP Server environment.

`user_doc_source` *source*

Specifies the source of user information for user document directories if `user_doc_enable` is "yes." In most cases, the only valid value is UNIXSYS, and users are defined through the operating system (for example, in `/etc/passwd` or NIS).

In Solaris ISP Server, if virtual FTP servers have been defined in Sun Directory Services, you may set this to `ISP`. User information will be taken from the directory server, and the value of a user's `ispContentDirectory` will be used.

See the "NOTES" section for information on user directories in a Solaris ISP Server environment.

servlet_engine{} Directives

Specifies parameters of the servlet engine

`chaining_enable yes | no`

Enables or disables servlet chaining. This enables the servlet engine to run a sequence of servlets in a specified order to fulfil one single servlet request. Host administrators can specify a chain of servlets to be executed sequentially. Default is "`yes`".

`cookie_comment` *comment*

Specifies the value of the comment field in cookies with session IDs. Default is "`Sun Web Server Session Tracking Cookie`".

`cookie_domain` *domain*

Specifies the domain where cookies with session IDs are valid. For example, if a cookie has a domain of "`www.A.com`", then only "`www.A.com`" will recognize it as a valid cookie. All other servers will reject this cookie.

`cookie_max_age` *seconds*

Specifies the value of the `max-age` field sent for cookies with session IDs. A cookie with `cookie_max_age` 0 expires immediately.

`cookie_name` *name*

Specifies the name of a cookie used to carry the session ID when cookies are enabled. Default is "`swssessionid`".

`cookie_path` *path*

Specifies the value of the path field sent for cookies with session IDs. This allows you to set the URL path within which the cookie is valid. Pages outside of this path cannot read the cookie. This path is relative to `site_path`. Default is "`/`".

`cookie_secure yes | no`

Specifies the value of the secure field sent for cookies with session IDs. This directive indicates whether a cookie should only be used under a secure server condition, such as SSL. Default is "`no`".

`dynamic_linking_enable all | local | remote | none`

Allows the specified servlets access to dynamic libraries. Default is "`local`".

`file_access_enable all | local | remote | none` *relative_path*

Allows specified servlets to have access to file resources, for example, read/write a file on local disk. Default is "`local`".

`network_access_enable all | local | remote | none`

Allows specified servlets to have access to network resources e.g. open a socket. Default is "`local`".

`reload_enable yes | no`

Allows or disallows the servlets to reload. If the servlet classfile changes, a servlet instance (reflecting the changes) can be reloaded. Reloading can be performed by either the server or the host administrator, depending on whether the hosts are sharing a servlet engine instance. The default is "`yes`".

`remote_enable yes | no`

Enables or disables remote servlets. Enabling remote servlets allows the servlets from remote sites to be loaded by the server. The default is "`yes`".

`properties_file` *path*

Path to the "`servlets.properties`" file. Each servlet engine instance can have a list of preloaded servlets that it wishes to load and initialize as soon as the server starts. This list is specified in a servlet properties file. Path can be either absolute or relative to *site_path*. Default is "`conf/servlets.properties`".

`servlets_path` *absolute_path*

Specifies the path to all loadable local servlets. This is a colon separated list of directories and jar files where the servlet engine will look for servlets (for example, `/directory/directory:/directory/directory`). Default is "`site_path/servlets`".

`se_log_enable yes | no`

Enables or disables the servlet error logging. The default is "`no`".

`se_log_cycle_time` *[[days,]hours:]minutes*

Sets the maximum age for servlet error log files for the current host. If a servlet error log is older than the number of `se_log_cycle_time` minutes, then a new servlet error log file is started with an incremented sequence number. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. The log cycle time can be specified in days, hours, or minutes. For example, a log cycle time of 24 hours can be 1,0:0 (1 day), 24:0 (24 hours), or 1440 (1440 minutes). Default is 1 day.

`se_log_max_files` *integer*

Sets the maximum number of servlet error log files that Sun WebServer will keep for this host. Log files end with a sequence number suffix which is incremented when a new log file is created. If the sequence goes beyond `se_log_max_files`, it is reset to 1 and the first servlet error log file is overwritten. This prevents the number of log files from growing without limit. Set `se_log_max_files` to "-1" to have no limit on the number of log files. Default is 7 files.

`se_log_max_file_size` *bytes*

Sets the maximum file size, in bytes, for servlet error log files for the current host. If a servlet error log exceeds `se_log_max_file_size`, a new log file is started with an incremented sequence number. Default is 1048576 bytes (1 MB).

