Updated for 8.2.2

Data ONTAP® 8.2

Commands: Manual Page Reference

For 7-Mode, Volume 2

NetApp, Inc.
495 East Java Drive
Sunnyvale, CA 94089
U.S.

Telephone: +1 (408) 822-6000
Fax: +1 (408) 822-4501
Support telephone: +1 (888) 463-8277
Web: www.netapp.com
Feedback: doccomments@netapp.com

Part number: 215-08873_B0
June 2014

Table of Contents

Legal Information

Copyright

Trademarks

About the Data ONTAP Commands: Manual Page Reference, Volume 2

The *Commands: Manual Page Reference* document is a compilation of all the manual (man) pages for Data ONTAP commands, special files, file formats and conventions, and system management and services. It is provided in two volumes, each of which includes a complete index of all man pages in both volumes.

The *Commands: Manual Page Reference* includes only admin level commands. Advanced commands are not included in the *Commands: Manual Page Reference* because they should only be used under NetApp Global Support guidance.

Manual pages are grouped into sections according to standard UNIX naming conventions and are listed alphabetically within each section. The following tables list the types of information for which Data ONTAP provides manual pages and the reference volume in which they can be found.

Contents of Volume 1

Manual page section	Section titles	Information related to
1	Commands	Storage system administration

Contents of Volume 2

Manual page section	Section titles	Information related to
4	Special Files	Formatting of media
5	File Formats and Conventions	Configuration files and directories
8	System Management and Services	Protocols, service daemons, and system management tools

Manual pages can be displayed at the storage system command line.

Terminology

Storage systems that run Data ONTAP are sometimes also referred to as *filers*, *appliances*, *storage appliances*, or *systems*.

The na prefix for manual page names

All Data ONTAP manual pages are stored on the storage system in files whose names are prefixed with the string "na_" to distinguish them from client manual pages. The prefixed names are used to refer to storage system manual pages from other manual pages and sometimes appear in the NAME field of the manual page, but the prefixes do not need to be part of commands.

Viewing manual pages at the command line

To view a manual page for a command at your storage system command line (console), enter the following:

man *command*

Note: Data ONTAP commands are case sensitive.

To see a list of all commands from the storage system command line, enter a question mark (?) after the host prompt.

Manual pages about using manual pages

Useful manual pages about using manual pages are the help(1) and the man(1) manual pages. You can use the `man help` command to view information about how to display the manual page for a particular command. You can use the `man man` command to view information about how to use the `man` command.

Manual Pages by Section in This Volume and Complete Index of Both Volumes

Manual Pages By Section

Section 4: Special Files

Using device files such as tape.

[Section 1 | **Section 4** | Section 5 | Section 8 | Complete Index]

tape	Information on the tape interface

Section 5: File Formats and Conventions

Formats for human-readable configuration files, such as those found in **`/etc`** on the root volume.

[Section 1 | Section 4 | **Section 5** | Section 8 | Complete Index]

auditlog	contains an audit record of recent administrative activity
backuplog	captures significant events during file system or volume backup and recovery activities.
boot	Directory of Data ONTAP executables
cifs_homedir.cfg	configuration file for CIFS home directories
cifs_nbalias.cfg	configuration file for CIFS NetBIOS aliases
clone	Log of clone activities
cloned_tapes	List of non-qualified tape drives attached to the node
crash	Directory of system core files
dgateways	Default gateways list
dumpdates	Data base of filesystem dump times
exports	A list of export entries for all file system paths that Data ONTAP exports automatically when NFS starts up.
fsecurity	Definition file for an fsecurity job
ftpusers	file listing users to be disallowed ftp login privileges
group	group file
hosts	Host name data base
hosts.equiv	List of hosts and users with rsh permission
httpd.access	authentication controls for HTTP access
httpd.group	Names of HTTP access groups and their members
httpd.hostprefixes	configuration of HTTP root directories for virtual hosts
httpd.log	Log of HTTP
httpd.mimetypes	map of file suffixes to MIME ContentType

httpd.passwd	file of passwords required for HTTP access
httpd.translations	URL translations to be applied to incoming HTTP requests
messages	record of recent console messages
netgroup	Network groups data base
networks	Network name data base
nsswitch.conf	Configuration file for name service switch
passwd	Password file
psk.txt	psk.txt is disabled in this release of Data ONTAP.
qual_devices	Table of qualified disk and tape devices
quotas	quota description file
rc	system initialization command script
registry	registry database
resolv.conf	configuration file for domain name system resolver
rmtab	Remote mounted file system table
serialnum	System serial number file
services	Internet services
shadow	shadow password file
sis	Log of Advanced Single Instance Storage (SIS) activities
sm	network status monitor directory
snapmirror	Log of SnapMirror Activity
snapmirror.allow	List of allowed destination nodes
snapmirror.conf	volume and qtree replication schedules and configurations
stats_preset	stats preset file format
symlink.translations	Symbolic link translations to be applied to CIFS path lookups
syslog.conf	syslogd configuration file
tape_config	Directory of tape drive configuration files
treecompare	Log of treecompare activities
usermap.cfg	mappings between UNIX and Windows NT accounts and users
zoneinfo	time zone information files

Section 8: System Management and Services

Protocols and service daemons, such as **rshd** and **snmpd**, and system management tools, such as **autosupport** and **syslogd**.

[Section 1 | Section 4 | Section 5 | **Section 8** | Complete Index]

cifs	Common Internet File System (CIFS) Protocol
cli	Data ONTAP command language interperter (CLI)
dns	Domain Name System
http	HyperText Transfer Protocol
nfs	Network File System (NFS) Protocol
nis	NIS client service
pcnfsd	(PC)NFS authentication request server
protocolaccess	Describes protocol access control
rlmaccess	Describes SSH access control to the RLM.
rmt	Remote magtape protocol module
rquotad	remote quota server
rshd	remote shell daemon
snmpd	snmp agent daemon
spaccess	Describes SSH access control to the SP.
syslogd	Logs system messages.

Man Page Complete Index

acpadmin (1)	Commands for managing Alternate Control Path Administrator
aggr (1)	Commands for managing aggregates, displaying aggregate status, and copying aggregates
arp (1)	Address resolution display and control
auditlog (5)	contains an audit record of recent administrative activity
autosupport (1)	notification daemon
autosupport_destinations (1)	Displays a summary of the current AutoSupport destinations.
autosupport_history (1)	Displays and retransmits recent AutoSupports.
autosupport_manifest (1)	Displays AutoSupport content manifest.
autosupport_trigger (1)	Displays and modifies AutoSupport trigger configuration.
awa (1)	Control the Automated Workload Analyzer (AWA)
backup (1)	manages backups
backuplog (5)	captures significant events during file system or volume backup and recovery activities.
bmc (1)	Commands for use with a Baseboard Management Controller (BMC)
boot (5)	Directory of Data ONTAP executables
bootfs (1)	Boots file system accessor command (ADVANCED).
cdpd (1)	Views the neighbors of the storage controller that are discovered using Cisco Discovery Protocol(CDP) v1 and associated statistics.
cf (1)	Controls the takeover and giveback operations of the nodes in a High Availability (HA) pair.
charmap (1)	command for managing per-volume character maps

file (1)	Manages individual files.
flexcache (1)	commands for administering FlexCache volumes
floppyboot (1)	Describes the menu choices at the Boot Menu prompt.
fpolicy (1)	configure file policies
fsecurity (1)	Summary of fsecurity commands
fsecurity (5)	Definition file for an fsecurity job
fsecurity_apply (1)	Creates a security job based on a definition file and applies it to the file system.
fsecurity_cancel (1)	Cancels outstanding fsecurity jobs
fsecurity_help (1)	Displays a description and usage information for fsecurity commands
fsecurity_remove-guard (1)	Removes the Storage-Level Access Guard from a volume or qtree
fsecurity_show (1)	Displays the security settings on files and directories
fsecurity_status (1)	Displays the status of outstanding fsecurity jobs
ftp (1)	Displays FTP statistics.
ftpd (1)	File transfer protocol daemon
ftpusers (5)	file listing users to be disallowed ftp login privileges
group (5)	group file
halt (1)	Stops the node.
help (1)	Prints summary of commands and help strings.
hostname (1)	Sets or displays node name.
hosts (5)	Host name data base
hosts.equiv (5)	List of hosts and users with rsh permission
http (8)	HyperText Transfer Protocol
httpd.access (5)	authentication controls for HTTP access
httpd.group (5)	Names of HTTP access groups and their members
httpd.hostprefixes (5)	configuration of HTTP root directories for virtual hosts
httpd.log (5)	Log of HTTP
httpd.mimetypes (5)	map of file suffixes to MIME ContentType
httpd.passwd (5)	file of passwords required for HTTP access
httpd.translations (5)	URL translations to be applied to incoming HTTP requests
httpstat (1)	display HTTP statistics
ic (1)	High Availability (HA) interconnect command
if_addr_filter_info (1)	displays the MAC address filter usage information for the specified network interface
ifconfig (1)	Configures network interface parameters.
ifgrp (1)	Manages interface group (ifgrp) configuration.
ifinfo (1)	Displays driver-level statistics for network interfaces.
ifstat (1)	Displays device-level statistics for network interfaces.
igroup (1)	Commands for managing initiator groups

portset (1)	Commands for managing portsets
priority (1)	Commands for managing priority resources
priv (1)	Controls per-connection privilege settings.
protocolaccess (8)	Describes protocol access control
psk.txt (5)	psk.txt is disabled in this release of Data ONTAP.
qtree (1)	Creates and manages qtrees.
qual_devices (5)	Table of qualified disk and tape devices
quota (1)	Controls node disk quotas.
quotas (5)	quota description file
radius (1)	Manages RADIUS client protocol and components.
rc (5)	system initialization command script
rdate (1)	Sets system date from a remote host.
rdfile (1)	Reads a WAFL file.
reallocate (1)	Command for managing reallocation of files, LUNs, volumes, and aggregates
reboot (1)	Stops and then restarts the node.
registry (5)	registry database
resolv.conf (5)	configuration file for domain name system resolver
restore (1)	file system restore
restore_backup (1)	Commands for restoring backup through network in an HA pair configuration.
revert_to (1)	Reverts file system to a previous release.
rlm (1)	Commands for use with a Remote LAN Module (RLM)
rlmaccess (8)	Describes SSH access control to the RLM.
rmt (8)	Remote magtape protocol module
rmtab (5)	Remote mounted file system table
route (1)	Manually manipulates the routing table.
routed (1)	Network RIP and router discovery routing daemon
rquotad (8)	remote quota server
rshd (8)	remote shell daemon
rshstat (1)	Prints information about active rsh sessions.
rtsold (1)	Router solicitation daemon
san (1)	Glossary for NetApp specific SAN terms
sasadmin (1)	Commands for managing Serial Attached SCSI (SAS) adapters
sasstat (1)	Commands for managing Serial Attached SCSI (SAS) adapters
savecore (1)	Saves a core dump.
sectrace (1)	Manages permission tracing filters.
secureadmin (1)	Command for secure administration of the appliance
serialnum (5)	System serial number file
services (5)	Internet services

tape_config (5)	Directory of tape drive configuration files
timezone (1)	Sets and obtains the local timezone.
traceroute (1)	Prints the route that packets take to network host.
traceroute6 (1)	print the route IPv6 packets take to a network node
treecompare (5)	Log of treecompare activities
ucadmin (1)	Commands for managing Fibre Channel and converged networking adapters
uptime (1)	Shows how long the system has been up.
useradmin (1)	Administers node access controls.
usermap.cfg (5)	mappings between UNIX and Windows NT accounts and users
version (1)	Displays Data ONTAP version.
vfiler (1)	Commands for vfiler operations
vif (1)	DEPRECATED, please use the na_ifgrp(1) command instead.
vlan (1)	Manages VLAN interface configuration.
vmservices (1)	services for Data ONTAP running in a virtual machine
vol (1)	Commands for managing volumes, displaying volume status, moving volumes, and copying volumes.
vscan (1)	Control virus scanning for files on the node.
wafl_aac (1)	displays allocation area cache/statistics information of the specified volume.
wcc (1)	Manages WAFL credential cache.
wrfile (1)	Writes a WAFL file.
ypcat (1)	Prints values from a NIS database.
ypgroup (1)	Displays the group file entries cached locally from the NIS server if NIS is enabled.
ypmatch (1)	Prints matching values from a NIS database.
ypwhich (1)	Displays the NIS server if NIS is enabled.
zoneinfo (5)	time zone information files

tape

NAME

na_tape - Information on the tape interface

DESCRIPTION

The Data ONTAP system supports up to 64 local tape drives (tape drives connected directly to the system). The tape drive interface follows a UNIX-like device name allowing use of a **rewind**, **norewind** or **unload/reload** device. The device name can be the classic *cstnd* format, or of the format *c.name.d* where:

c
: describes the rewind/unload characteristic of the device. Use **r** to specify the **rewind** device, use **nr** to specify the **norewind** device, or use **ur** to specify the **unload/reload** device. The **norewind** device will not rewind when the tape device is closed. The **unload/reload** device is used with sequential tape loaders and will unload the current tape volume and attempt to load the next tape volume (note that the server will wait up to one minute for the next volume to become ready before aborting the reload of the next volume). The rewind device will rewind the tape volume to beginning-of-tape on close.

st
: the **st** portion of the device name is always present in the classic format, and is one of the options in the *name* format. It specifies that you are requesting a SCSI tape device.

n
: the alias number (in decimal) of the tape drive to use. The **st** and *n* parameters together - **stn** constitute a tape "alias". See the **storage alias** command for information about tape aliases and device addresses.

d
: the density (or format) to use for tape write operations. Consists of one of the four letters **l** (low), **m** (medium), **h** (high) or **a** (advanced).

name
: specifies a tape alias, an electrical name or a serial number(SN) corresponding to the tape device. The electrical name and SN formats can contain an optional parameter for the device SCSI logical unit. This parameter is expressed as **Llun.** See the **storage alias** command for further information about the format of the *name* parameter.

Each tape device is automatically associated with an alias. If an alias assignment does not already exist at the first discovery of a tape device, the system will create an alias for it. FC and SAS devices receive SN aliases, and SCSI devices receive electrical aliases by default. The alias will remain associated with the SN or electrical name -- even through boot -- until the alias is changed.

The **storage alias** and **storage unalias** commands allow the user to view existing aliases and delete aliases respectively. The **storage alias** command also allows the user to change an existing alias name or to assign the alias prior to a tape device being discovered. You can assign the tape alias by specifying the alias name with the electrical name or serial number with the **storage alias** command.

EXAMPLES

The density specifications for an Exabyte 8505 8mm drive:

```
l  Exabyte 8200 format, no compression
m  Exabyte 8200 format with compression
h  Exabyte 8500 format, no compression
a  Exabyte 8500 format with compression
```

Examples of tape drive names:

```
nrst0l
nr.st0.l
r.9a.1L1.a
ur.switch1:5.h
nr.SN[HU104514FJ].m
```

The **sysconfig -t** command displays the tape drives on your system, the device alias associated with each tape device, and the device's available density settings. The following is an example of the output from a sysconfig command on a system with one tape device attached:

toaster> **sysconfig -t**

```
Tape drive (0.6)  Exabyte 8505 8mm
rst0l  -    rewind device,         format is: EXB-8200   2.5GB
nrst0l -    no rewind device,      format is: EXB-8200   2.5GB
urst0l -    unload/reload device,  format is: EXB-8200   2.5GB
rst0m  -    rewind device,         format is: EXB-8200C  (w/compression)
nrst0m -    no rewind device,      format is: EXB-8200C  (w/compression)
urst0m -    unload/reload device,  format is: EXB-8200C  (w/compression)
rst0h  -    rewind device,         format is: EXB-8500   5.0GB
nrst0h -    no rewind device,      format is: EXB-8500   5.0GB
urst0h -    unload/reload device,  format is: EXB-8500   5.0GB
rst0a  -    rewind device,         format is: EXB-8500C  (w/compression)
nrst0a -    no rewind device,      format is: EXB-8500C  (w/compression)
urst0a -    unload/reload device,  format is: EXB-8500C  (w/compression)
```

The **storage show tape** command shows the electrical name, WWN and serial number associated with the device and the corresponding alias:

toaster> **storage show tape**

```
Tape Drive:          0.6
Description:         Exabyte 8505 8mm
Serial Number:       IE71E024
World Wide Name:
Alias Name(s):       st0
Device State:        available
```

SEE ALSO

na_sysconfig(1)

auditlog

NAME

na_auditlog - contains an audit record of recent administrative activity

SYNOPSIS

<logdir>/auditlog

<logdir> is **/etc/log** for nodes and **/logs** for NetCache appliances.

DESCRIPTION

If the option **auditlog.enable** is on, the system logs all input to the system at the console/telnet shell and via rsh to the auditlog file. The data output by commands executed in this fashion is also logged to auditlog. Administrative servlet invocations (via HTTP, typically from FilerView) and API calls made via the ONTAPI interface are also logged to the auditlog. A typical message is:

Wed Feb 9 17:34:09 GMT [rshd_0:auditlog]: root:OUT:date: Wed Feb 9 17:34:09 GMT 2000

This indicates that there was an rsh session around Wed Feb 9 17:34:09 GMT which caused the **date** command to be executed. The user performing the command was root. The type of log is data output by the system as indicated by the **OUT** keyword.

Commands typed at the node's console or executed by rsh are designated by the **IN** keyword as in:

Wed Feb 9 17:34:03 GMT [rshd_0:auditlog]: :IN:rsh shell: RSH INPUT COMMAND is date

The start and end of an rsh session are specially demarcated as in

Wed Feb 9 17:34:09 GMT [rshd_0:auditlog]: root:START:rsh shell:orbit.eng.mycompany.com

and

Wed Feb 9 17:34:09 GMT [rshd_0:auditlog]: root:END:rsh shell:

The maximum size of the auditlog file is controlled by the **auditlog.max_file_size** option. If the file gets to this size, it is rotated (see below).

Every Saturday at 24:00, **<logdir>/auditlog** is moved to **<logdir>/auditlog.0, <logdir>/auditlog.0** is moved to **<logdir>/auditlog.1**, and so on. This process is called rotation. Auditlog files are saved for a total of six weeks, if they do not overflow.

If you want to forward audit log messages to a remote syslog log host (one that accepts syslog messages via the BSD Syslog protocol specified in RFC 3164), modify the node's /etc/syslog.conf file to forward messages from the node's "local7" facility to the remote host. Do this by adding a line like:

local7.*
    @1.2.3.4

to /etc/syslog.conf. An IP address has been used here, but a valid DNS name could also be used. Note that using a DNS name can fail if the node is unable to resolve the name given in the file. If that happens, your messages will not be forwarded.

On the log host, you'll need to modify the syslog daemon's configuration file to redirect syslog message traffic from the "local7" facility to the appropriate configuration file. That is typically done by adding a line similar to the one shown above for the node:

local7.*
    /var/logs/filer_auditlogs

Then restart the daemon on the log host, or send an appropriate signal to it. See the documentation for your log host's syslog daemon for more information on how to make that configuration change.

FILES

<logdir>/auditlog
    auditlog file for current week. **<logdir>/auditlog.[0-5]** auditlog files for previous weeks

SEE ALSO

na_syslog.conf(5)

backuplog

NAME

na_backuplog - captures significant events during file system or volume backup and recovery activities.

SYNOPSIS

/etc/log/backup

DESCRIPTION

Node captures significant dump/restore/smtape-related events and the respective times at which they occur. All events are recorded in one-line messages in **/etc/log/backup.** Dump/restore events are described first, followed by smtape events.

The following are the dump/restore events node monitors:

Start
Dump/restore starts.

Restart
Restart of a dump/restore.

End
Dump/restore completes successfully.