`se_log_prefix` *path/prefix*

Sets the directory and log file name prefix for servlet error logs for a given host. The string must have either an absolute or relative path name followed by the prefix that will be used for this servlet engine's log files. Separate servlet engines must have different prefix names so that there is no conflict in writing to the logs. Each servlet engines' log files will have a name in the form:

`<pathname>/prefix.<sequence>`

For example, `<site_path>/logs/se_log.2`. Sequence is incremented and a new file created whenever `se_log_cycle_time` or `se_log_max_file_size` is reached. If a log file contains no entries, then no new log file will be generated regardless of how much time has passed. Default is "`logs/se_log`".

`security_access_enable all | local | remote | none` *path/prefix*

Allows the specified servlets to have access to security resources, for example, classLoaders. The path must be an absolute path. Default is "`none`".

`send_auth_hdrs yes | no`

Specifies whether to send authorization headers to servlets. Authorization headers hold information about client authentication such as encoded user name and password. Default is "`no`".

`session_invalidation_time` *[[days,]hours:]minutes*

Specifies the length of time that a sesssion is allowed to remain unused before it is invalidated. Default is 30 minutes.

`session_max_residents` *integer*

Specifies the number of sessions allowed to remain in memory. If the number of sessions exceeds this number, then sessions are swapped out to

disk (beginning with the least-recently used session) to reduce the number of resident sessions. Default is 4096.

`session_protocol_switch_rewriting yes | no`

Specifies whether session ID is added to URLs when URL dictates a switch from "`http`" to "`https`" or vice-versa. Used only in servlet URL rewriting. Default is "`no`".

`session_swap_directory` *path*

Specifies the directory path where the swapped sessions reside. The path can be either absolute or relative to `site_path`.

`singlethreadmodel_init_pool_size` *integer*

Specifies the initial number of instances of a single servlet to be spawned in the case of SingleThreadModel servlets. Default is 5.

`singlethreadmodel_max_pool_size` *integer*

Specifies the maximum number of instances of a single servlet to be spawned in the case of SingleThreadModel servlets. Default is 20.

`system_access_enable all | local | remote | none`

Allows or disallows the specified servlets to have access to system resources, for example, call `System.Exec()`. Default is "`local`".