Abort
The operation aborts.

Error
Dump/restore hits an unexpected event.

Options
Logs the options as users specify.

Tape_open Output device is opened successfully.

Tape_close Output device is closed successfully.

Phase_change
As dump/restore completes a stage.

Dump specific events:

Snapshot
When the snapshot is created or located.

Base_dump
When a valid base dump entry is located.

Logging events for dump/restore:

Start_logging Logging begins.

Stop_logging
Logging ends.

Each dump/restore event record is in the following format:

TYPE TIME_STAMP IDENTIFIER EVENT (EVENT_INFO)

TYPE
Either dmp(dump), rst(restore) or log events.

TIME_STAMP
Shows date and time at which event occurs.

IDENTIFIER
Unique ID for the dump/restore.

EVENT
The event name.

EVENT_INFO
Event specific information.

A typical dump/restore event record message looks like:

dmp Thu Apr 5 18:54:56 PDT 2001 /vol/vol0/*home(5)* **Start (level 0, NDMP)**

In the particular example:

TYPE
= **dmp**

TIME_STAMP
= **Thu Apr 5 18:54:56 PDT 2001**

IDENTIFER
= /vol/vol0/*home(5)*

EVENT
= **Start**

EVENT_INFO
= **level 0, NDMP**

The following is the smtape manager event:

MGR-Start
smtape manager starts.

Events specific for smtape backup:

BKP-Start
Backup job starts.

BKP-Abort
Backup job aborts.

BKP-End
Backup job ends.

BKP-Tape-Chg Backup job is waiting for tape change.

BKP-Continue Backup job continues after tape change.

BKP-Params
Starting parameters for backup job.

BKP-DW-Start Data warehouse starts for backup job.

BKP-DW-End
Data warehouse ends for backup job.

BKP-Warning
Warning from backup job.

BKP-Tape-Stats Tape statistics for backup job.

Events specific for smtape restore:

RST-Start
Restore job starts.

RST-Abort
Restore job aborts.

RST-End
Restore job ends.

RST-Tape-Chg Restore job is waiting for tape change.

RST-Continue Restore job continues after tape change.

RST-Params
Starting parameters for restore job.

Events related to smtape operations from CLI:

CLI-Backup
smtape backup command.

CLI-Restore
smtape restore command.

CLI-Abort
smtape abort command.

CLI-Continue
smtape continue command.

Each smtape event record has the following format:

JOB_ID TIME_STAMP VOL_PATH EVENT (EVENT_INFO)

JOB_ID
Unique number for a backup/restore job. Manager and CLI events do not have job IDs.

TIME_STAMP
Shows date and time at which event occurs.

VOL_PATH
Volume path associated with a backup/restore job.

EVENT
Event name.

EVENT_INFO
Event specific information.

A typical smtape event record message looks like:

5 Tue Nov 4 19:48:56 GMT /vol/vol1 BKP-Start (level 0 backup of Backup Set ID c6872e62-ac3b-11dd-ab3c-00a0980868f8)

In the particular example:

JOB_ID
= **5**

TIME_STAMP
= **Tue Nov 4 19:48:56 GMT**

VOL_PATH
= **/vol/vol1**

EVENT
= **BKP-Start**

EVENT_INFO
= **level 0 backup of Backup Set ID c6872e62-ac3b-11dd-ab3c-00a0980868f8**

All event messages go to **/etc/log/backup.** On every Sunday at 00:00, **backup** is rotated to **backup.0** and **backup.0** is moved to **backup.1** and so on. Up to 6 log files(spanning up to 6 weeks) are kept.

The registry option **backup.log.enable** controls the enabling and disabling of the logging with values **on** and **off** respectively. The functionality is enabled by default. (See na_options(1) for how to set options.)

FILES

/etc/log/backup
backup log file for current week. **/etc/log/backup.[0-5]** backup log files for previous weeks

SEE ALSO

na_restore(1)

boot

NAME

na_boot - Directory of Data ONTAP executables

SYNOPSIS

/etc/boot

DESCRIPTION

The **boot** directory contains copies of the executable files required to boot the node. The **download** command (see na_download(1)) copies these files from **/etc/boot** into the node's boot block, from which the system boots.

FILES

/etc/boot
Directory of Data ONTAP executables. Files are placed in /etc/boot after the tar or setup.exe has decompressed them. These files vary from release to release.

SEE ALSO

na_download(1)

cifs_homedir.cfg

NAME

na_cifs_homedir.cfg - configuration file for CIFS home directories

SYNOPSIS

/etc/cifs_homedir.cfg

DESCRIPTION

The configuration file **/etc/cifs_homedir.cfg** is used to configure home directory paths for users which access the node using the CIFS network protocol.

EXAMPLE

This is a sample **/etc/cifs_homedir.cfg** file with one CIFS home directory path. The node will look for a CIFS home directory for user "Bill" by appending the user's name to the path. From the example below, the node will provide user "Bill" a CIFS home directory at /vol/userVol/users/Bill if that directory exists.

```
#
# This file contains the path(s) used by the node to determine if a
# CIFS user has a home directory. See the System Administrator's Guide
# for a full description of this file and a full description of the
# CIFS homedir feature.
#
# There is a limit to the number of paths that may be specified.
# Currently that limit is 1000.
# Paths must be entered one per line.
#
# After editing this file, use the console command "cifs homedir load"
# to make the node process the entries in this file.
#
# Note that the "#" character is valid in a CIFS directory name.
# Therefore the "#" character is only treated as a comment in this
# file if it is in the first column.
#
# Two example path entries are given below.
# /vol/vol0/users1
# /vol/vol1/users2
#
# Actual path entries follow this line.
/vol/userVol/users
```

EFFECTIVE

Any changes take effect after running the 'cifs homedir load' command.

PERSISTENCE

Changes are persistent across system reboots.

FILES

/etc/cifs_homedir.cfg

SEE ALSO

na_cifs_homedir(1)

cifs_nbalias.cfg

NAME

na_cifs_nbalias.cfg - configuration file for CIFS NetBIOS aliases

SYNOPSIS

/etc/cifs_nbalias.cfg

DESCRIPTION

The configuration file **/etc/cifs_nbalias.cfg** is used to configure NetBIOS aliases for the node. A NetBIOS alias allows the node to be accessed by a CIFS client using an alternate name for the node.

EXAMPLE

This is a sample **/etc/cifs_nbalias.cfg** file with one NetBIOS alias.

```
#
# This file contains NetBIOS aliases used by the node.
# See the System Administrator's Guide for a full
# description of this file.
#
# There is a limit to the number of aliases that may be specified.
# Currently that limit is 200.
#
# Aliases must be entered one per line.
#
# After editing this file, use the console command "cifs nbalias load"
# to make the node process the entries in this file.
#
# Note that the "#" character is valid in a CIFS NetBIOS alias.
# Therefore the "#" character is only treated as a comment in this
# file if it is in the first column.
#
# Actual NetBIOS alias name(s) for the node follow this line.
NODEALIAS01
```

EFFECTIVE

Any changes take effect once CIFS services are restarted

PERSISTENCE

Changes are persistent across system reboots.

FILES

/etc/cifs_nbalias.cfg

SEE ALSO

na_cifs_nbalias(1)

clone

NAME

na_clone - Log of clone activities

SYNOPSIS

/etc/log/clone

DESCRIPTION

The **clone** log file contains a log of clone activities for the node. The file lives in **/etc/log** on the root volume.

Every Sunday at midnight, **/etc/log/clone** is moved to **/etc/log/clone.0**; **/etc/log/clone.0** is moved to **/etc/log/clone.1**; and so on. The suffix can go up to 5, so the old **/etc/log/clone.5** will be deleted. Clone activities are saved for a total of seven weeks.

Each entry of the **/etc/log/clone** file is a single line containing the following space-separated fields.

```
timestamp Volume:vol-name event_info
```

The following is a description of each field.

timestamp
Displayed in **ctime()** format, e.g. Fri Jul 17 20:41:09 GMT 2008. Indicates the time this event was recorded.

vol-name
The volume name on which clone operation is performed:

event_info
The event which is being logged. These are the current event types with their operation info:

Clone Start ID:<clone-id> Clone File:<clone-file> Source File:<source-file> Total Blocks:<total-blocks>

Corresponds to "clone start" command.

Clone End ID:<clone-id> Clone File:<clone-file> Source File:<source-file> <end-reason> Total Blocks:<totalblocks> Blocks Copied: <blocks-copied>

Corresponds to clone operation has been completed successfully, unsuccessfully or stopped by user.

Clone Stop ID:<clone-id> Clone File:<clone-file> Source File:<source-file>

Corresponds to "clone stop" command.

Clone Restart Successful/Failed ID:<clone-id> Clone File:<clone-file> Source File:<source-file> Total Blocks:<total-blocks>

Corresponds to "clone restart" operation.

Clone Boot <info>

Corresponds to reboot and provides the information about clone boot work on the volume, whether it is completed successfully or failed with error.

EXAMPLE

A clone operation started with source file as f1 and clone file as f1_1. then the clone operation was stopped by user. The clone log file should have the following entries:

```
Tue Oct 21 09:03:18 GMT 2008 Volume: vol1 [sid: 0] Clone Start ID: 1, Clone File: f1_1, Source File: f1, Total Blocks: 786432
Tue Oct 21 09:03:24 GMT 2008 Volume: vol1 [sid: 0] Clone Stop ID: 1, Clone File: f1_1, Source File: f1
Tue Oct 21 09:03:26 GMT 2008 Volume: vol1 [sid: 0] Clone End ID: 1, Clone File: f1_1, Source File: f1 (Clone operation aborted by user), Total Blocks: 786432,Blocks Copied: 0
Tue Oct 21 09:11:17 GMT 2008 Volume: vol2 [sid: 0] Clone Restart Successful. ID: 2, Clone File: f2, Source File: f2_1, Total Blocks: 50
```

FILES

/etc/log/clone
Clone log file for current week.

/etc/log/clone.[0-5]
Clone log files for previous weeks.

SEE ALSO

`na_clone(1)`

cloned_tapes

NAME

na_cloned_tapes - List of non-qualified tape drives attached to the node

SYNOPSIS

/etc/cloned_tapes

DESCRIPTION

If you attach a tape drive that NetApp Inc has not tested with the node, enter information about the tape drive in the **/etc/cloned_tapes** file. This file enables the node to register the drive as a clone of a qualified drive.

If the node boots with a nonqualified tape drive and the **/etc/cloned_tapes** file does not exist, the node creates a sample file, when the first "mt" command for the tape is executed.

Each entry in the **/etc/cloned_tapes** file corresponds to one tape drive. Specify the entry in one of the following formats:

clone_vendor_id clone_product_id EMULATES *vendor_id product_id*

clone_product_id EMULATES *product_id*

The "storage show tape supported" command provides a list the product_id and vendor_id values of qualified drives.

EXAMPLE

The following entry in the **/etc/cloned_tapes** file enables the node to register the Quantum DLT9000 tape drive, which has not been tested with the node, as a clone of the Quantum DLT7000 tape drive:

QUANTUM DLT9000 EMULATES QUANTUM DLT7000

SEE ALSO

na_storage(1)

crash

NAME

na_crash - Directory of system core files

SYNOPSIS

/etc/crash

DESCRIPTION

If a node crashes, it creates a core file in the **crash** directory. The core files are very useful for finding and fixing bugs in Data ONTAP, so please notify NetApp Global Services of any core files on your node.

See na_savecore(1) for more details about how core files are saved.

FILES

/etc/crash/core.*
saved core files **/etc/crash/core.*-small** compact core file.

/etc/crash/bounds
suffix for next core file

/etc/crash/minfree
free KB in FS to maintain after savecore

SEE ALSO

na_savecore(1)

dgateways

NAME

na_dgateways - Default gateways list

SYNOPSIS

/etc/dgateways

DESCRIPTION

The use of **/etc/dgateways** file has been deprecated. Either add a static default gateway in **/etc/rc** or enable router discovery in **routed** to discover multiple default gateways.

The **/etc/dgateways** file is used by the old **routed** command to construct a set of potential default gateways. The file comprises a series of lines, each in the following format:

gateway metric

gateway is the name or address of a gateway to be used as a potential default gateway.

metric is a metric indicating the preference weighting of the gateway. 1 is the value to use for highest preference, 15 for the least. If no value is specified, *metric* will default to the value 1.

There can be a maximum of 128 valid entries in the **/etc/dgateways** file - additional ones will be ignored, with an error message being displayed. Duplicate gateway names or addresses are not allowed - only the first one encountered in the file will be added by **routed** to the default gateway table, and the additional ones will produce error messages.

EXAMPLE

Here are typical lines from the **/etc/dgateways** file:

```
main_router    1
backup_router  2
```

SEE ALSO

na_rc(5),

NOTES

The use of **/etc/dgateways** file has been deprecated.

dumpdates

NAME

na_dumpdates - Data base of filesystem dump times

SYNOPSIS

/etc/dumpdates

DESCRIPTION

The **dump** command (see na_dump(1)) uses **/etc/dumpdates** to keep track of which subtrees have been dumped and when. Each line in **dumpdates** contains the subtree dumped, the dump level, and the creation date of the snapshot used by **dump**. There is only one entry per subtree at a given dump level. **dumpdates** may be edited to change any of the fields, if necessary.

EXAMPLE

This shows the dumpdate file for a system on which **/home** and **/export** are backed up using **dump**.

```
/home      0 Tue  Nov  2  10:56:27     1993
/export    0 Tue  Nov  2  13:51:17     1993
/export    1 Tue  Nov  5  18:31:17     1993
/home      1 Tue  Nov  5  18:45:27     1993
```

FILES

/etc/dumpdates

SEE ALSO

na_dump(1)

exports

NAME

na_exports - A list of export entries for all file system paths that Data ONTAP exports automatically when NFS starts up.

SYNOPSIS

/etc/exports

DESCRIPTION

The **/etc/exports** file contains a list of export entries for all file system paths that Data ONTAP exports automatically when NFS starts up. The **/etc/exports** file can contain up to 10,240 export entries. Each export entry can contain up to 4,096 characters, including the end-of-line character. To specify that an export entry continues onto the next line, you must use the line continuation character "\".

An export entry has the following syntax:

```
path -option[,option...]
```

where *path* is a file system path (for example, a path to a volume, directory, or file) and *option* is one of the following export options:

actual=*path*
Specifies the actual file system path corresponding to the exported file system path. You can use this option to move files to new locations without requiring NFS clients to mount new file system paths. The actual file system path you specify must exist. You cannot specify an exported file system path that consists of a single forward slash (/), which would mislead some automounters. Note: NFSv4 clients will not see an exported path using the **actual** option unless the export path is only one level deep and is not **/vol**.

anon=*uid*|*name*
Specifies the effective user ID (or name) of all anonymous or root NFS client users that access the file system path. An anonymous NFS client user is an NFS client user that does not provide valid NFS credentials; a root NFS client user is an NFS client user with a user ID of 0. Data ONTAP determines a user's file access permissions by checking the user's effective user ID against the NFS server's **/etc/passwd** file. By default, the effective user ID of all anonymous and root NFS client users is 65534. To disable root access by anonymous and root NFS client users, set the **anon** option to 65535. To grant root user access to all anonymous and root NFS client users, set the **anon** option to 0.

nosuid
Disables creation of **setuid** and **setgid** executable files and **mknod** commands on the file system path. Unless the file system is a root partition of a diskless NFS client, you should set the **nosuid** option to prevent NFS client users from creating **setuid** executable files and device nodes that careless or cooperating NFS server users could use to gain root access. Pre-existing **setuid** and **setgid** executable files will continue to function as intended.

ro | **ro**=*clientid*[:*clientid*...]
Specifies which NFS clients have read-only access to the file system path. To give all NFS clients read-only access, specify the **ro** option. Otherwise, specify the **ro=** option followed by a colon-delimited list of NFS client identifiers. To exclude NFS clients from the list, prepend the NFS client identifiers with a minus sign (-). Unless you specify the **ro**, **ro=**, or **rw=** option, Data ONTAP uses the **rw** option, giving all NFS clients read-write access to the file system path.

rw | **rw**=*clientid*[:*clientid*...]
Specifies which NFS clients have read-write access to the file system path. To give all NFS clients read-write access, specify the **rw** option. Otherwise, specify the **rw=** option followed by a colon-delimited list of NFS client identifiers. To exclude NFS clients from the list, prepend the NFS client identifiers with a minus sign (-). Unless you specify the **ro**, **ro=**, or **rw=** option, Data ONTAP uses the **rw** option, giving all NFS clients read-write access to the file system path. Note: Unlike in Data ONTAP releases prior to 6.5, if you specify the **rw=** option, Data ONTAP does not use the **ro** option as the default for all other NFS clients.

root=*clientid*[:*clientid*...]
Specifies which NFS clients have root access to the file system path. If you specify the **root=** option, you must specify at least one NFS client identifier. To exclude NFS clients from the list, prepend the NFS client identifiers with a minus sign (-).

sec=*sectype*[:*sectype*...]
Specifies the security types that an NFS client must support to access the file system path. To apply the security types to all types of access, specify the **sec=** option once. To apply the security types to specific types of access (anonymous, non-super user, read-only, read-write, or root), specify the **sec=** option at least twice, once before each access type to which it applies (**anon**, **nosuid**, **ro**, **rw**, or **root**, respectively). Note: You cannot apply the same security type to more than one access type. By default, an NFS client must support the **sys** security type to access a file system path.

Specify any combination of the following security types as a colon-delimited list:

none
No security. Data ONTAP treats all of the NFS client's users as anonymous users.

sys
Standard UNIX (AUTH_SYS) authentication. Data ONTAP checks the NFS credentials of all of the NFS client's users, applying the file access permissions specified for those users in the NFS server's **/etc/passwd** file. This is the default security type.

krb5
Kerberos(tm) Version 5 authentication. Data ONTAP uses data encryption standard (DES) key encryption to authenticate the NFS client's users.

krb5i
Kerberos(tm) Version 5 integrity. In addition to authenticating the NFS client's users, Data ONTAP uses message authentication codes (MACs) to verify the integrity of the NFS client's remote procedure requests and responses, thus preventing "man-in-the-middle" tampering.

krb5p
Kerberos(tm) Version 5 privacy. In addition to authenticating the NFS client's users and verifying data integrity, Data ONTAP encrypts NFS arguments and results to provide privacy.

Note: Before specifying the **krb5**, **krb5i**, or **krb5p** option, you must enable Kerberos V5 security using the **nfs setup** command. For more information, see na_nfs(1)**.**

Specifying an NFS client identifier

To specify which NFS clients have read-only, read-write, and root access to a file system path (using the **ro=**, **rw=**, and **root=** options, respectively), you must specify an NFS client identifier. An NFS client identifier is a host name, netgroup name, IP address, subnet, or DNS domain.

A host name is an alphanumeric string associated with an IP address. Data ONTAP uses the first definition that it finds in the **/etc/hosts** file, searching the NIS, LDAP, DNS, and local versions in the order specified in the **/etc/nsswitch.conf** file.

A netgroup name is an alphanumeric string associated with a group of host names. Data ONTAP uses the first definition that it finds in the **/etc/netgroup** file, searching the NIS, DNS, and local versions in the order specified in the **/etc/nsswitch.conf** file. Note: DNS does not support netgroups.

To specify that a name is a netgroup name, not a host name, thus preventing Data ONTAP from searching the **/etc/hosts** file unnecessarily, prepend the name with an "at" (@) character.

To specify that all netgroup names begin with an "at" (@) character, thus preventing Data ONTAP from searching the **/etc/hosts** or **/etc/netgroups** file unnecessarily, set the **nfs.netgroup.strict** option to **on**. For more information, see na_options(1)**.**

Note: If a name is defined as both a host name and a netgroup name, Data ONTAP assumes the name is a host name.

An IP address uniquely identifies a machine on an IP network. For IPv4, a machine IP is in dotted decimal format (AAA.BBB.CCC.DDD), and for IPv6, machine IP is of the form [AAAA:BBBB:CCCC:DDDD::FFFF]. For example:

```
104.342.403.224 (IPv4)
BA32:235C:5D24:23F::32 (IPv6)
```

A subnet is a group of machines that share a common network. To specify a subnet, use the following short form:

```
subnetaddr/subnetbits
```

where *subnetaddr* is the subnet IP address and *subnetbits* is the number of bits in the subnet mask.

You can also use the following long form, but Data ONTAP automatically converts this long form to the short form:

```
[networkaddr] subnetaddr [subnetmask] subnetmask
```

where *networkaddr* is the network IP address, *subnetaddr* is the subnet IP address, and *subnetmask* is the subnet mask.

A DNS domain is an alphanumeric value starting with a period (.) that identifies a group of machines. For example:

```
.frogs.fauna.mycompany.com
```

EXTENDED DESCRIPTION

To edit the **/etc/exports** file, you must either use a text editor on an NFS client that has root access to the storage system or run the **exportfs** command with the **-b**, **-p**, or **-z** option on the storage system command line.

Enabling automatic updating
If the **nfs.export.auto-update** option is **on**, Data ONTAP updates the **/etc/exports** file automatically when you create, rename, or destroy a volume. In this case, when you create a volume, if an administration host is defined, Data ONTAP adds the following export entry to the **/etc/exports** file:

```
path -sec=sys,root=adminhostid,nosuid
```

If an administration host is not defined, Data ONTAP adds the following entry to the **/etc/exports** file:

```
path -sec=sys,rw,nosuid
```

When you rename a volume, Data ONTAP automatically replaces the old volume name, wherever it appears in **/etc/exports** file, with the new volume name. When you delete a volume, Data ONTAP removes all corresponding entries from the **/etc/exports** file.

If the **nfs.export.auto-update** option is **off**, Data ONTAP does not update the **/etc/exports** file automatically when you create, rename, or destroy a volume; instead, it adds a message to the system log that notifies you to update the **/etc/exports** file manually.

Specifying ro, ro=, rw, and rw=
The following sections describe how to specify the **ro**, **ro=**, **rw**, and **rw=** options given their defaults, invalid combinations, and order of precedence.

Defaults:

* If you do not specify the **ro**, **ro=**, or **rw=** option, Data ONTAP uses the **rw** option by default.

* Unlike in Data ONTAP releases prior to 6.5, if you specify a list of NFS clients with read-write access using the **rw=** option, Data ONTAP does not use the **ro** option as the default for all other NFS clients.

Invalid combinations:

* You cannot specify the **ro** option with the **ro=** option.

* You cannot specify the **rw** option with the **rw=** option.

* You cannot exclude an NFS client identifier from the **ro=** or **rw=** option and include the same NFS client identifier in the other option.

Order of precedence:

* The **ro** option takes precedence over the **rw** option.

* The **ro=** option takes precedence over the **rw** option.

* The **rw=** option takes precedence over the **ro** option.

* The **ro=** option takes precedence over the **rw=** option.

* A host name or IP address in the **ro=** or **rw** option takes precedence over a netgroup, subnet, or domain in the other option.

* Host names and IP addresses take precedence from left to right within an option.

Upgrading the /etc/exports file
Whenever you invoke the **exportfs** command to export file systems specified in the **/etc/exports** file (for example, whenever you invoke **exportfs -a** or **exportfs -r**), Data ONTAP automatically upgrades the **/etc/exports** file to a format compatible with the current Data ONTAP release.

Data ONTAP no longer supports the **access** option; therefore, Data ONTAP automatically converts all export entries containing an **access** option to an equivalent export entry containing the **ro=** or **rw=** option.

For example, if an export entry uses the **access** option to specify that an NFS client has read-write access:

```
/vol/vol0 -access=hostname
```

Data ONTAP upgrades the export entry to use the **rw=** option instead:

```
/vol/vol0 -rw=hostname
```

Note: Unlike in Data ONTAP releases prior to 6.5, if you specify the **rw=** option, Data ONTAP does not use the **ro** option as the default for all other NFS clients.

Similarly, if an export entry uses the **access** option to specify that an NFS client has read-only access:

```
/vol/vol0 -access=hostname,ro
```

Data ONTAP upgrades the export entry to use the **ro=** option instead:

```
/vol/vol0 -ro=hostname
```

In addition, if an export entry specifies subnets in long form:

```
/vol/vol0 -rw="network 10.45.67.0 netmask 255.255.255.0"
```

Data ONTAP upgrades them to short form:

```
/vol/vol0 -rw=10.45.67.0/24
```

Note: Data ONTAP always preserves the ordering of NFS client identifiers within an option. Also, upgrading has no effect on the **root=**, **rw=**, and **ro=** options because their formatting has not changed.

Upgrade examples
Old:

```
/vol/vol0 -anon=0
```

New:

```
/vol/vol0 -rw,anon=0
```

Old:

```
/vol/vol0 -access=pets:workers:alligator:mule,rw=dog:cat:skunk:pig:horse:ox:mule
```

New:

```
/vol/vol0 -ro=pets:workers:alligator,rw=dog:cat:skunk:pig:horse:ox:mule
```

This can be rewritten as:

```
/vol/vol1 -ro=pets:workers:alligator,rw=pets:workers
```

And should be:

```
/vol/vol1 -ro=alligator,rw=@pets:@workers
```

Reverting the /etc/exports file
To revert the **/etc/exports** file to a format compatible with the Data ONTAP 6.5 or 6.4 release, run the **exportfs -d 6.5** command or **exportfs -d 6.4** command, respectively.

When you run the **exportfs -d 6.5** command, Data ONTAP:

* Removes all "at" (@) symbols, which denote netgroups.

* Consolidates multiple security contexts into one security context. If the **ro** and/or **rw** options exist in any security context, Data ONTAP removes the **ro=** and **rw=** options, respectively, from the other security contexts. Data ONTAP merges security contexts from left to right.

When you run the **exportfs -d 6.4** command, Data ONTAP:

* Reverts the **/etc/exports** file to a format compatible with the Data ONTAP 6.5 release (see above).

* Replaces **anon**=*clientid* with **anon**=*uid*.

* Removes **nosuid**.

* Removes all domain names, each of which starts with a period (.).

* Removes all excluded NFS client identifiers, each of which starts with a minus sign (-).

* Removes the **rw** option.

* Replaces **rw**=*clientid*,**ro** with **rw**=*clientid*.

* Replaces **rw**=*clientidX*,**ro**=*clientidY*

with **access**=*clientidX*+*clientidY*,**rw**=*clientidX*.

* Removes **ro**=*clientid*,**rw**.

Note: This access restriction cannot be expressed in a format that is compatible with the Data ONTAP 6.4 release.

* Replaces **ro**=*clientid* with **access**=*clientid*,**ro**.

* Replaces **rw**=*clientid* with **access**=*clientid*,**rw**=*clientid*.

Note: After running the **exportfs -d 6.4** command, you must manually edit all **rw=** and **root=** options in the **/etc/exports** file to:

* Replace netgroup names with the host names.

* Reduce the number of host names to less than 255.

* Reduce the number of characters to 4,096 or less.

When reverting the **/etc/exports** file, Data ONTAP displays messages on the console notifying you of any export entries that require manual editing.

Managing duplicate entries
Data ONTAP processes export entries in sequential order, using only the last export entry in the **/etc/exports file** for a specific file system path. Therefore, you should not add multiple export entries for the same file system path, whether exported or actual, to the **/etc/exports** file.

For example, if you add the following export entries to the **/etc/exports** file:

```
/vol/vol0/ -ro
/vol/vol0/ -rw
```

Data ONTAP exports **/vol/vol0** to all NFS clients for read-write access.

And, if you add the following export entries to the **/etc/exports** file:

```
/vol/vol1/ -actual=/vol/vol0,ro
/vol/vol2/ -actual=/vol/vol0,rw
```

Data ONTAP exports **/vol/vol2/** to all NFS clients for read-write access, mapping it internally to **/vol/vol0**. Data ONTAP does not export **/vol/vol1/**.

Debugging mount and access problems
For information about debugging mount and access problems, see na_exportfs(1)**.**

EXAMPLES

For the following examples, assume the **/etc/netgroup** file contains the following entries:

```
farm pets livestock workers
pets (dog,,) (cat,,) (pig,,) (parrot,,)
livestock (cow,,) (pig,,) (chicken,,) (ostrich,,)
workers (dog,,) (horse,,) (ox,,) (mule,,)
predators (coyote,,) (puma,,) (fox,,) (crow,,)
```

Read and write access: netgroups
The following example exports **/vol/vol0** to **horse** for read-write access:

```
/vol/vol0 -anon=0,rw=horse
```

Note: Unlike in Data ONTAP releases prior to 6.5, all other NFS clients do not get read-only access.

The following example exports **/vol/vol0** to **horse** for read-write access and all other NFS clients for read-only access:

```
/vol/vol0 -anon=0,ro,rw=horse
```

Each of the following examples exports **/vol/vol0** to **workers** (**dog**, **cat**, **pig**, and **parrot**) for read-only access and all remaining farm animals for read-write access:

```
/vol/vol0 -ro=@workers,rw=@farm
/vol/vol0 -rw=@farm,ro=@workers
```

The following example exports **/vol/vol0** to all NFS clients except **workers** for read-write access:

```
/vol/vol0 -rw=@farm:-@workers
```

Note: The **workers** do not have any access at all.

The following example exports **/vol/vol0** to **pets** for read-write access and **livestock** for read-only access, but denies access to **workers**:

```
/vol/vol0 -rw=@pets:-@workers,ro=@livestock
```

Read and write access: subnets
The following example exports /vol/vol0 to all NFS clients in the 10.56/16 subnet for read-write access and all NFS clients in the 10.56.17/24 subnet for read-only access:

```
/vol/vol0 -ro=10.56.17/24,rw=10.56/16
```

The following example exports /vol/vol0 to all NFS clients in the subnet A1C0:4C34:5D32:6F34::1/64 for read-only access and all NFS clients whose IPv6 address is BA32:235C:5D24:23F::32 for read-write access.

```
/vol/vol0 -ro=[A1C0:4C34:5D32:6F34::1]/64,rw=[BA32:235C:5D24:23F::32]
```

The following example exports **/vol/vol0** to 10.56.17.5 and 10.56.17.6 for read-write access and to all remaining NFS clients in the 10.56.17/24 subnet for read-only access:

```
/vol/vol0 -ro=10.56.17/24,rw=10.56.17.5:10.56.17.6
```

Read and write access: domains
The following example exports **/vol/vol0** to all NFS clients in the **.frogs.fauna.mycompany.com** domain for read-only access and to all remaining clients in the **.fauna.mycompany.com** domain for read-write access:

```
/vol/vol0 -ro=.frogs.fauna.mycompany.com, rw=.fauna.mycompany.com
```

Excluding NFS client identifiers
Data ONTAP gives precedence to NFS client identifiers from left to right within an access control list; therefore, if you exclude an NFS client identifier from a list, the order in which you specify netgroups, subnets, and domains becomes important if the same NFS client appears in more than one netgroup, subnet, or domain.

For example, suppose **cat**, which belongs to the **farm** and **pets** netgroups, requests read-write access to **/vol/vol0**.

Data ONTAP grants **cat** read-write access if you specify the following export entry:

```
/vol/vol0 -ro,rw=@farm:-@pets
```

But Data ONTAP denies **cat** read-write access if you specify the following export entry in which the order of the netgroups in the **rw=** list is reversed:

```
/vol/vol0 -ro,rw=-@pets:@farm
```

In the first example, Data ONTAP gives precedence to the **farm** netgroup, which is included in the read-write access list. In the second example, Data ONTAP gives precedence to the **pets** netgroup, which is excluded from the read-write access list.

Specifying an actual path
The following example exports **/vol/vol0/home/user1** as **/vol/vol0/user1** to NFSv2/v3 clients for read-write access:

```
/vol/vol0/user1 -actual=/vol/vol0/home/user1,sec=sys,rw
```

The following example exports **/vol/vol0/home** as **/myhome** to NFSv2/v3/v4 clients for read-write access:

```
/myhome -actual=/vol/vol0/home,sec=sys,rw
```

Controlling anonymous access
The following example exports **/vol/vol0** to all NFS clients for read-write access, but prevents access by anonymous and root NFS client users:

```
/vol/vol0 -sec=sys,rw,anon=65535
```

The following example exports **/vol/vol0** to all NFS clients for read-write access, giving anonymous and root NFS client users an effective user ID of 100:

```
/vol/vol0 -sec=sys,rw,anon=100
```

The following example exports **/vol/vol0** to all NFS clients for read-write access, giving anonymous and root NFS client users an effective user ID of 0 (root):

```
/vol/vol0 -sec=sys,rw,anon=0
```

Controlling root access
The following example exports **/vol/vol0** to **adminhost** for root access and all other NFS clients for read-write access:

```
/vol/vol0 -sec=sys,rw,root=adminhost
```

The following example exports **/vol/vol0** to **adminhost** for root access and all other NFS clients for read-write access, but prevents **adminhost** from creating **setuid** executables and device nodes:

```
/vol/vol0 -sec=sys,rw,root=adminhost,nosuid
```

Controlling access by sectype
The following example exports **/vol/vol0** to all NFS clients supporting the **krb5** security type for read-write access and all remaining NFS clients in the **.farm.mycompany.com** domain for read-only access:

```
/vol/vol0 -ro=.farm.mycompany.com,sec=krb5,rw
```

The following example exports **/vol/vol0** to all hosts supporting no security type for read-write access and all hosts supporting the **krb5**, **krb5i**, or **krb5p** security type for read-write and root access:

```
/vol/vol0 -sec=sys:none,rw,sec=krb5:krb5i:k4b5p,rw,anon=0
```

FILES

/etc/hosts Maps IP addresses to host names and aliases.

/etc/netgroup Maps group names to hosts.

/etc/nsswitch.conf Specifies the order in which Data ONTAP searches local, NIS, DNS, and LDAP files.

/etc/passwd Specifies user information.

SEE ALSO

na_hosts(5)**,** na_passwd(5)

fsecurity

NAME

na_fsecurity - Definition file for an fsecurity job

DESCRIPTION

The fsecurity definition files describe an fsecurity job, which is used as input to the na_fsecurity_apply(1) command, and contains a list of tasks that will be run against the file system. This file can have any convenient name, and can be stored in any convenient location in the local volumes. The name of the file is given as a parameter to the na_fsecurity_apply(1) command.

SYNTAX

The definition file can be located anywhere in the file system, in either ASCII or Unicode format. The first line is always the file's signature, with task definitions on each subsequent line.

The file signature is currently *cb56f6f4*, and it will be updated when new versions of the file are supported. It is important that this is the only value on the line, including spaces.

Each task is a comma-separated list of values that are defined as follows:

```
type,subtype,"path",propagation mode,"security definition"
```

type
1 - Security Descriptor Definition Language (SDDL)

subtype
0 - Standard
1 - Storage-Level Access Guard (Guard)

path
The path to the target file system object, in double-quotes.

propagation mode
0 - Propagate inheritable permissions to all subfolders and files
1 - Do not allow permissions on this file or folders to be replaced (**Not implemented**)
2 - Replace existing permissions on all subfolders and files with inheritable permissions

security definition
The security definition that will be applied to the specified
path. The format is described by the **type** field, and is always enclosed in double-quotes.

For more information about SDDL syntax and proper formatting of the security description value, see "Security Descriptor String Format" at the following URL:
http://msdn2.microsoft.com/en-us/library/aa379567.aspx

NOTE This file can also be generated by the **secedit** utility. It is available for download from the NOW Tool Chest.

EXAMPLE

This is a sample fsecurity definition file which propagates a security descriptor down the /vol/vol0/qtree hierarchy. The definition allows Everyone full control, and the second line sets a Guard security descriptor which denies the ability to Write.

```
cb56f6f4
1,0,"/vol/vol0/qtree",0,"D:(A;CIOI;0x1f01ff;;;Everyone)"
1,1,"/vol/vol0/qtree",0,"D:(D;CIOI;0x000002;;;Everyone)"
```

EFFECTIVE

Any changes take effect after running the na_fsecurity_apply(1) command.

PERSISTENCE

Changes are persistent across system reboots.

SEE ALSO

na_fsecurity(1)

ftpusers

NAME

na_ftpusers - file listing users to be disallowed ftp login privileges

SYNOPSIS

/etc/ftpusers

DESCRIPTION

The /etc/ftpusers file is an ASCII file that lists users for whom ftp login privileges are disallowed. Each ftpuser entry is a single line of the form:

user_name

where user_name is the user's login name.

By default there is no /etc/ftpusers file, and therefore ftp login privileges are allowed to all users.

EFFECTIVE

Any changes take effect immediately

PERSISTENCE

Changes are persistent across system reboots.

group

NAME

na_group - group file

SYNOPSIS

/etc/group

DESCRIPTION

The **/etc/group** database contains information for each group in the following form:

groupname:password:gid:user-list

The following list describes the required fields:

groupname
: The name of the group.

password
: The group's password, in an encrypted form. This field may be empty.

gid
: An integer representing the group; each group is assigned a unique integer.

user-list
: The user list is a comma-separated list of users allowed in the group.

EXAMPLE

Here is a sample group file:

```
project:asderghuIoiyw:12:dan,dave
myproject::11:steve,jerry
```

SEE ALSO

na_quota(1), na_cifs_setup(1)

hosts

NAME

na_hosts - Host name data base

SYNOPSIS

/etc/hosts

DESCRIPTION

The **hosts** file contains information regarding the known hosts on the network. For each host an entry should be present with the following information:

Internet-address official-host-name aliases

When both IPv4 and IPv6 addresses are configured for a particular host, there will be a separate entry in the file for each address. Items are separated by any number of blanks and/or tab characters. A ‘‘#’’ indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. The maximum line length is 1022 characters. There is no way to continue an entry past the end of the line.

This file may be created from the official host data base maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown hosts.

IPv4 network addresses are specified in the conventional ‘‘.’’ (dot) notation. IPv6 addresses are specified in any of the conventional forms i.e., the colon delimited compressed form or the mixed IPv6 and IPv4 notation. Host names may contain any alphanumeric character, but not field delimiters, newline, or comment characters.

FILES

/etc/hosts

SEE ALSO

na_nis(8)

hosts.equiv

NAME

na_hosts.equiv - List of hosts and users with rsh permission

SYNOPSIS

/etc/hosts.equiv

DESCRIPTION

The **hosts.equiv** file contains a list of hosts on which you can enter a node command through the remote shell protocol (**rsh**).

Hosts specified in this file are considered the trusted hosts of the node.

It is also possible to use **hosts.equiv** for other protocols such as ssh (both interactive and non-interactive) and telnet. Additionally, access to ONTAPI (ONTAP management APIs) over HTTP and HTTPS can use **hosts.equiv** authentication by setting the node option httpd.admin.hostsequiv.enable.

Each line in **hosts.equiv** has the following format:

hostname [*username*]
+@netgroup [*username*]

If the host on which you enter the node command is a UNIX host, the user name is optional. If the host on which you enter the node command is a PC, you must enter the user name for that PC in the **/etc/hosts.equiv** file.

We can also specify a group of hosts using **netgroup**. Hence all hosts in that netgroup are allowed to access the node.

If you do not specify a user name for a UNIX host, you must be root on that host to execute a node command through **rsh**.

If multiple users on the same host should have access to the node through **rsh**, enter each user name on a separate line.