EXAMPLES

EXAMPLE 1

A sample web site configuration file with an enabled servlet engine:

```
#
# Copyright (c) 1998, by Sun Microsystems, Inc.
# All rights reserved.
#
# Sun WebServer Site Configuration
url  {
    doc_root    /var/http/sws_server/websites/www.A.com/public
    log_type    eclf
    user_doc_enable     yes

    map_file            conf/map.conf
    realm_file          conf/realms.conf
    access_file         conf/access.conf
    content_file        conf/content.conf
```

```
    # If you need more than the global mime types in
    # /etc/http/mime.types, create this file:
    #
    # mime_file         conf/mime.types

    se_enable                       yes
    servlet_token                   "/servlet/"
    servlet_engine {
        properties_file        /var/http/sws_server/websites/\
    www.A.com/conf/servlets.properties
        servlets_path          //var/http/sws_server/websites/\
    www.A.com/servlets/
        se_log_prefix          "/var/http/logs/A.com_se_log"
        se_log_enable          yes
        se_log_max_files       7
        se_log_max_file_size   1048576
        se_log_cycle_time      1,0:0

        reload_enable          yes
        remote_enable          yes
        chaining_enable        yes
        network_access_enable          local
        file_access_enable             local
        dynamic_linking_enable         local
        system_access_enable           local
        security_access_enable         local

        session_swap_directory         "/tmp/sessionSwap"
        session_max_residents          500
        cookie_name                    "specialSessionId"
        cookie_comment                 "Session Tracking Cookie"
    }
}
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO

htIntro(4), **httpd.conf**(4), **map.conf**(4), **hthost**(1m)

NOTES

If you are running on the Solaris ISP Server™ software, user information (including user content directories) can be accessed through LDAP and Sun Directory Services.

To use LDAP for user content directories, a virtual FTP site matching the Sun WebServer web site (virtual host) must exist in the Directory Services. Refer to the Sun™ Internet FTP Server™ 1.1 online help for information on creating virtual FTP sites.

If `user_doc_source` is set to "`ISP`," Sun WebServer will attempt to map `~user` URLs to the value of `ispContentDirectory` in Sun Directory Services.

The `ispContentDirectory` is relative to the `ispRootDirectory` defined in the `ispService` entry for a virtual FTP site. The file system directory must be readable and writable by either the `uidNumber` or `gidNumber` defined in a subscriber's entry in the directory service.

When the CGI and servlet logs are enabled, they capture output from the standard error streams of CGI scripts and servlets in cycled log file sets that Sun WebServer manages. Enabling these logs can impair Sun WebServer's throughput because scripts and servlets often use the log files for debugging messages which can flood the log files; this file system traffic can degrade performance on a heavily loaded web site. However, to debug CGI scripts and servlets, or to record all the CGI and servlet output, you must enable the error log. You can minimize the disk space requirements by carefully choosing the cycling parameters from the web site management screens in the Sun WebServer GUI.

All configuration parameters are editable through the Sun WebServer GUI. For administrators who use the command-line utilities or do not have access to the Sun WebServer GUI (for example, if the administrator is remote and does not have access through a firewall), there are some directives that can not be changed using the command-line utilities. These directives must be manually edited in the configuration files. In this case, the changes made directly in the configuration files may conflict with the edits from the GUI and the command-line utilities.

All non-servlet related directives (other than `servlet_token`) are not explicitly supported in the command-line utilities. If they are modified through an editor, then the synchronization of file writes are handled through UNIX (outside of the file locking mechanism used by both the administration and the command-line utilities). This could lead to file inconsistencies not encountered if one performed management exclusively through the Sun WebServer GUI.

NAME

map.conf – Creates an alias to a path on the file system or a redirection to a remote URL from a URI on the host.

SYNOPSIS

site_path`/conf/map.conf`

DESCRIPTION

A map directive allows you to redirect requests for a URL on a host to any other URL or to a different directory. `htmap` administers maps to establish an alias to another resource, make a resource outside of the `doc_root` accessible to a client, or partition the name space into various classes of resources such as CGI, imagemap, or servlet. The URL token that is redirected does not have to correspond to any actual resource.

Servlets use maps for creating servlet chains or file aliases.

A map definition consists of the following:

- URI token
- Resource target
- Class type (optional)

Syntax

Each line in the `map.conf` file has the following format:

uri_token resource_target class_type

URI_token Specifies a URI path on the given host beginning with a slash (`/`).

resource_target Specifies a URI path on the given host if it begins with a slash (`/`), or an absolute URL to any network location if it does not begin with a (`/`). For either local or remote redirects, *class_type* must be REMOTE. It can also be an absolute path on the file system if it begins with a (`/`) and the *class_type* is `ADMIN`, `CGI`, `IMAP`, `NULL`, `SERVLET`, or `STATS`.

class_type Specifies the class type. Valid entries are as follows:

Note - *class_type* is not case sensitive.

ADMIN	Treats the resource target as the Sun WebServer administration directory.
CGI	Treats the resource target as a CGI directory. All files located here will be treated as executable scripts.
DOOR	Treats the aliased file or directory as a resource door. Resource doors are multithreaded server daemons which run independently of the web server. With resource doors, Sun WebServer is able to pass incoming requests on to user-developed programs through the Solaris ™ doors mechanism. For more information on Sun WebServer resource doors, refer to the “Site URL Aliases Screen” section in the online help.
IMAP	Treats the resource target as an imagemap alias.
NULL	Treats the resource target as a regular file alias.
REMOTE	Treats the resource target as a remote alias.
STATS	Treats the resource target as an interface to server statistics.
SERVLET	Treats the resource target as a servlet or a chain of servlets. URLs which begin with the aliased token are treated as requests for servlets.

EXAMPLES

EXAMPLE 1

A directory alias can specify “CGI” as a class token, and all files in the directory and its subdirectories will be treated as CGI scripts. Typically, each host has a map directive to define the location of `/cgi-bin`. An example of creating an alias for the `/cgi-bin` is

```
map "/cgi-bin" "/var/http/www/cgi/" CGI
```

EXAMPLE 2

A directory alias can also specify “SERVLET” as a class token, and the aliased *resource_target* is treated as a servlet or a chain of servlets. An example of creating an alias for the *resource_target* is

```
map "/servlets" "servlet1" SERVLET
```

where “`servlet1`” is a servlet defined in `servlet.properties`.

Another example of creating an alias for the *resource_target* is

```
map "/combo" "S1,S2" SERVLET
```

where ”`/combo`” defines a chain of servlets and “`S1`” and “`S2`” are servlets defined in `servlet.properties`.

EXAMPLE 3

If map has only one parameter, it disables any redirects or aliases for that token.

```
map   /cgi-bin    /var/http/public/x/y    CGI
map   /search     /searchServlet    SERVLET
map   /home.html  /var/http/docs/index.html   NULL
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO **htIntro**(4), **htmap**(1m)

NAME

realms.conf – Defines realms of user and group information employed by access control lists on a Sun™ WebServer™ web site.

SYNOPSIS

site_path`/conf/realms.conf` - Web site realms

`/etc/http/realms.conf` - Server realms for server administration

DESCRIPTION

Realms in Sun™ WebServer™ define sets of protection spaces or authentication domains consisting of user names, groups, and passwords. Sun WebServer uses realm information to determine how a user is authenticated. For example, a UNIX-based realm stores user and password information as well as group information in appropriate files or tables if distributed NIS / NIS+ is used. For `HTPASSWD` relams, you can define your own set of users and groups in a realm. Regardless of how the realm information is stored and accessed, the access control settings require realms to protect resources.

Realms are also differentiated based on how they are used. Two different realms can have different names with the same underlying users and groups database. This gives additional flexibility in naming the authentication domains displayed in the browser.

Most browsers display the realm name in the prompt when a user name and password are required, so the realm name should indicate to users the purpose for password protection and which user name and password to use.

Realm files may be created by **`htrealm`**(1m) or by the Sun WebServer GUI.

A realm definition consists of the following:

- Realm identifier
- Source of user information: `HTPASSWD`, `ISP`, `ISPADMIN`, or `UNIXSYS`

Note - `ISP` or `ISPADMIN` realms are only valid if you are running Sun WebServer in an environment where Sun Directory Service for the Solaris ISP Server™ software has been installed.

- List of realm members with permission to modify the realm itself
- Directory location of user information for `HTPASSWD` realms

Syntax

The following syntax rules apply to the `realms.conf` file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored
- White space is ignored in directive definitions.

- Some directives accept a list of values. Separate multiple values by white space. If more than one line is required to list all values, escape all but the last newline with a backslash (\) at the end of the line.
- All directives are grouped in blocks surrounded by curly braces ({ and }). Any amount of white space, newlines, or directive definitions may appear between an opening curly brace and its matching close, including directive blocks that also use matched curly braces to contain a definition.

Each realm definition is in the following form:

```
realm <identifier> {
  realm_source UNIXSYS | ISP | ISPADMIN | HTPASSWD
  [ realm_dir <data_directory> ]
  administrator {
    [user <realm_user_name>[ <realm_user_name>...]]
    [ group <realm_group_name>[ <realm_group_name>...] ]
  }
}
```

The syntax and definition of each directive are explained in the following *Directives* section.

Directives

The following keyword directives are valid in the `realms.conf` file:

`administrator { admins }`	Defines the realm users and groups that have permission to modify realm data. The *admins* directive may have a `user` directive or have a `group` directive. If neither user nor group is specified, then the site administrator becomes the default administrator.
`group` *group_name*`[` *group_name*`...]`	Names groups of users that have permission to modify realm data. The *group_name* directive is an optional directive valid in the `administrator` block. Separate multiple group names with white space.
`isp_component` *component_id_version*	Defines the component identification, the version of Solaris ISP Server, and the Administrator realm (`ISPADMIN`). The default value is "SUNWhttp–2.1".

`realm` *identifier* { *definition* }

Defines a realm. There may be multiple realm definitions in the `realms.conf` file, as long as each has a unique *identifier*. The *identifier* directive can be any arbitrary string of alphanumeric data (no special characters). White space is allowed when enclosed in double quotes.

The *definition* consists of realm directives, and must include at least a `realm_source`.

[`realm_dir` *data_path***]**

Defines a directory relative to the site path where the `users` and `groups` files for an `HTPASSWD` realm are stored. `realm_dir` is required and valid only if `realm_source` is `HTPASSWD`. It can be either an absolute path or a path relative to realms.conf, or it can be left unspecified. The default value is `realms/`*realmname*`/`.