EXAMPLE

The following **hosts.equiv** file allows both **root** and **joe_smith** to enter node commands through **rsh** on a UNIX host named **adminhost**. It also allows **joe_smith** to enter node commands through **rsh** from all hosts in netgroup **ourhosts**:

adminhost
adminhost joe_smith
+@ourhosts joe_smith

SEE ALSO

na_options(1)

httpd.access

NAME

na_httpd.access - authentication controls for HTTP access

SYNOPSIS

/etc/httpd.access

DESCRIPTION

The HTTP daemon can apply authentication controls to individual users or groups on a per directory basis. The file **/etc/httpd.access** specifies the following items for each access-controlled tree:

the path to the tree

the authority required to authenticate access to the tree

the lists of users or groups who are permitted access when authenticated

The syntax is the same as the access control syntax used by NCSA and Apache. However, the **httpd.access** file only supports a subset of directives supported by NCSA and Apache. You can copy an existing NCSA or Apache access to the node without editing or reformatting.

SYNTAX

The supported directives are:
<Directory directory_name>
</Directory>
AuthName *Title phrase*
require user *user_id*[, *user_id*,...] **require group** *group_id*[, *group_id*,...]

where *Title phrase* is a word or phrase that is passed to the authentication dialog as a title for the dialog that prompts the user for a password.

EXAMPLES

The following example restricts access to the file **/home/htdocs/private/bob** so that only user dole can access it, after supplying the required password. The authentication dialog is titled ‘‘My private stuff.’’

<Directory /home/htdocs/private/bob> AuthName My private stuff
<Limit GET>
require user dole
</Limit>
</Directory>

The **<Limit GET>** and **</Limit>** directives are not supported, but are retained for format consistency with NCSA and Apache. The node just ignores them.

The following example restricts access to the directory tree **/home/htdocs/private/conspiracy** to the group ‘‘guyinblack’’, which consists of the users whose IDs are cancer, deepthroat, mrx, and skinner. The authentication dialog is titled ‘‘Area 51.’’

<Directory /home/htdocs/private/conspiracy> AuthName Area 51
<Limit GET>
require group guyinblack
</Limit GET>
</Directory>

In this example, ‘‘guyinblack’’ is defined by the following entry in **/etc/httpd.group**:

guyinblack: cancer deepthroat mrx skinner

The following example requires the client to provide a Windows Domain username and password to access the directory tree **/home/htdocs/win.** The authentication dialog is ‘‘Windows(tm) Authentication’’ This authentication dialog, typed exactly as presented here, is required to enforce NTLM authentication.

<Directory /home/htdocs/win>
AuthName Windows(tm) Authentication </Directory>

If this authentication control is used the Node must have

CIFS
running, and either be a member of a Windows Domain or be using Local User authentication.

EFFECTIVE

Any changes take effect within 5 minutes

PERSISTENCE

Changes are persistent across system reboots.

SEE ALSO

na_httpd.group(5).

BUGS

Only the directives listed above are supported; other directives that may appear in NCSA or Apache access files are ignored.

httpd.group

NAME

na_httpd.group - Names of HTTP access groups and their members

SYNOPSIS

/etc/httpd.group

DESCRIPTION

The file declares the names of groups and the user IDs of the members of each group, for use by the HTTP daemon in executing the access controls declared in **/etc/httpd.access**.

SYNTAX

group_id1:*user_id1* [*user_id2* ...]

EFFECTIVE

Any changes take effect within 5 minutes

PERSISTENCE

Changes are persistent across system reboots.

SEE ALSO

na_httpd.access(5).

httpd.hostprefixes

NAME

na_httpd.hostprefixes - configuration of HTTP root directories for virtual hosts

SYNOPSIS

/etc/httpd.hostprefixes

DESCRIPTION

The **httpd.hostprefixes** file maps virtual hosts used in HTTP to corresponding root directories. The same configuration file is used for both IP virtual hosts (defined by the IP address used for connecting to the server) and HTTP virtual hosts (defined by the **Host:** header used in HTTP requests).

Each virtual host has a corresponding subdirectory within the directory specified by the option **httpd.rootdir**. This subdirectory is called the virtual host root directory. Clients connected to a virtual host can only access files within the virtual host root directory.

In the **httpd.hostprefixes** file, each line consists of a virtual host root directory followed by the names and IP addresses of a virtual host. If you specify an IP address, the virtual host root directory is associated with the given virtual host for IP-level virtual hosting. If you specify a name, the virtual host root directory is associated with the virtual host with that name, using HTTP-level virtual hosting. If the node can resolve that name to an IP address, which is used for an IP-level host alias (see the **alias** option in na_ifconfig(1)), the node uses that IP address in the same way as it would if you specified the IP address in the **httpd.hostprefixes** file.

If the **/etc/httpd.hostprefixes** file is edited, it is read again by the HTTP server after the changes are saved.

SETUP

1. Enable **httpd.enable** and set HTTP Root directory **httpd.rootdir**

2. Configure network interface with HTTP Virtual Host Addresses. For example, to add the 207.68.156.50 as HTTP Virtual Host address to the network interface e0a, enter the following command:

```
toaster> ifconfig e0a alias 207.68.156.50
```

NOTE: In Data ONTAP 7.3 and later releases, VH interface is no longer supported for HTTP Virtual Hosting.

3. Edit /etc/httpd.hostprefixes file and map the Virtual Host addresses to respective subdirectories within the directory specified by the option **httpd.rootdir.** For example, to map the Virtual Host address 207.68.156.50 specified in Step 2 above to the **httpdir1** subdirectory within **httpd.rootdir,** add the following entry to the **/etc/httpd.hostprefixes** file:

```
/httpdir1 207.68.156.50
```

4. Test HTTP virtual host setup by sending HTTP request to the Virtual Host address added and mapped in Step 2 and 3 above.

EXAMPLE

This example maps requests sent to **www.customer1.com** to the **customer1** subdirectory of **httpd.rootdir** and requests directed at a host with IP address 207.68.156.58 to the subdirectory **customer2**.

```
/customer1 www.customer1.com
/customer2 207.68.156.58
```

If the command

```
toaster> ifconfig e0a alias www.customer1.com
```

had been issued before the configuration file was read, requests destined for the IP address of **www.customer1.com** would also be mapped to the **/customer1** subdirectory, regardless any the **Host:** header they included.

EFFECTIVE

Any changes take effect within 5 minutes

PERSISTENCE

Changes are persistent across system reboots.

SEE ALSO

na_options(1)

httpd.log

NAME

na_httpd.log - Log of HTTP

SYNOPSIS

/etc/log/httpd.log

DESCRIPTION

The HTTP server logs an entry for every file retrieved via HTTP. This log, written to **/etc/log/httpd.log**, is stored in the "Common Log Format," which is used by many World Wide Web servers.

Each entry in **/etc/log/httpd.log** consists of one line with seven fields. The fields are, in order:

address
The IP address of the HTTP client requesting the file.

rfc931
This field is always "-".

authuser
This field is always "-".

date
The time and date the request was is reported in the format "[Day/Mon/Year:HH:MM:SS]", which is logged in universal time (GMT) rather than the local time zone.

request
A quoted string is recorded for the method (request type) and file involved in the request.

result
The status code for the request, as defined in RFC 1945, the HTTP protocol specification. (See below.)

bytes
The size of the file in bytes.

Possible values for *result* codes include:

200
Success: the requested file was transmitted.

302 Redirected (see **/etc/httpd.translations**).

304
Not modified (client cache used).

400
Bad request.

401
Unauthorized request.

403
Access to file prohibited.

404
File not found.

503
HTTP server disabled.

The size of the log file can be restricted by the option **httpd.log.max_file_size**.

SEE ALSO

na_httpd.translations(5)
RFC 1945, "Hypertext Transfer Protocol -- HTTP/1.0"

BUGS

Some Web servers report size statistics differently for result codes other than 200. For example, a file size of 0 is often reported for result code 304 (Not modified).

The log file grows automatically and is never reset. It is your responsibility to rotate files and empty the log files regularly.

httpd.mimetypes

NAME

na_httpd.mimetypes - map of file suffixes to MIME ContentType

SYNOPSIS

/etc/httpd.mimetypes

DESCRIPTION

For HTTP/1.0 and higher protocols, a MIME header is returned in the reply of every GET request. This header includes a "Content-Type" field, whose contents is determined by examining the suffix of the file being transmitted.

The **/etc/httpd.mimetypes** file contains the mapping of filename suffixes to MIME Content-Type. The format of each line is: suffix, Content-Type. Comments are introduced with a "#".

The node is not shipped with the **/etc/httpd.mimetypes** file. Instead, the node's system files include a sample file named **/etc/httpd.mimetypes.sample**. Before you start using HTTP, make a copy of **/etc/httpd.mimetypes.sample** and name the copy **/etc/httpd.mimetypes**.

If the file **/etc/httpd.mimetypes** is not installed, the HTTP server looks for the file **/etc/httpd.mimetypes.sample** as a fallback.

EXAMPLE

 # map .ps files to PostScript type:
ps application/postscript

EFFECTIVE

Any changes take effect within 5 minutes

PERSISTENCE

Changes are persistent across system reboots.

httpd.passwd

NAME

na_httpd.passwd - file of passwords required for HTTP access

SYNOPSIS

/etc/httpd.passwd

DESCRIPTION

The password file containing the encrypted form of the password that an HTTP client must supply to have access to a file in a controlled-access directory tree, as declared in **/etc/httpd.access**.

The password is encrypted in the regular UNIX style. User of NCSA or Apache can use their **htpasswd** program to generate the user_id:passwd pair.

The HTTP access control does not use the existing CIFS password database on the node because in http basic authentication, in each request for protected pages, the value of *passwd* is sent over the network in clear text, and without encryption would compromise the user's password.

SYNTAX

user_id1:encrypted_passwd1
used_id2:encrypted_passwd2
...

EFFECTIVE

Any changes take effect within 5 minutes

PERSISTENCE

Changes are persistent across system reboots.

SEE ALSO

na_httpd.access(5).

httpd.translations

NAME

na_httpd.translations - URL translations to be applied to incoming HTTP requests

SYNOPSIS

/etc/httpd.translations

DESCRIPTION

The HTTP daemon supports four URL translation rules to filter incoming HTTP requests. The HTTP daemon applies each rule in succession, stopping at the first successful **Redirect**, **Pass**, or **Fail** rule:

Map *template result*
Any request which matches *template* is replaced with the *result* string given.

Redirect *template result*
Any request which matches *template* is redirected to the *result* URL. Note that this must be a full URL, e.g., beginning with "http:".

Pass *template* [*result*]
Any request which matches *template* is granted access, and no further rule processing occurs. An optional *result* can be used in place of the matching URL.

Fail *template*
Any request which matches *template* is denied access. Rule processing stops after a matched **Fail**.

Both templates and results may contain wildcards (a star "*" character). The wildcard behaves like a shell wildcard in the *template* string, matching zero or more characters, including the slash ("/") character. In the *result* string, a wildcard causes text from the corresponding match in the *template* string to be inserted into the result.

EXAMPLE

This example redirects CGI queries to **cgi-host**, prevents accesses to **/usr/forbidden**, and maps requests for images to a local image directory:

```
#
# Example URL translations
#
Redirect /cgi-bin/* http://cgi-host/*
Fail /usr/forbidden/*
Map /image-bin/* /usr/local/http/images/*
```

EFFECTIVE

Any changes take effect within 5 minutes

PERSISTENCE

Changes are persistent across system reboots.

messages

NAME

na_messages - record of recent console messages

SYNOPSIS

/etc/messages

DESCRIPTION

The default behavior of the node **syslogd** daemon (see na_syslogd(8)) is to print all logging messages of priority **info** or higher to the console, and to the **messages** file. A typical message is:

Fri Jun 10 14:31:37 PDT 2005 [rc]: NetApp Release 7.1 boot complete.

Every Saturday at 24:00, **/etc/messages** is moved to **/etc/messages.0**, **/etc/messages.0** is moved to **/etc/messages.1**, and so on. Message files are saved for a total of six weeks.

FILES

/etc/messages
messages file for current week **/etc/messages.[0-5]** messages file for previous weeks

SEE ALSO

na_syslog.conf(5)

netgroup

NAME

na_netgroup - Network groups data base

SYNOPSIS

/etc/netgroup

DESCRIPTION

netgroup defines network wide groups used for access permission checking during remote mount request processing. Each line defines a group and has the format:

groupname member-list

Each element in member-list is either another group name or a triple of the form:

(hostname, username, domainname)

The *hostname* entry must be fully qualified if the specified host is not in the local domain.

The node can also use the **netgroup** NIS map.

Since the node uses netgroups only in **/etc/exports** (see na_exports(5)), the *username* entry is ignored. The *domainname* field refers to the domain in which the netgroup entry is valid. It must either be empty or be the local domain; otherwise the netgroup entry is ignored. An empty entry allows a single **/etc/netgroup** file to be used for nodes in multiple domains.

A group definition can be at most 4096 bytes even when '\'s are used to extend the definition over several lines. The maximum nesting level when group names are used in the *member-lists* of other groups is 1000.

Modifications to the **/etc/netgroup** file may take upto 60 seconds to take effect.

EXAMPLE

This is a typical **netgroup** file:

trusted_hosts (adminhost,,) (zeus,,) (thor,,) (minerva,,)

untrusted_hosts
(sleepy,,) (dopey,,) (grumpy,,) (sneezy,,)

all_hosts
trusted_hosts untrusted_hosts

With this **netgroup** file it might make sense to modify **/etc/exports** to export / on the node only to *trusted_hosts*, but to export **/home** to *all_hosts*.

FILES

/etc/netgroup

/etc/exports
directories and files exported to NFS clients

/etc/hosts
host name data base

SEE ALSO

na_nis(8)

BUGS

The only place that netgroups can be used are in the options of the **exportfs** command (see *exportfs(1)*) and **/etc/exports**.

networks

NAME

na_networks - Network name data base

SYNOPSIS

/etc/networks

DESCRIPTION

The **networks** file contains information regarding the known networks which comprise the Internet. For each network a single line should be present with the following information:

official-network-name network-number aliases

Items are separated by any number of blanks and/or tab characters. A ‘‘#’’ indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. This file is normally created from the official network data base maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown networks.

Network number may be specified in the conventional ‘‘.’’ (dot) notation or as a 32 bit integer. Numbers may be specified in decimal (default), octal or hexadecimal. A number is interpreted as octal if it starts with the digit "0". A hexadecimal number must begin with "0x" or "0X." Network names may contain any printable character other than a field delimiter, newline, or comment character.

FILES

/etc/networks

nsswitch.conf

NAME

na_nsswitch.conf - Configuration file for name service switch

SYNOPSIS

/etc/nsswitch.conf

DESCRIPTION

The name service switch configuration file contains the preferred order in which name services will be contacted for name resolution by the node. For each map, the name services to be used and the lookup order is specified in this file. Currently four name services are supported. They are local files in the /etc directory, NIS, LDAP, and DNS. The maps or "databases" that are supported are hosts, passwd, shadows, group, and netgroups (LDAP is currently supported in the passwd, group, and netgroups map). Each line has the form:

map: order of name services

For example:

hosts: files nis dns ldap
passwd: files nis ldap

When trying to resolve a name, the services are contacted one by one, as per the order specified, until the name is successfully resolved. A name resolution failure occurs when no service can successfully resolve the name. When enumerating a map, enumeration happens over all the services specified for the map.

FILES

/etc/nsswitch.conf

SEE ALSO

na_setup(1)

passwd

NAME

na_passwd - Password file

SYNOPSIS

/etc/passwd

DESCRIPTION

The **passwd** file contains basic information about each user's account. It contains a one-line entry for each authorized user, of the form:

username:password:uid:gid:gcos_field:home_directory:login_shell

Required Fields:

username
The user's login name, not more than eight characters.

password
The user's password, in an encrypted form that is generated by the UNIX passwd function. However, if the encrypted password is stored in **/etc/shadow,** (see *shadow(5)*), the password field of **/etc/passwd** is empty.

uid
A unique integer assigned by the UNIX administrator to represent the user's account; its value is usually between 0 and 32767.

gid
An integer representing the group to which the user has been assigned. Groups are created by the UNIX system administrator; each is assigned a unique integer whose value is generally between 0 and 32767.

gcos-field
The user's real name. The name may be of any length; it may include capital letters as well as lower case, and may include blanks. The name may be empty.

home_directory The user's home directory. The home directory field may be empty.

login-shell
The default shell launched at login. This field may be empty.

EXAMPLE

Here is a sample passwd file when the /etc/shadow does not exist:

```
root:bDPu/ys5PBoYU:0:1:Operator:/:/bin/csh
dave:Qs5I6pBb2rJDA:1234:12:David:/u/dave:/bin/csh
dan:MNRWDsW/srMfE:2345:23:Dan::
jim:HNRyuuiuMFerx:::::
```

If the system keeps the passwords in the /etc/shadow, the file

/etc/passwd would be exactly the same but the password field would be empty.

```
root::0:1:Operator:/:/bin/csh
dave::1234:12:David:/u/dave:/bin/csh
dan::2345:23:Dan::
jim::::::
```

SEE ALSO

na_pcnfsd(8), na_cifs_access(1), na_cifs_setup(1)

psk.txt

NAME

na_psk.txt - psk.txt is disabled in this release of Data ONTAP.

qual_devices

NAME

na_qual_devices - Table of qualified disk and tape devices

SYNOPSIS

/etc/qual_devices

DESCRIPTION

The **qual_devices** file names storage devices qualified for use with Data ONTAP. This is a read-only file and must not be modified.

Disk and tape drives listed in this file are qualified for use with a Data ONTAP system. This file is read by the dynamic qualification process which is invoked to authenticate devices not listed in the internal tables of a particular Data ONTAP release. The dynamic qualification process may be invoked at system startup, controller takeover, or when a new device is detected.

WARNING

Do not modify or remove this file. However, it may be replaced with an updated version containing identification data for additionally qualified devices supplied by NetApp Inc.

NOTES

Each line in the file contains identification strings for a qualified device.

QUALIFICATION ERRORS

A qualification error will occur when Data ONTAP is unable to locate identification information for one or more storage devices detected by the system. To resolve qualification errors, verify the existence of /etc/qual_devices and ensure it represents the latest version available from NetApp Inc. Periodic console messages will be generated when a qualification error is present. All qualification errors MUST be resolved for continued system operation.

quotas

NAME

na_quotas - quota description file

SYNOPSIS

/etc/quotas

DESCRIPTION

The /etc/quotas file defines quotas that go into effect when quotas are enabled. All quotas are established on a per-volume basis. If a volume name is not specified in an entry of the /etc/quotas file, the entry applies to the root volume.

If any of the fields in the quota file contain special characters (#,-@) put the field in quotes ("@field#with,special-chars").

An entry in the quotas file can extend to multiple lines. However, the files, threshold, soft disk and soft files fields must be on the same line as the disk field; otherwise, they are ignored.

If you do not want to specify a value for a field in the middle of an entry, use a dash (-).

A line starting with a pound sign (#) is considered to be a comment.

If a quota target is affected by several /etc/quotas entries, the most restrictive entry or combination of entries applies.

The /etc/quotas file supports two types of character encoding: Unicode and root volume UNIX encoding (the language is specified for the root volume using the *vol lang* command).

You can edit the file from a PC or UNIX workstation. A file saved in a Unicode-capable editor, like Notepad, will be in Unicode. Otherwise it will be in the root volume UNIX encoding. Standard Generalized Markup Language (SGML) entities are allowed only in the root volume UNIX encoding.