`realm_source` *source*

Defines the source of user and group information for the realm. This directive is required in every realm definition. `realm_source` may be one of the following:

`HTPASSWD` Indicates that the user or group information is retrieved using the Sun WebServer users/group file format, and that user and group information will be maintained in the data directory named by `realm_dir`. The **`htrealm`**(1m) utility is used to

	create users and modify passwords.
`ISP`	Indicates that the user or group is stored in the Solaris ISP Server shared directory service. Changes to user and group information cannot be made through Sun WebServer.
`ISPADMIN`	Indicates that the principals are administrators in the Solaris ISP Server Sun™ Internet Administrator™. The –d flag takes the ISP-component ID and version (for example, “SUNWftp–2.0”).
`UNIXSYS`	Indicates that the operating system user and group definitions will be used to authenticate users in the realm. Sun WebServer employs a standard Pluggable Authentication Module (PAM) for authentication. Changes to user and group information cannot be made through Sun WebServer.

`user` *user_name*`[` *user_name*`...]`	Names realm users that have permission to modify realm data. The *user_name* can be specified in the `administrator` block. Separate multiple user names with white space.

EXAMPLES

EXAMPLE 1

/etc/pam.conf

```
httpd auth sufficient /usr/lib/security/pam_unix.so.1
httpd-isp auth sufficient /opt/SUNWisp/lib/pam_ldap.so.1 autohost
```

This configures Sun WebServer httpd to use the UNIX PAM library for authenticating for Solaris ISP Server subscribers stored in the LDAP-based directory. This does not use stacking, but uses different service names (httpd, httpd-isp).

EXAMPLE 2

Sample `realms.conf` file:

```
realm siteAdmin {
        realm_source HTPASSWD
        administrator {
                user user1
         }
}
realm SystemUsers {
        realm_source UNIXSYS

}
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO

`htIntro`(4), **`htrealm`**(1m)

NOTES Sun WebServer on Solaris 2.6 and greater uses a Pluggable Authentication Module (PAM) for authenticating principals in UNIXSYS and ISP realms using `/usr/lib/security/pam_unix.so`, and `/usr/lib/security/pam_ldap.so`, respectively. Refer to **`pam.conf`**(4) for details on how to set up PAM.

NAME

servlets.properties – Defines the servlet properties file in a general Java™ properties file format.

SYNOPSIS

site_path`/conf/servlets.properties`

DESCRIPTION

This file contains the name of each servlet and the initialization parameters of the servlet. You can either place the properties file in the default location, or specify the name of the `servlets.properties` file on start-up using the command `htservlet` and specifying –`i` *hostname* for the host name and –`p` *properties_file* for the name of the properties file.

The servlet.properties file may be updated using the Sun™ WebServer™ GUI or the **`htservlet`**(1m) utility.

You can specify the following servlet properties definitions:

- Code
- Code base
- Initial arguments

Syntax

The definitions in the `servlets.properties` file have the following format:

```
servlet.<servlet_name>.code=<servletclass>
servlet.<servlet_name>.codebase=<URL>
servlet.<servlet_name>.initArgs=<name>=<value>[,<name>=value...]
```

The following keyword directives are valid in the `servlet.properties` file:

servlet.<servlet_name>.code=<servletclass>

Names the name of the servlet main class file.

servlet.<servlet_name>.codebase=<URL>

Names the jar file or the URL of the servlet's codebase. Used only for remote servlets.

servlet.<servlet_name>.initArgs=<name>=<value>[,<name>=value...]

Defines the optional initial arguments passed to the servlet. Used in the format *name=value* [, *name=value...*]

EXAMPLES

EXAMPLE 1

A sample `servlets.properties` based on the default file:

```
#
# Copyright (c) 1998, by Sun Microsystems, Inc.
# All rights reserved.
#
# servlets.properties
#
#       Sample SWS Servlets Properties File
#
################################################

# the list of servlets to be loaded at startup time
servlets.startup=simple hello


# Date Servlet -- JSDK Sample Servlet
servlet.date.code=DateServlet

# Finger Servlet -- JSDK Sample Servlet
servlet.finger.code=FingerServlet

# Hello World  Servlet -- JSDK Sample Servlet
servlet.hello.code=HelloWorldServlet

# Snoop Servlet -- JSDK Sample Servlet
servlet.snoop.code=SnoopServlet
servlet.snoop.initArgs= port=23, host=localhost

# Simple Servlet -- JSDK Sample Servlet
servlet.simple.code=SimpleServlet
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO

htIntro(4), **htservlet**(1m), **httpd.conf**(4)

NAME

users – Defines the users in an HTPASSWD realm

SYNOPSIS

site_path/conf/realms/*HTPASSWD_realm*/users

DESCRIPTION

The users file lists the users in an HTPASSWD realm.

A user definition consists of a user name and password. You may add new users to the users file by **htrealm**(1m)or by the Sun WebServer GUI. To change a user's password, use **htpasswd**(1m) or the Sun WebServer GUI.

The password is displayed with BASE64 encoding. If you have installed FrontPage Apache Emulator, this field may be in UNIX encrypted format.

Syntax

The following syntax rules apply to the users file:

- The pound sign (#) is a comment character. All characters from a # to the end of a line are ignored
- White space is ignored in user definitions.

Each user definition is in the following form:

username:encoded_password

EXAMPLES

EXAMPLE 1

Sample users file:

```
# the user name and password database
 user1:YmFycnk=
 user2:YWRtaW4=
 user3:YHUvwaAx=
 user4:YTkoiWAt=
```

ATTRIBUTES

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

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWhttpc
Interface Stability	Evolving

SEE ALSO

htIntro(4), **htpasswd**(1m), **htrealm**(1m), **realms.conf**(4)

NOTES Do not edit `users` manually. Use **`htrealm`**(1m) to add or delete users, and **`htpasswd`**(1m) to set or edit passwords.

Index

A

access control for on-campus NNTP sites, 254
access.conf file, 319
accessing directory services, 36
    ispGetLdapInfo, 38
    ispGetLdapServers, 41
    ispGetTopDn, 43
    ispldap, 12
    IspLdapService, 45
active configuration file, 220
Active file
    renumbering, 195
add a virtual anonymous FTP server — ftpaddhost, 82
Administration tasks
    daily, 195
alias URLs for Sun WebServer web sites, 382
archive command, 149
archiving
    articles, 149
Article expiration
    control file, 225
article-batching backend, 151
articles
    archiving, 149
Articles
    removing, 195
authenticating administrators, 49

B

batcher command, 151
batchfiles
    rewrite, 200
buffchan command, 154
buffered file-writing backend, 154
bulk load directory entries, 29

C

CGI and Servlet log files, 362
change Sun WebServer user passwords, 288
check inn configuration and database files, 177
commands
    archive, 149
    batcher, 151
    buffchan, 154
    crosspost, 157
    ctlinnd, 159
    cvtbatch, 166
    expire, 167
    expireover, 170
    expirerm, 172
    fastrm, 173
    filechan, 175
    inncheck, 177
    innstat, 181
    innwatch, 182
    innxbatch, 183
    isppammod, 188
    makeactive, 190
    makehistory, 192
    news.daily, 195
    newsrequeue, 200
    nntpget, 202
    nntpsend, 203

overchan, 206
prunehistory, 207
rnews, 209
scanlog, 211
snsd, 212
snsnews, 213
summary of, 145
tally.control, 214
tally.unwanted, 215
writelog, 216
commands, innxmit, 185
Configuration data for InterNet News programs, 232
Configuration files, 235
active, 220
configuration files
checking, 177
Configuration files
control.ctl, 222
distrib.pats, 224
expire.ctl, 225
history, 228
hosts.nntp, 230
inn.conf, 232
moderators, 239
newsfeeds, 242
newslog, 251
nnrp.access, 254
nntpsend.ctl, 256
overview, 257
passwd.nntp, 258
sns.conf, 259
summary, 218
configuration for Sun WebServer security realms, 386
configuration for Sun WebServer web sites, 365
Configure
PAM for News Service LDAP authentication, 188
configure anonymous FTP — ftpconfig, 84
configuring Sun WebServer servers, 334
Connections
reader, feeder, 212
content negotiation settings for HTTP 1.1, 324
content.conf file, 324
control file for Usenet article expiration, 225
Control messages
specify handling of, 222
control.ctl configuration file, 222
controls Usenet supervision using innwatch, 235
convert
Usenet batch file to INN format, 166
create link
cross-posted articles, 157
Create newsgroup
tracking, 214
create Sun WebServer access control lists, 267
create Sun WebServer security realms, 291
create Sun WebServer server instances, 297
create URL aliases, 284
creating HTPASSWD realm groups in Sun WebServer, 329
cross-posted articles
create link, 157
crosspost command, 157
ctlinnd command, 159
cvtbatch command, 166