Format of each entry:

target type[@vol] disk [files] [thres] [sdisk] [sfiles]

For example:

```
# Quota Target      type                disk  files thold sdisk sfile
# ------------      -----               ----  ----- ----- ----- -----

# Restrict user 'bob' to have 100M of space in the qtree '/vol/home'.
# Also, allow him to create 10,240 files and warn him when he goes
# over 90M and/or 9,216 files
bob                 user@/vol/home      100M  10K   90M   90M   9K
```

target
Specifies an explicit user, group or qtree to which the quota is being applied. An asterisk (*) applies the quota as a default to all members of the type specified in the entry that do not have an explicit quota in /etc/quotas.

user
A user can be specfied as a: unix user name, numerical user ID, windows account name, Windows SID or a comma separated list of multiple users.

-
A unix user name, as defined in the /etc/passwd file or in the password NIS map e.g. jsmith or "user,#special,chars"

-
A numerical unix user ID (If you specify 0 (root), no limits you set will be enforced, but usage will still be tracked.) e.g. 20

- The pathname of a file owned by that user
e.g. /vol/file_owned_by_jsmith

(NOTE: The quota restrictions only apply to the user that owns the file, not the file itself.)

- A Windows account name, which consists
of the domain name and the account name separated by a backslash e.g. "tech support\john#smith" or corp\jsmith

(NOTE: A file created by a member of the BUILTIN\Administrators group is owned by the BUILTIN\Administrators group, not by the member. When determining the amount of disk space or the number of files used by that user, Data ONTAP does not count the files that are owned by the BUILTIN\Administrators group.)

-
The text form of a Windows SID that represents a Windows account e.g. S-1-5-32-544

- If you want
to specifiy multiple users that are to be affected by a quota, use a comma separated list of the users. Each user in this list may be one of: unix user name, numerical user ID, file pathname, Windows account or Windows SID. e.g. john,jess,steph

e.g.
jsmith,23,"/vol/,qtree",/vol/file

(NOTE: The list may extend to multiple lines, but the last item must be on the same line as the quota type, disk limit, file limit and warning threshold values.) **group**
May be one of the following:

-
Unix group name, as defined in the /etc/group file or in the group NIS map e.g. eng1

- A numerical group ID
e.g. 30

(NOTE: If the group ID is 0 (root), no limits you set will be enforced, but usage will still be tracked.)

- The pathname of a file owned by that group
  e.g. /vol/vol1/archive

(NOTE: Specifying a file or directory does **not** affect the quota on that file or directory, only the GID.)

tree
May be one of the following:

- Complete path name to an existing qtree
  e.g. /vol/vol0/qtree

-
  If the qtree contains special characters, put the path in quotes. e.g "/vol/vol0/,qtree,with,special#chars"

default
An asterisk (*) is used to specify a default quota. This quota will be applied to users, groups or qtrees that are not specifically mentioned in the /etc/quotas file. This includes new users, groups or qtrees created after the default entry takes effect. e.g. * group 500m

(Apply a 500M restriction to all groups in the root vol.)

- Default
  user and group quotas can be specified on a per qtree or per volume basis. e.g. * user@/vol/vol0/qtree 500M

- Default qtree quotas can be specified on a
  per volume basis. e.g. * tree@/vol/vol0 500M

(NOTE: If the volume/qtree string is omitted from the type, the quota applies to the root volume.)

type[@/vol/dir/qtree_path]
Specifies what type the target is: user, group or tree. If the type is user or group, this field can optionally restrict this user or group quota to a specific volume or qtree.
user
- Restrict quota to the root volume e.g. bill user 500m

-
  Restrict quota to a specific volume e.g. bill user@/vol/vol1 500m

- Restrict quota to a specific qtree

e.g.
  bill user@/vol/vol1/qtree 500m

- Restrict quota to a specific qtree
  with special characters

e.g.
bill "user@/vol/vol1/,tree#@" 500m

group
- Restrict quota to the root volume e.g. dev group 500m

-
Restrict quota to a specific volume e.g. dev group@/vol/vol1 500m

- Restrict quota to a specific qtree

e.g.
dev group@/vol/vol1/qtree 500m

- Restrict quota to a specific qtree
with special characters (#,/..)

e.g.
dev "group@/vol/vol1/,qtree,@#" 500m

tree
- Restrict qtree in the target qtree e.g.
"/vol/vol0/,qtree#with,special,chars" tree 500m

-
Restrict all qtrees in the specified volume e.g. * tree@/vol/vol0 500M

disk The maximum amount of disk space that can be used by the target within the root volume, specified volume or qtree. The value in this field represents a hard limit that cannot be exceeded. Do **not** leave this field blank.
e.g. 10K, 100M, 5G, 5g,
Rules for specifying a value in this field: - This field may be "-" to indicate no limit, which is useful for tracking usage on a per-user/ per-group basis. (See example 3 below.)

- K is equivalent to 1,024
bytes, M means 2^20 bytes and G means 2^30 bytes. (Disk limits cannot be specified in terms of terabytes.)

-
If you omit K, M or G, Data ONTAP assumes K.

- It is not case sensitive.
Thus K, k, M, m, G or g all work.

- The value must be an integer.

-
If the quota limit is larger than the amount of space available in the volume, the quota will still be enabled and a warning will be printed to the console.

- Disk space limits are always rounded up to
  the nearest multiple of 4KB for translation into 4KB disk blocks.

- The maximum value you can enter
  for this field is 16TB.

files The maximum number of files that can be created by the target in the root volume specified volume or qtree. This is a hard limit that cannot be exceeded.
e.g. 12,000 , 2K , 1G , *Rules for specifying a value in this field:*

- Use a hyphen (-)
  if you don't want to impose a limit on the number of files.

-
  This field may be left blank to indicate no limit. If you leave this field blank, you cannot specify values for threshold, soft disk or soft files.

- Not case sensitive, i.e. K/k, M/m and
  G/g all work. K is equivalent to 1,024 files, M means 2^20 files and G means 2^30 files. You can omit the K, M or G. e.g. 100 (Means the maximum number of files is 100)

- The maximum value you
  can enter in the files field is 3G.

-
  If you do not specify K, M or G, Data ONTAP uses the literal value. K is **not** assumed in this case.

- The value must be an integer.

- The files field must be on the
  same line as the disk field. Otherwise, the files field is ignored.

threshold
The disk space usage point at which a *warning* of approaching quota limits is logged to the storage system's console and a SNMP trap is generated. If a write causes the quota target to exceed the threshold, the write still succeeds as long as no other limits are exceeded.
e.g. 5G , 5g , 100M , 10K , *Rules for specifying a value in this field:*

- Same as the disk field.

-
  The threshold field must be on the same line as the disk field. Otherwise, the threshold field is ignored.

soft disk
Same as the threshold field except that when the target's usage goes back under the soft disk limit, another syslog message and SNMP trap are generated. e.g. 5G , 5g , 100M , 10K , *Rules for specifying a value in this field:*

- Same as the disk field.

- The soft disk field must be
on the same line as the disk field. Otherwise, the soft disk field is ignored.

soft files
Same as soft disk, but for files. Specifies the number of files that the quota target can use before a warning is issued.
e.g. 1000, 5G , 5g , 100M , 10K , *Rules for specifying a value in this field:*

- Same as the files field.

- The soft file field must be
on the same line as the disk field. Otherwise, the soft file field is ignored.

domain directive
The **QUOTA_TARGET_DOMAIN <domain>** directive applies to all lines following it in the /etc/quotas file. When a domain is specified, the domain and a backslash are prepended to user names. See examples below.

user mapping directive

The QUOTA_PERFORM_USER_MAPPING directive applies to all lines following it in the /etc/quotas file. When ON, Data ONTAP's user name mapping support will be enabled. This means that unix user names will be mapped to their corresponding Windows account names and will be treated as a single quota target.

When OFF, this mapping does not occur. Quota user mapping is OFF unless explicitly enabled by using this directive. See examples below.

EXAMPLES

```
# Quota Target      type                 disk  files thold sdisk sfile
# -------------     -----                ----  ----- ----- ----- -----

# Restrict the user 'mhoward' to 500M of disk space and 51,200 files in
# the root volume
mhoward             user                 500M  50K

# Restrict the user 'lfine' to 500M of disk space, but give no
# restriction on the number of files
lfine               user@/vol/home       500M

# Place no restriction on either disk or file usage for the user
# 'tracker.'  (Useful for tracking usage on a per-user or per-group
# basis.)
tracker             user                 -     -

# Restrict the group 'stooges', and the qtree 'export' to 750MB and
# 76,800 files
stooges             group@/vol/vol0      750M  75K
/vol/vol0/export    tree                 750M  75K

# Restrict mhoward's usage in the export qtree to 50MB and 5,120
# files
mhoward          user@/vol/vol0/export   50M   5K
```

```
# Restrict the group 'stooges' to 100MB and 10,240 files in the export
# qtree
stooges          group@/vol/vol0/export  100M  10K

# Quota Target      type                 disk  files thold sdisk sfile
# -------------     -----                ----  ----- ----- ----- -----

# Restrict all users, except the ones in this file, to have 100M
# of space in the qtree '/vol/home'.  Also, allow them to create 10,240
# files and warn them when they go over 90M and/or 9,216 files
*                   user@/vol/home       100M  10K   90M   90M   9K

# Restrict all groups, except ones in this file, to have less than
# 500M of space and 70K files in the vol '/vol/vol0'.
*                   group@/vol/vol0      500M  70K

# Restrict all other qtrees in the root volume to be less than 500M in
# size and have less than 50K files
*                   tree                 500M  50K

# Restrict all other users in the qtree '/vol/vol0/export' to use less
# than 20M of space and 2K files
*                 user@/vol/vol0/export   20M   2K

# Restrict all other groups in the qtree '/vol/vol0/export' to use less
# than 200M of space and 20K files.  Warn if they go over 150M.
*                 group@/vol/vol0/export  200M  20K   150M

# Restrict the qtree '/vol/home' to a max size of 500M and 50K files
*                   tree@/vol/home       500M  50K

# Monitor the Window's user 'bill' in the domain 'corp' and warn him if
# his usage goes over 100M
corp\bill           user                 -     -     100M

corp\joe, fin\joe   user                 200M  40K   160M
corp\sue, sue       user                 100M  20K
corp\ann            user                 100M  -     90M

# Quota Target      type                  disk  files thold sdisk sfile
# -------------     -----                 ----  ----- ----- ----- -----
QUOTA_TARGET_DOMAIN corp

# The following entry will become corp\jim
jim                 user                 200M  -     -

# The following entry will become corp\beth
beth                user                 120M  50K   -
QUOTA_TARGET_DOMAIN
QUOTA_PERFORM_USER_MAPPING ON
# If corp\sam maps to usam, the following entry will become
#  corp\sam, usam   user .....
30 blocks
corp\sam            user                 50M
```

```
# If umary maps to corp\mary, the following entry will become
#  umary, corp\mary user ....
umary               user                300M
QUOTA_PERFORM_USER_MAPPING OFF
```

SEE ALSO

na_usermap.cfg(5)

rc

NAME

na_rc - system initialization command script

SYNOPSIS

/etc/rc

DESCRIPTION

The command script **/etc/rc** is invoked automatically during system initialization. Since the node has no local editor, **/etc/rc** must be edited from an NFS client with root access to **/etc**. Alternately, you can use the **setup** command to generate a new **/etc/rc** file without using NFS.

EXAMPLE

This is a sample **/etc/rc** file as generated by **setup**:

```
#Auto-generated by setup Tue Jun 2 21:23:52 GMT 1994
hostname toaster.mycompany.com
ifconfig e0 `hostname`-0
ifconfig e1a `hostname`-1
route add default MyRouterBox 1
routed on
timezone Atlantic/Bermuda
savecore
```

FILES

/etc/rc

SEE ALSO

na_nfs(1), na_setup(1), na_timezone(1)

registry

NAME

na_registry - registry database

SYNOPSIS

/etc/registry

DESCRIPTION

The file **/etc/registry** stores a variety of persistent information for ONTAP. For example, the **options** command uses this file to save option values, eliminating the need to manually add lines to the /etc/rc file.

Do not edit this file directly; if you do, some aspects of ONTAP will not operate correctly. Several backups of the registry database exist and are automatically used if the original registry becomes unusable. In particular, **/etc/registry.lastgood** is a copy of the registry as it existed after the last successful boot.

If you back up the configuration files in the /etc directory, the /etc/registry file should be included. After restoring all the configuration files, a reboot will be required to complete the restore (for example, in order to reload the registry, and to re-execute /etc/rc).

ERRORS

If the /etc/rc file contains an explicit "options" statement whose value conflicts with the value of the option stored in the registry, you will see an error message at boot time like this:

```
** Option cifs.show_snapshot is being set to "true" in /etc/rc, and this
** conflicts with a value - "off" - loaded from the registry.
** Commands in /etc/rc always override the registry at boot time,
** so the value of cifs.show_snapshot is now "true".
```

Similarly, if you execute the "options" statement interactively, and the /etc/rc file contains an explicit "options" statement for the same option, you may see an error message such as this:

```
** Option autosupport.enable is being set to "off", but this conflicts
** with a line in /etc/rc that sets it to "on".
** Options are automatically persistent, but the line in /etc/rc
** will override this persistence, so if you want to make this change
** persistent, you will need to change (or remove) the line in /etc/rc.
```

By removing the explicit options statements from /etc/rc, you can eliminate these warnings about inconsistencies between /etc/rc and the registry.

FILES

/etc/registry (primary registry)
/etc/registry.bck (first-level backup)
/etc/registry.lastgood (second-level backup)

resolv.conf

NAME

na_resolv.conf - configuration file for domain name system resolver

SYNOPSIS

/etc/resolv.conf

DESCRIPTION

The resolver configuration file contains information that is read by the resolver routines. The file is designed to be human readable and contains a list of keywords with values that provide various types of resolver information. Semicolon (';') or pound ('#') starts comment. So, any character after ';' or '#' is ignored until the next line. Lines in bad formats are ignored entirely.

The different configuration options are:

nameserver *address*
This specifies the Internet address (in dot notation) of a name server that the resolver should query. Up to 3 name servers may be listed, one per keyword. If there are multiple servers, the resolver queries them in the order listed. When a query to a name server on the list times out, the resolver will move to the next one until it gets to the bottom of the list. It will then restart from the top retrying all the name servers until a maximum number of retries are made.

search *domain-list*
This specifies the search list for host-name lookup. The search list is normally determined from the local domain name; by default, it begins with the local domain name, then successive parent domains that have at least two components in their names. This may be changed by listing the desired domain search path following the **search** keyword with spaces or tabs separating the names. Most resolver queries will be attempted using each component of the search path in turn until a match is found. Note that this process may be slow and will generate a lot of network traffic if the servers for the listed domains are not local, and that queries will time out if no server is available for one of the domains.

The search list is currently limited to six domains with a total of 256 characters.

The keyword and value must appear on a single line, and the keyword (e.g. **nameserver**) must start the line. The value follows the keyword, separated by white space.

FILES

/etc/resolv.conf

SEE ALSO

na_rc(5), RFC 1034, RFC 1035

rmtab

NAME

na_rmtab - Remote mounted file system table

SYNOPSIS

/etc/rmtab

DESCRIPTION

/etc/rmtab maintains the list of client mount points between server reboots. The list of client mount points can be obtained by using the **MOUNTPROC_DUMP** remote procedure call, or by using the UNIX *showmount(1)* command. When the server successfully executes a mount request from a client, the server appends a new entry to the file. When the client issues an unmount request, the corresponding entry is marked as unused. When the server reboots, unused entries are deleted from the file.

BUGS

Entries may become stale if clients crash without sending an unmount request. The file may be removed before rebooting the server in which case the server will lose information about any active client mount entries on reboot.

serialnum

NAME

na_serialnum - System serial number file

SYNOPSIS

/etc/serialnum

DESCRIPTION

The file **/etc/serialnum** should contain the serial number of your machine.

If **/etc/serialnum** does not exist, it is an indication that your machine could not obtain the serial number from the hardware. In this case you need to enter the serial number manually. The serial number is found on the back of the machine in the lower right hand corner. You should see a tag that says:

NetApp SN: xxxx

Use a text editor to create **/etc/serialnum** and put the machine's serial number in it. The file should contain a single line that only has the serial number. The file is used to help NetApp's customer service group process your AutoSupport email more efficiently.

FILES

/etc/serialnum

WARNINGS

A warning is issued to the console if **/etc/serialnum** contains a different value other than the hardware serial number in which case it is automatically overwritten with the hardware serial number. Also if the hardware serial number and **/etc/serialnum** do not exist, then a warning is issued to the console.

services

NAME

na_services - Internet services

SYNOPSIS

/etc/services

DESCRIPTION

The **services** file contains information mapping between port numbers and service names. This file exists purely for reference purposes and is not currently used by Data ONTAP. Modifying entries in this file will have no effect on the node. Removing entries will not disable ports or services. For information on how to change which port numbers a service uses (if possible), see the relevant manual page for that service. Such changes will not update the **services** file.

Each line contains a service name followed by a port number, a ‘‘/’’, and a protocol, for example 20/tcp. Legal protocol names are ‘‘tcp’’ and ‘‘udp’’. Port numbers are decimal numbers in the range of 0 to 65535. A service name may contain any printable character other than the comment character (i.e. no spaces, tabs, newlines, or ‘‘#’’).

Items are separated by any number of blanks and/or tab characters. A ‘‘#’’ indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

FILES

/etc/services

SEE ALSO

na_hosts(5)

shadow

NAME

na_shadow - shadow password file

SYNOPSIS

/etc/shadow

DESCRIPTION

The **shadow** file provides more secure storage for the user's password (which would otherwise be in **/etc/passwd).** When the password field of an entry in **/etc/passwd** is empty, **/etc/shadow** must contain a corresponding entry with the same user name but a non-empty encrypted password.

username:password:

The following list explains the required fields:

username
: The user's login name, not more than eight characters.

password
: The user's password, in an encrypted form that is generated by the UNIX passwd function.

There can be other fields in the **/etc/shadow** file following the ":" after the **password.**

EXAMPLE

Here is a sample shadow password file entry:

```
dave:Qs5I6pBb2rJDA:
```

SEE ALSO

na_pcnfsd(8), na_nsswitch.conf(5)

sis

NAME

na_sis - Log of Advanced Single Instance Storage (SIS) activities

SYNOPSIS

/etc/log/sis

DESCRIPTION

The **sis** log file contains a log of SIS activities for this node. The file lives in **/etc/log** on the root volume.

Every Sunday at midnight, **/etc/log/sis** is moved to **/etc/log/sis.0**; **/etc/log/sis.0** is moved to **/etc/log/sis.1**; and so on. The suffix can go up to 5, so the old **/etc/log/sis.5** will be deleted. SIS activities are saved for a total of seven weeks.

Each entry of the **/etc/log/sis** file is a single line containing the following space-separated fields.

```
timestamp path session-ID event_info
```

The following is a description of each field.

timestamp
Displayed in **ctime()** format, e.g. Fri Jul 17 20:41:09 GMT 2008. Indicates the time this event was recorded.

path
The full path to a SIS volume as shown below

```
/vol/volume_name
```

session-ID
The session ID is as shown below:

```
[sid: 1220249325]
```

event_info
The event which is being logged. Some events may have extra information in parentheses. The current event types are:

Sis Restart
When a SIS operation resumes from a checkpoint. The event is augmented within parenthesis with the stage from which it is restarting.(Restarting from [- | gathering | sorting | saving_pass1 | saving_pass2 checking |
checking_pass1 | checking_pass2])

Begin (*operation information)*

When a SIS operation is first kicked off, there can be multiple reasons which trigger it. The event is augmented with the following additional information in parenthesis.

schedule : If the SIS operation is kicked off as per the configured or default schedule.

sis start scan : Corresponds to "sis start -s", when we are instructed to scan the entire file system for duplicated blocks.

sis check : If we are specifically instructed to perform fingerprint database checking.

sis start snapvault : If the snapvault initiated the SIS operation.

sis start : When the SIS operation is kicked off to perform deduplication based on the changelogs.