D

Daily administration tasks, 195
database files
checking, 177
define Sun WebServer access control lists, 319
defining servlets for Sun WebServer, 392
Delete newsgroup
tracking, 214
directory information API, 36
distrib.pats configuration file, 224

E

email addresses for moderated newsgroups, 239
error logging for Sun WebServer scripts and servlets, 362
Expiration
control file, 225
expire
news overview database entries, 170
usenet article and history files, 167
expire command, 167
expire.ctl configuration file, 225
expired articles, removing, 172

expireover command, 170
expirerm command, 172

F

fastrm command, 173
Feeder Daemon
    configuration files, checking, 177
file-writing backend for News, 175
filechan command, 175
FTP server logfile — xferlog, 116
FTP Virtual Server list — ftpservers, 114
ftpaccess — ftpd configuration file, 99
ftpaddhost — add a virtual anonymous FTP server, 82
ftpconfig — configure anonymous FTP, 84
ftpconversions — ftpd conversions database, 111
ftpcount — show current number of users for each class, 86
ftpd — Internet File Transfer Protocol server, 89
ftpd configuration file — ftpaccess, 99
ftpd conversions database — ftpconversions, 111
ftpd individual user host access file — ftphosts, 113
ftphosts — ftpd individual user host access file, 113
ftpintro.1m — introduction to the host configuration software command-line utilities for the Sun™ Internet FTP Server™., 81
ftpintro.4 — Introduction to the host configuration files for the Sun™ Internet FTP Server™., 98
ftpservers — FTP Virtual Server list, 114
ftpshut — shut down the ftp servers at a given time, 87

G

groups file, 329

H

hcjump, 5
hclfmd, 7
hclfmd.conf, 54
hcstartup, 9
history configuration file, 228
History database
    recovering, 192
history files
    expiring, 167
host configuration
    hcjump, 5
    hclfmd, 7
    hclfmd.conf, 54
    hcstartup, 9
hosts.nntp configuration file, 230
htaccess command, 267
htcontent command, 273
hthost command, 280
htIntro file, 263, 315
htmap command, 284
htpasswd command, 288
htrealm command, 291
htserver command, 297
htservlet command, 302
httpd event logs, 354
httpd init.d script, 313
httpd request logs, 357
httpd-instances.conf file, 331
httpd.conf file, 334

I

IDIA, 36
in.ftpd — Internet File Transfer Protocol server, 89
INN log files
    scan and summarize, 211
INN system, print snapshot, 181
inn.conf configuration files, 232
inncheck command, 177
innd
    check configuration and database files, 177
    controlling, 159
    database files, checking, 177
INND
    feeder connections, 212
innd
    monitoring, 182

start news feeder daemon, 213
stop news feeder daemon, 213
throttling, 182
innstat command, 181
innwatch command, 182
innwatch configuration file, 235
innxbatch command, 183
innxmit command, 185
installation
hcjump, 5
hcstartup, 9
instances list for Sun WebServer servers, 331
Internet File Transfer Protocol server — ftpd, 89
InterNet News programs
configuration data, 232
introduction to man pages of Sun WebServer configuration and log files, 315
introduction to Sun WebServer command-line utilities, 263
Introduction to the host configuration files for the Sun™ Internet FTP Server™. — ftpintro.4, 98
introduction to the host configuration software command-line utilities for the Sun™ Internet FTP Server™. — ftpintro.1m, 81
ispGetLdapInfo, 38
ispGetLdapServers, 41
ispGetTopDn, 43
ispldap, 12
IspLdapService, 45
ISPMC_aar, 49
isppammod command, 188
isprshp, 15

L

LDAP authentication
configure PAM for News Service, 188
list of hosts that feed NNTP news., 230
list of sites to feed via nntpsend, 256
log configuration file
snsd, 251
log file management
hclfmd, 7
hclfmd.conf, 54
Log files
rotating, 195
scan and summarize, 211
log files for Sun WebServer, 357
logfiles
add entry to, 216

M

makeactive command, 190
makehistory command, 192
manage Sun WebServer Java servlets, 302
manage Sun WebServer web sites, 280
map.conf URL alias configuration file, 382
mcadd, 124
mcaddadm, 125
mcadmpwd, 127
mcdsclean, 129
mcdsinit, 131
mchelp, 16
mchostls, 133
mcreg, 18
mcrm, 135
mcrmadm, 137
mcunreg, 27
Moderated newsgroups
moderator email addresses, 239
moderators configuration file, 239
modify HTTP 1.1 content meta-data, 273
monitoring
innd, 182