Undo
Corresponds to "sis undo" command.

Stage (*amount_processed*)
An event is logged at the end of each stage along with the amount of processing that was done in that stage. The different stages can be Sort, Dedup Pass1, Dedup Pass2 and Verify. Note the Verify event is logged at the start of sis check operation. The events for each are shown below :

```
Thu Sep 01 10:31:05 GMT 2008 /vol/dense_vol [sid: 12] Sort (2560 fp entries)

Thu Sep 22 10:33:03 GMT 2008 /vol/dense_vol [sid: 12] Dedup Pass1 (0 dup entries)

Thu Oct 13 10:35:00 GMT 2008 /vol/dense_vol [sid: 12] Dedup Pass2 (2559 dup entries)

Thu Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 12] Verify
```

End (*processed_size* KB)
When a long-running SIS operation (either Begin or Undo) completes successfully. The size of data processed is included in the event.

Error (*Error_message*)
If a SIS operation aborts or fails to start, the cause of the error is indicated.

Config (*schedule_string*)
When a "sis config" command successfully set or modified the SIS schedule on a volume. The new schedule string is logged with the event.

Enable
When the SIS is enabled on a volume.

Disable
When the SIS is disabled on a volume.

Stats (*statistics string*)
When each changelog is processed ,statistics are logged with this event.

Info (*operation information)*
Some of the operations that are logged within parenthesis in the **Info** event are :

sis start : This corresponds to the event when user issues the sis operation based on changelogs.

sis check : When a sis check operation starts to perform fingerprint database checking.

sis start scan : This information is logged when a "sis start -s" command is issued.

sis start schedule : When a sis operation starts based on its schedule.

operation pending : The maximum number of sis operations running is 8. If a sis operation is issued or scheduled when this upper limit is already reached, it gets queued as a pending operation and prints this message in **Info** event.

starting pending operation : A sis operation is queued when 8 sis operations are already running. This message is logged when later on system becomes free and the pending operation starts its execution at the time of schedule start.

EXAMPLE

On the successful completion of such a sis start -s operation, the log file should have the following entries:

```
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 11] Info (sis start scan)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 11] Begin (sis start scan)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 11] Sort (0 fp entries)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 11] Dedup Pass1 (0 dup entries)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 11] Dedup Pass2 (0 dup entries)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 11] Stats (blks gathered 0,finger
prints sorted 0,dups found 0,new dups found 0,blks deduped 0,finger prints checked 0,finger
prints deleted 0)
Tue Jul 12 02:02:05 GMT /vol/dense_vol [sid: 11]  End (0 KB)
```

On the successful completion of a sis start operation, the log file should have the following entries:

```
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 9] Info (sis start)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 9] Begin (sis start)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 9] Sort (0 fp entries)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 9] Dedup Pass1 (0 dup entries)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 9] Dedup Pass2 (0 dup entries)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 9] Stats (blks gathered 0,finger
prints sorted 0,dups found 0,new dups found 0,blks deduped 0,finger prints checked
0,finger prints deleted 0)
Tue Jul 12 02:02:05 GMT 2008 /vol/dense_vol [sid: 9] End (0 KB)
```

A SIS operation initiated by schedule and based on change log is the most common case. In this case a pending operation has started its execution. On the successful completion of such an operation, the log file should have the following entries:

```
Tue Jul 12 02:01:03 GMT 2008 /vol/dense_vol [sid: 0] Info (starting pending operation)
Tue Jul 12 02:01:03 GMT 2008 /vol/dense_vol [sid: 0] Begin (schedule)
Tue Jul 12 02:01:04 GMT 2008 /vol/dense_vol [sid: 0] Sort (128000 fp entries)
Tue Jul 12 02:01:04 GMT 2008 /vol/dense_vol [sid: 0] Dedup Pass1 (0 dup entries)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 0] Dedup Pass2 (127999 dup entries)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 0] Stats (blks gathered 0,finger prints sorted 0,dups found
0,new dups found 127999,blks deduped 127541,finger prints checked 0,finger prints deleted 0)
Tue Jul 12 02:02:22 GMT 2008 /vol/dense_vol [sid: 0] End (2356080 KB)
```

The log file will have following entries if sis start operation starts from a checkpoint corresponding to saving_pass2 stage :

```
Thu Sep 18 03:32:23 GMT 2008 /vol/dense_vol [sid: 15] Sis Restart (Restarting from saving_pass2 stage )
Thu Sep 18 03:32:23 GMT 2008 /vol/dense_vol [sid: 15] Begin (sis start)
Thu Sep 18 03:32:23 GMT 2008 /vol/dense_vol [sid: 15] Dedup Pass2 (130559 dup entries)
Thu Sep 18 03:32:30 GMT 2008 /vol/dense_vol [sid: 15] Stats (blks gathered 0,finger prints sorted 0,dups found 0,new dups found 130559,blks deduped 130091,finger prints checked 0,finger prints delet...
Thu Sep 18 03:32:30 GMT 2008 /vol/dense_vol [sid: 15] End (522240 KB)
```

On the successful completion of such a sis check operation, the log file should have the following entries: (sis check)

```
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 14] Info (sis check)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 14] Begin (sis check)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 14] Verify
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 14] Merge(0 stale entries)
Tue Jul 12 02:01:05 GMT 2008 /vol/dense_vol [sid: 14] Stats (blks gathered 0,finger
prints sorted 0,dups found 0,new dups found 0,blks deduped 0,finger prints
checked 0,finger prints deleted 0)
Tue Jul 12 02:02:05 GMT 2008 /vol/dense_vol [sid: 14] End (0 KB)
```

If a SIS operation aborts, the Error event will replace the End event.

```
Fri Jul 15 00:40:31 GMT 2008 /vol/dense_vol [sid: 18] Begin(schedule)
Fri Jul 15 18:58:26 GMT 2008 /vol/dense_vol [sid: 18] Error (Volume is full)
```

The Undo is the only other long-running event, similar to the Begin event, is terminated by either End or Error.

```
Fri Jul 15 18:58:26 GMT 2008 /vol/dense_vol [sid: 19] Undo
Fri Jul 15 18:58:26 GMT 2008 /vol/dense_vol [sid: 19] End (34670 KB)
```

The Enable, Disable and Config events are only logged when they complete successfully.

```
Fri Jul 15 18:58:26 GMT 2008 /vol/dense_vol [sid: 20] Enable
Fri Jul 15 18:58:26 GMT 2008 /vol/dense_vol [sid: 20] Disable
Fri Jul 15 18:58:26 GMT 2008 /vol/dense_vol [sid: 20] Config (sun-sat@0-23)
```

FILES

/etc/log/sis
SIS log file for current week.

/etc/log/sis.[0-5] SIS log files for previous weeks.

SEE ALSO

na_sis(1)

sm

NAME

na_sm - network status monitor directory

SYNOPSIS

/etc/sm

DESCRIPTION

The network status monitor provides information about the status of network hosts to clients such as the network lock manager. The network status monitor keeps its information in the **/etc/sm** directory.

The **/etc/sm/state** file contains an integer that is incremented each time the node is booted.

The **/etc/sm/monitor** file contains a list of network hosts the node is monitoring.

The **/etc/sm/notify** file contains a list of network hosts that made an NLM lock request to the node. Each time the node reboots, it tries to notify the hosts of its new state information. You can remove this file if you want the node to stop notifying the hosts in this file.

BUGS

If the node cannot resolve a host name in the **/etc/sm/notify** file or if a host in the **/etc/sm/notify** file does not exist on the network any more, the node logs an error message each time it tries to contact the host. The error message is similar to the following:

```
[sm_recover]: get RPC port for  failed
```

To stop the error messages, remove the **/etc/sm/notify** file.

snapmirror

NAME

na_snapmirror - Log of SnapMirror Activity

SYNOPSIS

/etc/log/snapmirror

DESCRIPTION

The SnapMirror log file contains a log of SnapMirror activity for this node. The file lives in **/etc/log** on the root volume of both the source and destination nodes. When the option **snapmirror.log.enable** is set to **on**, all the SnapMirror activities will be recorded in this log file. See na_options(1) for details regarding how to enable and disable this option. Every Sunday at 00:00, **/etc/log/snapmirror** is moved to **/etc/log/snapmirror.0**, **/etc/log/snapmirror.0** is moved to **/etc/log/snapmirror.1**, and so on. The suffix can go up to 5. This process is called rotation. SnapMirror log entries are saved for a total of six weeks.

Each entry of the **/etc/log/snapmirror** file is a single line consisting of space-separated fields. All log entries begin with a type field and a timestamp field. The final field may be enclosed by parentheses, in which case it may contain spaces. The timestamp field contains a fixed number of spaces, and as such can be parsed as five space-delimited fields. Which fields appear, and in what order they appear in, is determined by the type field of log entry (which is the first field).

Following is a description of each field.

type Indicate the type of the entry, which also determines the format of the rest of the entry. It can be one of the following values:

log
log facility activity

Format: *type timestamp event_info...*

sys
system-wide activity

Format: *type timestamp event_info...*

tgt
snapvault target activity

Format: *type timestamp volume target event_info...*

src
source activity

Format: *type timestamp source destination event_info...*

dst
destination activity

Format: *type timestamp source destination event_info...*

cmd
user command activity

Format: *type timestamp source destination event_info...*

scn
replication check source activity

Format: *type timestamp source destination event_info...*

chk
replication check destination activity.

Format: *type timestamp source destination event_info...*

vol
volume-wide activity

Format: *type timestamp volume event_info...*

slk
softlock addition-deletion activity

Format: *type timestamp softlock event_info...*

timestamp
Displayed in **ctime()** format, e.g. Fri Jul 17 20:41:09 GMT. Indicates the time this event is recorded.

volume Specifies the name of the volume to which this entry applies.

target This is the name and type of the target for this entry. Targets are volume-wide actions, typically snapshot creations. It is displayed as two colonseparated fields, as follows:

target_type:*target_name*

The target name may be an empty string.

source This is the name of the source node and the volume name or qtree path to be mirrored. The name is specified as two colon-separated fields, as follows:

host:*path*

This field may be '-' when not applicable for the event.

destination
This is the name of the destination node and the volume name or qtree path of the destination. The name is specified as two colon-separated fields, same as in the *source* field.

This field may be '-' when not applicable for the event.

event_info
This field contains the event which is being logged. Some events may have extra information in parentheses.

Request (*IP address | transfer type*) A transfer request has been sent (destination) or received (source). On source side, the IP address of the destination node that made the request is included in parentheses. On destination side, the transfer type is included in the parentheses.

Start The beginning of a transfer.

Start (Snapshots to check=#num, level={data|checksum}, {check|fix}, {quick|full} mode)
The beginning of a replication check or fix session. The session options are included in the parentheses. All options appear on the destination side log but only the "snapshots to check" option appears in source side log.

Restart (*@ num* KB)
The beginning of a restarted transfer.

End (*num* KB done)
The completion of a transfer. The total size of the transfer in KB is included in the parentheses.

End (src_only=*num_1*, dst_only=*num_2*, mismatch=*num_3*)
The completion of a replication check or fix session. The summary of the session is included in the parentheses. The summary is present only on the destination side logs. Source side logs will not contain any summary information.

Abort (*error msg*)
A transfer is aborted. The error message is included in the parentheses.

Defer (*reason*)
Indicates a transfer is deferred because of a resource limitation. The reason for the deferment is included in the parentheses.

Sync_start
The start of synchronous mirroring mode for the SnapMirror relationship specified by this log entry.

Sync_end (*reason*)
The end of synchronous mirroring mode for the SnapMirror relationship specified by this log entry. The reason for dropping out of synchronous mode is included in the parentheses.

Quiesce_start
The beginning of quiesce process.

Quiesce_end
The completion of quiesce process.

Quiesce_failed (*reason*)
The failure of quiesce process. The reason for failure is included in the parentheses.

Rollback_start
The beginning of a rollback process for a qtree SnapMirror or SnapVault.

Rollback_end
The completion of a rollback process for a qtree SnapMirror or SnapVault.

Rollback_failed (*reason*)
The failure of a rollback process for a qtree SnapMirror or SnapVault. The reason for failure is included in the parentheses.

Coalesce_start (*snapshot*)
The beginning of a coalesce process for a SnapVault qtree. The base snapshot for the coalesce operation is included in the parentheses.

Coalesce_end
The completion of a coalesce process for a SnapVault qtree.

Coalesce_failed (*reason*)
The failure of a coalesce process for a SnapVault qtree. The reason for failure is included in the parentheses.

Target_start
The beginning of a SnapVault target.

Target_end
The completion of a SnapVault target.

Target_failed (*reason*)
The failure of a SnapVault target. The reason for failure is included in the parentheses.

Start_logging
SnapMirror log was enabled.

End_logging
SnapMirror log was disabled.

SnapMirror_on (*cause*)
SnapMirror was enabled on this host. The operation or process that caused SnapMirror to become enabled is specified in the parentheses.

SnapMirror_off (*cause*)
SnapMirror was disabled on this host. The operation or process that caused SnapMirror to become disabled is specified in the parentheses.

SnapVault_on (*cause*)
SnapVault was enabled on this host. The operation or process that caused SnapVault to become enabled is specified in the parentheses.

SnapVault_off (*cause*)
SnapVault was disabled on this host. The operation or process that caused SnapVault to become disabled is specified in the parentheses.

Resume_command
User issued **snapmirror resume** command.

Break_command
User issued **snapmirror break** command.

Release_command
User issued **snapmirror release** command.

Abort_command

Abort_command (*type*)
User issued **snapmirror abort** command. The *type* will only be present if the abort was issued with additional options which changed the type of the abort.

Resync_command (*common snapshot*)
User issued **snapmirror resync** command. The common snapshot for the resync operation is included in the parentheses.

Restore_resync_command (*common snapshot*) User issued **snapvault restore -r** command. The common snapshot for the resync operation is included in the parentheses.

Migrate_command
User issued **snapmirror migrate** command.

Request_check (*snapshot_name*)
A request for single snapshot during replication check session. This is source side log entry. Each snapshot being checked in a replication check session will have its entry. Name of snapshot is included in the parentheses.

Checking_snapshot *source snapshot_name* (*timestamp*, cpcount=*num_2*, snapid=*id*) to *dest_snapshot_name* (*timestamp*, cpcount=*count*, snapid=*id*) The beginning of a single snapshot comparison during replication check. It is logged on both source and destination.

Abort_check
replication check session for SnapMirror or SnapVault aborted. Reason of abort is included in the parentheses.

Abort_check_command
User issued replication **check abort** command. Corresponding log file entry appears with **cmd** *type*.

Data_differ ({block *blk_num* in *file_path* | VBN *vbn*})
Replication check found a data block mismatch. Either the block number and the inode path or Volume Block Number (VBN) is included in the parentheses.

Unique_in_src (*entry_type* for *entry_path*) Replication check found an entry only present in the *source*. The entry type and entry path are included in the parentheses.

Unique_in_dst (*entry_type* for *entry_path*) Replication check found an entry only present in the *destination*. The entry type and entry path are included in the parentheses.

Size_differ (*path*)
Replication check found a file size mismatch in specified inode. The inode path is included in the parentheses.

Type_differ (*path*)
Replication check found a inode type mismatch. The inode path is included in the parentheses.

UID_differ (*path*)
Replication check found a user ID mismatch for specified inode. The inode path is included in the parentheses.

GID_differ (*path*)
Replication check found a group ID mismatch for specified inode. The inode path is included in the parentheses.

Perm_differ (*path*)
Replication check found a permission or dosbit mismatch for specified inode. The inode path is included in the parentheses.

Atime_differ (*path*)
Replication check found a mismatch in the last access time for specified inode. The inode path is included in the parentheses.

Mtime_differ (*path*)
Replication check found a mismatch in the last modification time for specified inode. The inode path is included in the parentheses.

Ctime_differ (*path*)
Replication check found a mismatch in the last size/status change time for specified inode. The inode path is included in the parentheses.

Crtime_differ (*path*)
Replication check found a mismatch in the creation time for specified inode. The inode path is included in the parentheses.

Rdev_differ (*path*)
Replication check found a device number mismatch for specified inode. The inode path is included in the parentheses.

DOSbits_differ (*path*)
Replication check found a DOS bits mismatch for specified inode. The inode path is included in the parentheses.

ACL_differ (*path*)
Replication check found an NT or NFS V4 ACL mismatch for specified inode. The inode path is included in the parentheses.

Hardlink_differ (*path*)
Replication check found a hardlink for specified inode, but the inode on *destination* doesn't match between the links. The inode path is included in the parentheses.

Qtree_oplock_differ (*path*)
Replication check found oplock setting mismatch for a qtree. The qtree path is included in the parentheses.

Qtree_security_differ (*path*)
Replication check found security setting mismatch for a qtree. The qtree path is included in the parentheses.

Hole_uses_disk_space (*path*)
Replication check found unnecessary disk usage for specified inode, this however is not a mismatch. The inode path is included in the parentheses.

Convert_command
User issued **snapmirror convert** command.

Older_snapshot
Updating from a snapshot which is older than the current base snapshot.

Snapshot_delete (*snapshot name*)
A snapshot is deleted from this volume. The snapshot name is included in the parentheses.

Snapshot_replace (*snapshot name*)
A SnapVault snapshot has been replaced after a SIS operation with a newer snapshot of the same name. The snapshot name is included in the parentheses.

FILER_REBOOTED
The node is rebooted.

WORM_LOG_FAIL (*reason*)
Write to WORM log file failed. The reason for failure is included in the parentheses.

WORM_LOG_FAILURE_RECOVER_START
The beginning of the recovery of the failed WORM log entries.

WORM_LOG_FAILURE_RECOVER_END
The end of the recovery of the failed WORM log entries.

Softlock_add (*operation*)
A softlock is added. The operation that added the softlock is included in the parentheses.

Softlock_add_pending (*operation*)
A softlock is added as a pending softlock. The operation that added the softlock is included in the parentheses.

Softlock_delete (*operation*)
A softlock is deleted. The operation that deleted the softlock is included in the parentheses.

Softlock_delete_pending (*operation*) A pending softlock is deleted. The operation that deleted it is included in the parentheses.

Softlock_mark_pending (*operation*)
A softlock is marked as pending. The operation that marked it is included in the parentheses.

Resend log open fail
Opening of resend log which has checksum mismatches from last aborted VSM transfer, failed.

Logging checksum mismatch (*@VBN*)
Looging a checksum mismatch in resend log.

EXAMPLES

A typical entry in **/etc/log/snapmirror** looks like:

```
dst Fri Jul 17 22:50:18 GMT node1:srcvol node2:dstvol Request (Update)
```

The above example shows an update request recorded by the destination side for a SnapMirror relationship from node:srcvol to node2:dstvol that happened at the recorded time.

A typical Replication check session in **/etc/log/snapmirror** on destination looks like:

```
chk Wed Jan 19 01:07:39 GMT woolf:/vol/vol1 milton:/vol/vol1 Request (check)
chk Wed Jan 19 01:07:39 GMT woolf:/vol/vol1 milton:/vol/vol1 Start (Snapshots to check = 2, level= data, check, full)
chk Wed Jan 19 01:07:39 GMT woolf:/vol/vol1 milton:/vol/vol1 Checking_snapshot milton(0033587346)_vol1.5 (Jan 18...
chk Wed Jan 19 01:07:48 GMT woolf:/vol/vol1 milton:/vol/vol1 Checking_snapshot nightly.0 (Jan 18 00:00, cpcount =...
chk Wed Jan 19 01:07:57 GMT woolf:/vol/vol1 milton:/vol/vol1 End (src_only = 0, dst_only = 0, mismatch = 0)
```

A typical Replication check session in **/etc/log/snapmirror** on source looks like:

```
scn Wed Jan 19 00:58:27 GMT woolf:/vol/vol1 milton:/vol/vol1 Request (172.29.19.15)
scn Wed Jan 19 00:58:27 GMT woolf:/vol/vol1 milton:/vol/vol1 Start (Snapshots to check = 2)
scn Wed Jan 19 00:58:27 GMT woolf:/vol/vol1 milton:/vol/vol1 Request_check (milton(0033587346)_vol1.5)
scn Wed Jan 19 00:58:27 GMT woolf:/vol/vol1 milton:/vol/vol1 Checking_snapshot milton(0033587346)_vol1.5 (Jan 18...
scn Wed Jan 19 00:58:36 GMT woolf:/vol/vol1 milton:/vol/vol1 Request_check (nightly.0)
scn Wed Jan 19 00:58:36 GMT woolf:/vol/vol1 milton:/vol/vol1 Checking_snapshot nightly.1 (Jan 18 00:00, cpcount =...
scn Wed Jan 19 00:58:45 GMT woolf:/vol/vol1 milton:/vol/vol1 End
```

A typical softlock logging in **/etc/log/snapmirror** looks like:

```
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011e.054.node1:vol3 Softlock_add (Transfer)
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011b.054.node1:vol3 Softlock_delete (Transfer)
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011b.054.node1:vol3 Softlock_delete (Revert)
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011b.054.node1:vol3 Softlock_delete (Release)
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011b.054.node1:vol3 Softlock_delete (Clean_softlocks)
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011b.054.node1:vol3 Softlock_add (RSM_forward)
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011b.054.node1:vol3 Softlock_delete (RSM_forward)
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011b.054.node1:vol3 Softlock_delete (Snapmirror_destinations)
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011b.054.node1:vol3 Softlock_delete_pending (Transfer)
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011b.054.node1:vol3 Softlock_add_pending (Transfer)
slk Wed May 10 03:06:15 GMT  state.softlock.vol1.0000011b.054.node1:vol3 Softlock_mark_pending (Transfer)
```

FILES

/etc/log/snapmirror
SnapMirror log file for current week.

/etc/log/snapmirror.[0-5]
SnapMirror log files for previous weeks.

SEE ALSO

na_snapvault(1)

snapmirror.allow

NAME

na_snapmirror.allow - List of allowed destination nodes

SYNOPSIS

/etc/snapmirror.allow

DESCRIPTION

The **/etc/snapmirror.allow** file provides for one of two ways for controlling SnapMirror access to a source node.

The **snapmirror.access** option is the preferred method for controlling SnapMirror access on a SnapMirror source node. See na_options(1) and **na_protocolaccess (8)** for information on setting the option. If the option **snapmirror.access** is set to "legacy", the **snapmirror.allow** file defines the access permissions.

The **snapmirror.allow** file exists on the source node used for SnapMirror. It contains a list of allowed destination nodes to which you can replicate volumes or qtrees from that node.

The file format is line-based. Each line consists of the hostname of the allowed destination node.

The **snapmirror.checkip.enable** option controls how the allow check is performed. When the option is **off**, which is the default, the entries in the allow file must match the hostname of the destination node as reported by the **hostname** command. When the option is **on**, the source node resolves the names in the **snapmirror.allow** to IP addresses and then checks for a match with the IP address of the requesting destination node. In this mode, literal IPv4 addresses (e.g. 123.45.67.89), literal IPv6 addresses (e.g. fe:dc:ba:98:76:54:32:10) and fully qualified names (e.g. toaster.acme.com) may be valid entries in the allow file.

Note that the allow file entry must map to the IP address of the originating network interface on the destination node. For example, if the request comes from the IP address of a Gbit Ethernet interface e10 which is given the name "toaster-e10", then the allow file must contain "toaster-e10" or "toaster-e10.acme.com" so the name resolves to the correct IP address.

A local SnapMirror, between two volumes or qtrees on the same node, does not require an entry in the allow file.

EXAMPLE

The following **snapmirror.allow** file on a node allows nodes named **toaster** and **fridge** to replicate volumes or qtrees from it:

```
toaster
fridge
```

SEE ALSO

na_snapmirror.conf(5) na_protocolaccess(8)

snapmirror.conf

NAME

na_snapmirror.conf - volume and qtree replication schedules and configurations

SYNOPSIS

/etc/snapmirror.conf

DESCRIPTION

The **/etc/snapmirror.conf** file exists on the node containing the mirror used for SnapMirror.

There are two types of lines in the configuration file: lines that define mirror relationships and lines that define connections to source nodes to be used in the relationship definitions. Relationship definition lines are used to define the mirror relationships for destination volumes on this node. Connection definition lines are optional and are used to specify specific network connections to the source volume and allow the specification of dual paths to the source volume.

Each relationship line of the file specifies the volume or qtree to be replicated, arguments for the replication, and the schedule for updating the mirror. You may only have one line for each destination volume or qtree. The maximum number of relationship entries supported is limited to 1024. Any entry after this limit is ignored.

Each relationship entry of the **/etc/snapmirror.conf** file is a single line containing space-separated fields. The entry has this format:

source destination arguments schedule

If the *source* or *destination* field contains one or more space characters (on account of it including a qtree name with space(s)), then the field must be enclosed in double quotes. If the field value itself contains one or more double quotes, then each of these double quotes must be escaped by preceding it with an additional double quote.

The following list describes the fields in each entry:

source This is the name of the source host, and the volume name, or the path of the qtree to be mirrored. The name is specified as two colon-separated fields, as follows:

host:volname

host:/vol/volume/qtree

Note that the *host* field is not necessarily the hostname of the node (unlike the first field of the *destination* entry). You can specify a network resolvable name, IP address or connection name. The host field can be considered a definition of how to reach the source over the network.

destination
This is the hostname (must match the result of the hostname command) of the destination node and the name of the destination volume or the path of the destination qtree. The name is specified as two colon-separated fields, as follows:

name:volume

name:/vol/volume/qtree

The *name* field must match the hostname of the destination node (use the *hostname(1)* function to check this).

arguments
These are a comma-separated list of arguments for the transfer. To specify no arguments, enter a dash (‘‘-’’) in this field. Each argument is specified as a *key* and a *value* pair, as follows:

key=value

Currently, there are the following argument keys:

cksum This controls which checksum algorithm is used to protect the data transmitted by SnapMirror. Currently supported values are "none", "crc32c", and "crc32c_header_only". The value "crc32c_header_only" has been added only for volume SnapMirror and is not supported for synchronous SnapMirror and qtree SnapMirror.

kbs

The *value* for this argument specifies the maximum speed (in kilobytes per second) at which SnapMirror data is transferred over the network. The **kbs** setting is used to throttle network bandwidth consumed, disk I/O, and CPU usage. By default, the node transfers the data as fast as it can. The throttle value is not used while synchronously mirroring.

tries The *value* for this argument specifies the maximum number of attempts that the destination will make to complete a scheduled snapmirror update. A retry will be attempted on the first minute after the previous attempt was abandoned. Notice that retries are only attempted for retry-able errors, and that some errors do not count as a retry. The **tries** setting is used to limit the number of retries, for instance to assure that backup transfers are started within a designated backup window, or else abandoned entirely until the next scheduled update. The syntax is "tries=N" or "tries=unlimited", where N is greater or equal to 0, and N is less or equal to 1000000000. If this *value* is set to 0, the transfer is never started. If no try count is specified, the default is "unlimited". Manually started transfers are never retried irrespective of the the *value* of this argument.

restart
This controls the behavior of the SnapMirror scheduler with respect to restartability. If *value* is set to **always**, then an interrupted transfer will always restart, if it has a restart checkpoint and the conditions are the same as before the transfer was interrupted. If *value* is set to **never**, then an interrupted transfer will never restart, even if it has a restart checkpoint. By default, SnapMirror behaves like the **always** case, unless it has passed the next scheduled transfer time, in which case it will begin that scheduled transfer instead of restarting.

ignore_atime
The *value* for this argument can be **enable** or **disable**. This option only applies to Qtree SnapMirror relationships. When the value is **enable**, SnapMirror will ignore files which have only their access times changed for incremental transfers. When the value is **disable**, SnapMirror will transfer metadata for all modified files. If not specified, the default is **disable**.

outstanding (deprecated)
This argument controls the performance versus synchronicity trade-off for synchronous mirrors. The *value* for this argument is a number followed by the suffixes: **ops** (operations), **ms** (milliseconds) or **s** (seconds). Setting a value less than 10s configures the mirror to run in fully synchronous mode. Setting a value greater than or equal to 10s configures the mirror to run in semisynchronous mode. This argument is ignored for asynchronous mirrors. Please note that this is a deprecated option. Use the schedule field to specify the synchronous mode for the mirror.

wsize This sets the TCP window size to use for the connection. Due to how TCP negotiates window sizes, the size of the receive window will initially be large and gradually work its way down to the size specified.

visibility_interval
The *value* for this argument is a number optionally followed by the suffixes: **s** (seconds), **m** (minutes) or **h** (hours). If a suffix is not specified, *value* is interpreted as seconds. This argument controls the amount of time before an automatic snapshot is created on the source volume that is synchronously mirrored. The *value* is the number of seconds between automatically created snapshots. The default value is 3 minutes. A small number here can negatively affect the performance of the mirror. This argument is ignored for asynchronous mirrors.

compression
The *value* for this argument can be **enable** or **disable**. This argument can only be used when a connection definition is used for the relationship entry. Using this argument without a connection definition will throw an error message. When the value is **enable**, SnapMirror will compress/decompress the data that is transferred between the source and destination node. If not specified, the default is **disable**.

connection_mode
The *value* for this argument can be **inet** or **inet6**.

When the value is **inet6**, the connection between the primary and secondary will be established using IPv6 addresses only. If there is no IPv6 address configured for the primary, then the connection will fail. When the value is **inet**, the connection between the primary and secondary will be established using IPv4 addresses only. If there is no IPv4 address configured on the primary, then the connection will fail. When this argument is not specified, then the connection will be tried using both IPv6 and IPv4 addresses. **inet6** mode will have higher precedence than **inet** mode. If a connection request using **inet6** mode fails, SnapMirror will retry the connection using **inet** mode.

This argument is not meaningful when an IP address is specified instead of a hostname. If the IP address format and connection mode doesn’t match, the operation prints an error message and aborts.

schedule
This is the schedule used by the destination node for updating the mirror. It informs the SnapMirror scheduler when transfers will be initiated. The schedule field can contain the word **sync** to specify fully synchronous mirroring, **semisync** to specify semi-synchronous
mirroring, or a cron-style specification of when to update the mirror. The cron-style schedule contains four space-separated fields:

*minute hour day-of-month day_*o*f-week*

Each field consists of one or more numbers or ranges. If a field contains more than one value, the values are separated from each other by a comma. A field consisting solely of an asterisk (‘‘*’’) is the same as a field enumerating all possible legal values for that field. A field consisting solely of a dash (‘‘-’’) represents a null value; any *schedule* with a dash in one of its fields will never run any scheduled transfers. Values in a field can take any of the following forms:

number

first-last

first-last/step

A value with a dash in it specifies a range; it is treated as containing all the values between *first* and *last*, inclusive. A range value with a slash specifies skips of *step* size in the range. For example, the value of the entry ‘‘0-23/4’’ would be the same as that of the entry ‘‘0,4,8,12,16,20’’.

minute Which minutes in each hour to update on. Values are from 0 to 59.

hour Which hours in the day to update on. Values are from 0 to 23.

day-of-month
Which days in the month to update on. Values are from 1 to 31.

day-of-week
Which days in the week to update on. Values are from 0 (Sunday) to 6 (Saturday).

Whenever the current time matches all the specified *schedule* fields, a transfer from the *source* to the *destination* will be invoked.

The other type of line allowed in this file is a *connection definition* line. These lines define an alternate name for the source node that can be used as the source host in the relationship lines. They are used to describe more specifically the parameters for the connection(s) to the source node. SnapMirror supports the multi path specification for both asynchronous and synchronous mirrors.

Each connection definition is a single line giving a name to one or two pairs of IP addresses along with a mode of operation for the connection. The lines are specified in the following format:

name **=** *mode(source_ip_addr1* **,** *dest_ip_addr1* **) (** *source_ip_addr2* **,** *dest_ip_addr2* **)**

name This is the name of the connection you would like to define. This name is to be used as the source node in relationship definitions.

mode The mode is optional and specifies the mode in which two IP address pairs will be used. Two modes are allowed multiplexing and failover mode and are specified by using the **multi** and **failover** keywords. If not specified, multiplexing mode is used.

The multiplexing mode causes snapmirror to use both paths at the same time. If one should fail, it will switch to use the remaining path only and use both again should the failing path be repaired.

Failover mode causes snapmirror to use the first path as the desired path and only use the second path should problems arise with the first path.

source_ip_addr1

source_ip_addr2 dest_ip_addr1 dest_ip_addr2 These are resolvable network names or IP addresses that define a path through the network between the source and the destination. The source addresses are the IP addresses of interfaces to use on the source and respectively for the destination. The pairing denotes a path from source to destination.

EXAMPLES

The following **snapmirror.conf** entry indicates that node fridge's qtree **home**, in volume **vol2** will mirror qtree **home**, in volume **vol1** from the node toaster. Transfer speed is set at a maximum rate of 2,000 kilobytes per second. The four asterisks mean transfers to the mirror are initiated every minute, if possible. (If a previous transfer is in progress at the minute edge, it will continue; a new transfer will be initiated at the first minute edge after the transfer has completed.)

```
toaster:/vol/vol1/home fridge:/vol/vol2/home kbs=2000 * * * *
```

The following **snapmirror.conf** entry is similar to the above example, except that it shows how qtree names with spaces and double quotes can be specified. This entry indicates that node fridge's qtree **x y"z** in volume **vol2** will mirror qtree **x y"z** in volume **vol1** from the node toaster.

```
"toaster:/vol/vol1/x y""z" "fridge:/vol/vol2/x y""z" kbs=2000 * * * *
```

The following **snapmirror.conf** entry indicates that node mynode1's volume **home_mirror** will mirror volume **home** via the mynode0-gig interface. (The mynode0-gig interface is whatever IP address mynode1 can resolve that name to. In this case, it might be a gigabit ethernet link on node mynode0.) The mirror is updated at 9:30 a.m., 1:30 p.m., and 7:30 p.m., Monday through Friday. The asterisk means that the data replication schedule is not affected by the day of month; it is the same as entering numbers 1 through 31 (comma-separated) in that space. The dash in the *arguments* field indicates that both the **kbs** and **restart** arguments are set to default.

```
mynode0-gig:home mynode1:home_mirror - 30 9,13,19 * 1,2,3,4,5
```

The following **snapmirror.conf** entry makes transfers every half hour, with the first at 8:15 a.m., and the last at 6:45 p.m. The asterisks mean that the data replication schedule is not affected by the day of month or week; in other words, this series of transfers are initiated every day.

```
node1:build node2:backup - 15,45 8,9,10,11,12,13,14,15,16,17,18 * *
```

The following **snapmirror.conf** entry, between the **docs** qtree on dev and **docs_bak** on icebox, is kicked off on every Sunday, at 12:00 midnight.

```
dev:/vol/dept/docs icebox:/vol/backup/docs_bak - 0 0 * 0
```

The following **snapmirror.conf** entry, between the **home** and **backup** volume on icebox, is kicked off once every half-past the hour between 7:30 a.m. and 9:30 p.m., and once at midnight.

```
icebox:home icebox:backup - 30 0,7-21 * *
```

The following **snapmirror.conf** entry, between the **db** volumes on fridge-gig dev and icebox, is kicked off on every five minutes, starting at 0. (Note that fridge-gig is just a network interface name. In this case, it could be a gigabit ethernet link on fridge.)

```
fridge-gig:db icebox:db - 0-55/5 * * *
```

This can be extended to use the multiple path options and synchronous mirroring.

```
fridge-con = failover(fridge-gig,icebox-gig)(fridge-slow,icebox-slow)
fridge-con:db icebox:db - sync
```

This can further be extended to use Network compression for Asynchronous Volume SnapMirror transfers.

```
fridge-con = multipath(fridge-gig,icebox-gig)(fridge-slow,icebox-slow)
fridge-con:db icebox:db compression=enable * * * *
```

This changes the relationship into synchronous mode and the connection specifies that we should use a gigabit ethernet path for the mirroring where only if that connection fails, use a slower network connection. Even if you would like to use one path from source to destination, it is a good idea to specify a connection line in your configuration file. This can reduce problems seen with name resolution affects on the relationship configuration line.

CONCURRENT STREAM LIMITS

The number of concurrent replication streams are limited for each ONTAP platform. This limitation is put in order to restrict the overuse of resources and bandwidth on the source and destination of the streams. These limits do not scale with the capabilities of the platform, e.g. cpu, memory, networking, etc. The following tables give the maximum number of concurrent transfers that each platform may allow.

Personality: **Default**

Model	Maximum Transfers
FAS250	4
F810 F820 F825 FAS920 FAS270 GF270 GF825	8
F840 F880 FAS940 FAS960 GF940 GF960 GF980	16

The above platforms have the same maximum concurrent transfer limit for each transfer type.

Personality: **Default**

Model	Volcopy Sync SM Legacy QSM Legacy SV Src Dst	Legacy VSM Src	Legacy VSM Dst	MP VSM Src Dst	QSM SV Src Dst
FAS980 FAS3020 FAS3040 FAS3050 V3020 V3040 V3050	16	16	16	50	64
FAS3070 V3070	16	16	64	50	64
FAS6030 V6030	24	24	24	100	96
FAS6070 V6070	32	32	32	100	128

Personality: **Nearstore**

Model	Volcopy Src	Volcopy Dst	Legacy VSM Src	Legacy VSM Dst	MP VSM Src	MP VSM Dst	Sync SM Src Dst	Legacy QSM Legacy SV Src	Legacy QSM Legacy SV Dst	QSM SV Src	QSM SV Dst
R100 R150 R200	64	64	64	64	64	64	16	64	128	64	128
FAS3020	16	16	16	16	50	100	16	16	32	80	80
FAS3040	16	32	16	32	50	100	16	16	64	160	160
FAS3050	16	32	16	32	50	100	16	16	64	120	120

FAS3070	16	64	16	32	50	100	16	16	128	320	320
FAS6030	24	48	24	48	100	200	24	24	96	512	512
FAS6070	32	64	32	64	100	200	32	32	128	512	512

```
VSM Src - Volume Snapmirror Source
VSM Dst - Volume Snapmirror Destination
QSM Src - Qtree  Snapmirror Source
QSM Dst - Qtree  Snapmirror Destination
SV  Src - Snapvault Source
SV  Dst - Snapvault Destination
```

SEE ALSO

na_snapmirror.allow(5)

stats_preset

NAME

na_stats_preset - stats preset file format

SYNOPSIS

/etc/stats/preset

DESCRIPTION

The **stats** utility supports preset queries, using the -p argument. A preset includes the statistics to be gathered, and the format for display. Using presets not only saves typing when entering commands from the CLI, it also allows greater flexibility in formatting the data than is possible on the command line. Each preset is described in an XML file, stored in the applicance directory /etc/stats/preset. The name of each preset file is *presetname*.**xml**.