N

News Feeder Daemon, *see* snsnews(1m), snsd(1m), ctlinnd(1m),
news overview database
format specification, 257
update, 206
news overview database entries, expiring, 170
news-recovery , *see* active(4), ctlinnd(1m), dbz(3), filechan(1m), history(4), newsfeeds(4), makehistory(1m), makeactive(1m), newsrequeue(1m), snsnews(1m),

news.daily command, 195
newsfeed
read messages from a UUCP connection, 209
Newsfeed articles
distribution specifications, 242
newsfeeds configuration file, 242
newsgroup
track creation, 214
track deletion, 214
newsgroup creation
tracking, 214
newsgroup deletion
tracking, 214
newsgroups
list of active groups, 220
Newsgroups
moderators email addresses, 239
newsgroups
track unwanted, 215
newslog configuration file, 251
newsrequeue command, 200
nnrp.access configuration file, 254
NNTP
incoming connections, 212
NNTP news
list of feed hosts, 230
NNTP process
configuring, 259
NNTP server
retrieve articles from remote, 202
send articles to a remote, 203
sending xbatched Usenet articles to, 183
Usenet articles, sending to a remote, 190
NNTP sites
access control, 254
nntpget command, 202
nntpsend, 203
nntpsend command, 203
nntpsend.ctl configuration file, 256

O

overview configuration file, 257

P

PAM
configure for News Service LDAP authentication, 188
passwd.nntp configuration file, 258
Passwords
connecting to remote NNTP servers, 258
passwords for connecting to remote NNTP servers, 258
port numbers
setting for remote command daemon, 15
print snapshot of INN system, 181
prunehistory command, 207

R

Recover
batchfiles, 200
recover
Usenet history database, 192
redirect Sun WebServer URLs, 284
remote command daemon
setting the port number, 15
Remove
articles, 195
remove
file names from Usent history file, 207
remove a set of files quickly, 173
remove expired articles, 172
rewrite batchfiles, 200
rnews command, 209

S

scanlog command, 211
security realm configuration files, 386
send Usenet articles to a remote NNTP server, 185
send Usenet articles to remote NNTP site, 203
servlets properties, 392
show current number of users for each class — ftpcount, 86
shut down the ftp servers at a given time — ftpshut, 87
sisploa d, 29
SNS
configuration data, 259
sns.conf configuration files, 259
snsd

log configuration file, 251
snsd command, 212
snsnews command, 213
specify default values for Usenet distribution header, 224
specify handling of Usenet control messages, 222
specify where Usenet newsfeed articles get sent, 242
start
news feeder daemon, 213
start Sun WebServer instances, 313
Status report
creating, 195
stop
news feeder daemon, 213
Summary
configuration files, 218
Summary of commands, 145
Sun Internet Administrator
add administrator, 125
add service, 124
delete service, 135
help, 16
initial entries, 131
list components, 133
list services, 133
manage service, 124
password, 127
register component, 18
register service, 18
remove administrator, 137
remove service, 135
set administrator password, 127
unregister component, 27
unregister service, 27
version, 16
Sun News Server configuration data, 259
syslog logging in Sun WebServer, 354

T

tally.control command, 214
tally.unwanted command, 215
the record of current and recently expired Usenet articles, 228
throttling, innd, 182
track of newsgroup creation and deletions, 214

U

uninstall-sisp1.0, 33
uninstalling 1.0 components, 33
update
news overview database, 206
Usenet active newsgroups list, 220
Usenet article
expiring, 167
Usenet article expiration control file, 225
Usenet articles
get from remote NNTP server, 202
record of current and recently expired, 228
retrieve from remote NNTP server, 202
send to a remote NNTP server, 190
send to remote site, 203
send xbatched to a remote NNTP server, 183
Usenet articles, send to a remote NNTP server, 185
Usenet batch file to INN format, converting, 166
Usenet control messages
specify handling of, 222
Usenet distribution header
specify default values, 224
Usenet history database
recovering, 192
Usenet newsfeed articles
distribution specifications, 242
Usent history file
remove file names from, 207
users file, 394
users in HTPASSWD realms for Sun WebServer, 394
UUCP connection
read messages from, 209

V

validate
innd configuration and database files, 177

W

web site configuration files, 365
writelog commands, 216

X

xferlog — FTP server logfile, 116