PRESET FILE FORMAT

Preset Element
The main element of a preset file is a single **preset**. The preset consists of attributes, plus one or objects that should be included in the preset. A simple preset to display all information from the **system** object using the default formats might be:

<?xml VERSION = "1.0" ?>
<preset>
<object name="system">
</object> </preset>

Preset Attributes
The following attributes are available for the **preset** element.

orientation
Output orientation, "row" or "column", see -r/-c command line options.

outfile
Output file. See -o command line option. When used with a **stats start** and **stats stop** pair this option is only active with **stats stop**. In such pairs the same preset is typically used with both commands, although this is not mandatory.

interval
Interval between output. See -i command line option.

icount Number of outputs when using interval output. See -n command line option.

print_header
Whether or not to print a output header. Default: true

print_object_names
In row output, whether or not to include object names in the output. Default: true

print_instance_names
In column output, whether or not to include instance names as a column in the output. Default: true

print_footer
After printing a set of counters print a footer string. Default: false. In multiple-count outputs the footer is printed after each iteration.

pre_header
A header string that is printed prior to data headers. Default: none

use_regex
Allow extended regular expressions for instance and counter names. Default: false

print_zero_values
Determines whether counters with zero values should be displayed. The default setting displays all counters, except for counters that are flagged as not-zero-printing by default. The allowed values are **default**, **true** and **false**. This option only affects row output.

column_delimeter
In column output, the text to print between each column, changing the default TAB spacing.

catenate_instances
In column output, whether or not to catenate all instance counters into a long line, or to split the output so that each instance goes on its own line. Default: false

The following example specifies a preset with column output, that displays values each second:

```
<?xml VERSION = "1.0" ?>
<preset orientation="column" interval="1" > ...
</preset>
```

Objects
The **object** element specifies an object that is to be used in the preset. It has attributes, as listed below, and optional counters and instances.

The following example shows a preset using the **system** and **volume** objects:

```
<?xml VERSION = "1.0" ?>
<preset>
<object name="system">
...
</object>
<object name="volume">
...
</object>
</preset>
```

The following table lists object attributes.

name

Object name. If "*" is used, this means all objects. This attribute is mandatory

Object counters and instances
Each object may list which instances and/or which counters are to be used in the preset, using the **instance** and **counter** elements. If no instances or counters are listed then all instances, all counters are assumed.

Counters may be listed for an object, or for an instance. If a counter is listed for an object then it applies to all instances of the object in the preset. If a counter is listed for an instance then it only applies to that instance.

The following example shows a case where counter "global_counter" is being used for all instances, but "counter_0" is only being used for a specific instance.

<?xml VERSION = "1.0" ?>
<preset>
<object name="OBJNAME">
<instance name="instance0"> <counter name="counter_0"> </counter">
</instance>

<counter name="global_counter"> </counter>
</object>
</preset>
See below for more information on the syntax for counters and instances.

Counters
Object counters are specified with the **counter** element. The required attribute "name" specifies the counter name, or "*" may be used to indicate all counters for an object.

A counter also has the following elements:

title Title to be used in column headers.

width Column width in output, in characters.

The following example shows a column named "disk_io" formatted in a column 8 characters wide, with a column header of "Disk I/O":
<counter name="disk_io">
<title>Disk I/O</title>
<width>8</width>
</counter">

Instances
Object instance are specified with the **instance** element. The required attribute "name" attribute specifies the instance name.

An instance has the following optional elements:

counter
An instance-specific counter. The element may occur multiple times.

Note that if no counters are listed for an instance then the default set of counters for the preset will be used. This is either counters listed at the object level, or all counters for the object.

The following example shows an instance with two counters:

```
<instance name="instance0">
<counter name="counter0"> <title">Cnt0</title>
</counter">
<counter name="counter1"> <title">Cnt1</title>
</counter">
</instance">
```

EXAMPLE

The following example shows a preset with output similar to the **sysstat** command. It might be invoked as:

stats show -p sysstat -i 1

```
<?xml VERSION = "1.0" ?>
<!-- This preset is similar to the tradition ‘sysstat’
command, using column output -->
<preset orientation="column"
print_instance_names="false" catenate_instances="true" > <object name="system">
<counter name="cpu_busy"> <width>4</width>
<title>CPU</title>
</counter>
<counter name="nfs_ops"> <width>6</width>
<title>NFS</title>
</counter>
<counter name="cifs_ops"> <width>6</width>
<title>CIFS</title>
</counter>
<counter name="http_ops"> <width>6</width>
<title>HTTP</title>
</counter>
<counter name="net_data_recv"> <width>8</width>
<title>Net in</title> </counter>
<counter name="net_data_sent"> <width>8</width>
<title>Net out</title> </counter>
<counter name="disk_data_read"> <width>8</width>
<title>Disk read</title> </counter>
<counter name="disk_data_written"> <width>8</width>
<title>Disk write</title> </counter>
</object>
</preset>
```

SEE ALSO

na_stats.1

symlink.translations

NAME

na_symlink.translations - Symbolic link translations to be applied to CIFS path lookups

SYNOPSIS

/etc/symlink.translations

DESCRIPTION

When the CIFS server encounters a symbolic link (also called a "symlink," or "soft link"), it attempts to follow the link. If the symlink target is a path that starts with a "/", the node must interpret the rest of the path relative to the root of the file system. On the node, there is no way to know how NFS clients (which must be used to create the symlinks) might have mounted filesystems, so there is no reliable way to follow such absolute, or "rooted" symlinks. The **symlink.translations** file enables you to use absolute symlinks by mapping them to CIFS-based paths.

The entries in this file are similar to the httpd.translations file. There are two formats for file entries, as follows:

Map *template result*

Widelink *template [@qtree] result*

Any request that matches *template* is replaced with the *result* string given. Note that the result path for a "Map" entry must point to a destination within the share to which the client is connected. This is because the client has only been authenticated to that share; therefore access is limited to the same share for security reasons. A result path for a "Widelink" entry may point anywhere, thus the name "wide symlink" or widelink for short. Widelinks have these limitations-- the node share on which the symlink resides must be enabled for wide symlinks, the CIFS client must support Microsoft’s Dfs protocol, and the destination must be able to function as a Dfs leaf node. By using Dfs requests, the node causes the client to authenticate with the destination and thus enforces security. To enable a node share for "wide symlinks", use the "cifs shares -change" node console command.

Both templates and results might (and usually do) contain wildcards (a star "*" character). The wildcard behaves like a shell wildcard in the *template* string, matching zero or more characters, including the slash ("/") character. In the *result* string, a wildcard causes text from the corresponding match in the *template* string to be inserted into the result.

The entries are examined in the order they appear in the file until a match is found or the lookup fails.

EXAMPLES

This example maps absolute symlinks that start with "/u/home" to go to the node path "/vol/vol2/home". Also, symlinks starting with "/u" go to "/vol/vol0". Note that you should put the more restrictive entries first to avoid premature mapping since the matches are done in order.

```
#
# Example Map symlink translations
#
Map /u/home/* /vol/vol2/home/*
Map /u/* /vol/vol0/*
```

The next example maps absolute symlinks that start with "/u/docs/" to go to the node path "\\node_name\engr\tech pubs". Note that widelink result paths use CIFS pathname syntax (backslashes are separators, spaces in path components are allowed, and so on).

```
#
# Example Widelink symlink translation
#
Widelink /u/docs/* \\node_name\engr\tech pubs\*
```

The next example maps absolute symlinks that start with "/u/joe". Note that depending on how NFS mounts are set up, it is possible that there could be several absolute symlinks pointing to "/u/joe" which need to have differing destinations. The qtree in which a symlink resides can optionally be used to distinguish destinations. Thus, following an absolute symlink starting with "/u/joe" in qtree /vol/vol1/mixed takes the client to "\\node_name\home\joe", while symlinks in other qtrees take the client to "\\node_name\test tools\joe".

```
#
# Example Widelink symlink translations #
Widelink /u/joe/* @/vol/vol1/mixed \\node_name\home\joe\*
Widelink /u/joe/* \\node_name\test tools\joe\*
```

Note that there is no theoretical reason why a wide symlink can’t point to another node or indeed any NT server, though it may be difficult to imagine the translated link making sense to the Unix client which created the original symlink.

```
#
# More Widelink examples
#
Widelink /u/joe/* @/vol/vol1/mixed \\mynode2\users2\joe\*
Widelink /u/joe/* \\joe-PC\Program Files\*
```

SEE ALSO

na_cifs_shares(1)

syslog.conf

NAME

na_syslog.conf - syslogd configuration file

DESCRIPTION

The **syslog.conf** file is the configuration file for the **syslogd** daemon (see na_syslogd(8)). It consists of lines with two fields separated by tabs or spaces:

selector
    action

The *selector* field specifies the types of messages and priorities to which the line applies. The *action* field specifies the action to be taken if a message the **syslogd** daemon receives matches the selection criteria.

The *selector* field is encoded as a *facility*, a period (‘‘.’’), and a *level* , with no intervening white-space. Both the *facility* and the *level* are case insensitive.

The *facility* describes the part of the system generating the message, and is one of the following keywords: **auth**, **cron**, **daemon**, **kern**, and **local7.** Here’s a short description of each *facility* keyword:

kern
    Messages generated by the node kernel.

daemon
    System daemons, such as the **rshd** daemon (see na_rshd(8)), the routing daemon (see na_routed(1)), the SNMP daemon (see na_snmpd(8)), and so on.

auth
    The authentication system, for example, messages logged for Telnet sessions.

cron
    The system’s internal cron facility.

local7
    The system’s audit logging facility. All messages coming from the audit logging facility are logged at level debug.

The *level* describes the severity of the message, and is a keyword from the following ordered list (higher to lower): **emerg**, **alert**, **crit**, **err**, **warning**, **notice**, **info**, and **debug**.

Here is a short description of each *level* keyword:

emerg
    A panic condition that results in the disruption of normal service.

alert
A condition that should be corrected immediately, such as a failed disk.

crit
Critical conditions, such as hard disk errors.

err
Errors, such as those resulting from a bad configuration file.

warning
Warning messages.

notice
Conditions that are not error conditions, but that may require special handling.

info
Informational messages, such as the hourly uptime message (see na_uptime(1)**).**

debug
Debug messages used for diagnostic purposes. These messages are suppressed by default.

If a received message matches the specified *facility* and is of the specified *level* (or a higher *level*), the action specified in the *action* field will be taken.

Multiple *selectors* may be specified for a single *action* by separating them with semicolon (‘‘;’’) characters. It is important to note, however, that each *selector* can modify the ones preceding it.

Multiple *facilities* may be specified for a single *level* by separating them with comma (‘‘,’’) characters.

An asterisk (‘‘*’’) can be used to specify all *facilities* (except local7) or all *levels* .

The special *level* **none** disables a particular *facility* .

The *action* field of each line specifies the action to be taken when the *selector* field selects a message. There are four forms:

A pathname (beginning with a leading slash).
Selected messages are appended to the specified file.

A hostname (preceded by an at (‘‘@’’) sign).
Selected messages are forwarded to the **syslogd** daemon on the named host.

/dev/console.
Selected messages are written to the console.

An asterisk.
Selected messages are written to the console.

Blank lines and lines whose first non-blank character is a pound (‘‘#’’) character are ignored.

It is recommended that all **/etc/syslog.conf** files include the line

```
*.info          /etc/messages
```

so that all messages are logged to the /etc/messages file.

EXAMPLES

A configuration file might appear as follows:

```
# Log all kernel messages, and anything of level err or
# higher to the console.
*.err;kern.*                  /dev/console

# Log anything of level info or higher to /etc/messages.
*.info                        /etc/messages

# Also log the messages that go to the console to a remote
# loghost system called adminhost.
*.err;kern.*                  @adminhost

# Also log the messages that go to the console to the local7
# facility of another remote loghost system called adminhost2
# at level info.
*.err;kern.*                  local7.info@adminhost2

# The /etc/secure.message file has restricted access.
auth.notice                   /etc/secure.message
```

Also see the sample configuration file in **/etc/syslog.conf.sample**

FILES

/etc/syslog.conf
The **syslogd** configuration file. **/etc/syslog.conf.sample** Sample **syslogd** configuration file.

BUGS

The effects of multiple selectors are sometimes not intuitive. For example ‘‘daemon.crit,*.err’’ will select ‘‘daemon’’ facility messages at the level of ‘‘err’’ or higher, not at the level of ‘‘crit’’ or higher.

SEE ALSO

na_messages(5)

tape_config

NAME

tape_config - Directory of tape drive configuration files

SYNOPSIS

/etc/tape_config

DESCRIPTION

The **tape_config** directory contains NetApp-approved tape configuration files. These files allow Data ONTAP to recognize a tape drive and to properly set its various parameters without the tape drive parameters being built into Data ONTAP. Only NetApp-approved tape configuration files should be placed into the **tape_config** directory.

The **tape_config** directory of the latest release of Data ONTAP contains tape configuration files for tape drives that are configured exclusively with tape configuration files. Other approved files may be added to the directory by the user as tape qualifications are completed by NetApp Inc and configuration files become available.

All NetApp-approved tape configuration files may be found at http://support.netapp.com/NOW/download/tools/tape_config/index.shtml. To use configuration files shown in this page -- if your version of Data ONTAP does not already support the tape drive(s) -- first verify that the configuration file is approved for your version of Data ONTAP. Then copy the desired file(s) to the **/etc/tape_config** directory. The file(s) may be renamed if necessary. When an attached tape drive is accessed, Data ONTAP detects the presence of files in the directory and install the parameters for the tape drive.

SEE ALSO

na_cloned_tapes(5)

NOTES

External tape configuration files do not override built-in tape drive parameters. If the tape drive is already supported by Data ONTAP, remove the corresponding tape configuration file.

If a tape drive is represented in tape_config directory, remove any reference from the /etc/cloned_tapes file that attempts to cause the drive to use the parameters of another drive.

The command **storage show tape supported** displays all tape drives that are currently supported directly within Data ONTAP. If any tape drives are connected to the system, the command will also any show tape drives specified by tape configuration files.

treecompare

NAME

na_treecompare - Log of treecompare activities

SYNOPSIS

/etc/log/treecompare

DESCRIPTION

The **treecompare** log file contains a log of treecompare activities for this node. The file lives in **/etc/log** on the root volume.

Every Sunday at midnight, **/etc/log/treecompare** is moved to **/etc/log/treecompare.0**; **/etc/log/treecompare.0** is moved to **/etc/log/treecompare.1**; and so on. The suffix can go up to 5, so the old **/etc/log/treecompare.5** will be deleted. Treecompare activities are saved for a total of seven weeks.

Each entry of the **/etc/log/treecompare** file is a single line containing the following space-separated fields.

```
timestamp tree1 tree2 event_info
```

The following is a description of each field.

timestamp
Displayed in **ctime()** format, e.g. Fri Jul 17 20:41:09 GMT. Indicates the time this event was recorded.

tree1 The name of the host1 and the full path for tree1 as shown below:

```
host1:tree1_path
```

tree2 The name of the host2 and the full path for tree2 as shown below:

```
host2:tree2_path
```

event_info
The event which is being logged. Some events may have extra information in parentheses. The existing event types are:

Start (cmp_level={data|checksum}, {compare|ignore} NT ACL)
The beginning of a treecompare session. The command options are included in the parentheses.

End (tree1_only=*num_1*, tree2_only=*num_2*, mismatch=*num_3*)
The completion of a treecompare session. The summary of the session is included in the parentheses.

Abort (*error_msg*)
A treecompare operation was aborted. The error message is included in the parentheses.

Data_differ (block *blk_num* in *file_name*) Found a data block mismatch. The block number and the file name are included in the parentheses.

Unique_in_tree1 (*entry_type* for *entry_path*) Found an entry only present in the first tree. The entry type and entry path are included in the parentheses.

Unique_in_tree2 (*entry_type* for *entry_path*) Found an entry only present in the second tree. The entry type and entry path are included in the parentheses.

Size_differ (*file_name*)
Found a file size mismatch. The file name is included in the parentheses.

Type_differ (*entry_name*)
Found a directory entry type mismatch. The entry name is included in the parentheses.

UID_differ (*entry_name*)
Found a user ID mismatch for a directory entry. The entry name is included in the parentheses.

GID_differ (*entry_name*)
Found a group ID mismatch for a directory entry. The entry name is included in the parentheses.

Perm_differ (*entry_name*)
Found a permission mismatch for a directory entry. The entry name is included in the parentheses.

Atime_differ (*entry_name*)
Found a mismatch in the last access time for a directory entry. The entry name is included in the parentheses.

Mtime_differ (*entry_name*)
Found a mismatch in the last modification time for a directory entry. The entry name is included in the parentheses.

Ctime_differ (*entry_name*)
Found a mismatch in the last size/status change time for a directory entry. The entry name is included in the parentheses.

Crtime_differ (*entry_name*)
Found a mismatch in the creation time for a directory entry. The entry name is included in the parentheses.

Rdev_differ (*entry_name*)
Found a device number mismatch for a directory entry. The entry name is included in the parentheses.

DOSbits_differ (*entry_name*)
Found a DOS bits mismatch for a directory entry. The entry name is included in the parentheses.

ACL_differ (*entry_name*)
Found an NT ACL mismatch for a directory entry. The entry name is included in the parentheses.

Hardlink_differ (*entry_name*)
Found a hardlink for a directory entry, but the inode on tree2 doesn’t match between the links. The entry name is included in the parentheses.

Skip (*attr_type* for *entry_name*)
Skipped the comparison of an unsupported attribute type for a directory entry. The attribute type and the entry name are included in the parentheses.

Inode_Num_differ (*entry_name*)
Found an inode number mismatch for a directory entry. The entry name is included in the parentheses.

Inode_Gen_differ (*entry_name*)
Found an inode generation number mismatch for a directory entry. The entry name is included in the parentheses.

Inode_Tid_differ (*entry_name*)
Found an inode tree id mismatch for a directory entry. The entry name is included in the parentheses.

CIFS_reserve_differ (*entry_name*)
Found a cifs space reservation mismatch for a directory entry. The entry name is included in the parentheses.

HOLES_reserve_differ (*entry_name*)
Found a holes space reservation mismatch for a directory entry. The entry name is included in the parentheses.

BLOCK_reserve_differ (*entry_name*)
Found a block space reservation mismatch for a directory entry. The entry name is included in the parentheses.

QT_oplock_differ (*entry_name*)
Found oplock setting mismatch for a qtree. The entry name is included in the parentheses.

QT_security_differ (*entry_name*)
Found security setting mismatch for a qtree. The entry name is included in the parentheses.

QT_reserve_differ (*entry_name*)
Found space reservation setting mismatch for a qtree. The entry name is included in the parentheses.

EXAMPLE

A typical treecompare session in **/etc/log/treecompare** looks like:

```
Tue Jun 24 00:05:20 GMT fridge:/vol/src1/.snapshot/snap1/qt1 toaster:/vol/
dst4/.snapshot/snap1/qt1 Start (cmp_level = data, compare NT ACL)
Tue Jun 24 00:05:44 GMT fridge:/vol/src1/.snapshot/snap1/qt1 toaster:/vol/
dst4/.snapshot/snap1/qt1 End (tree1_only = 0, tree2_only = 0, mismatch = 0)
```

This example shows a treecompare session which used comparison level *data* and did compare NT ACLs. At the end of the session, the summary shows no mismatches were found.

The next example shows a log with several mismatches.

```
Tue Jun 24 00:07:31 GMT fridge:/vol/src1/.snapshot/snap2/qt1 toaster:/vol/
dst4/.snapshot/snap1/qt1 Start (cmp_level = checksum, ignore NT ACL)
Tue Jun 24 00:07:32 GMT fridge:/vol/src1/.snapshot/snap2/qt1 toaster:/vol/
dst4/.snapshot/snap1/qt1 Atime_differ (.)
Tue Jun 24 00:07:32 GMT fridge:/vol/src1/.snapshot/snap2/qt1 toaster:/vol/
```

```
dst4/.snapshot/snap1/qt1 Atime_differ (./subd1)
Tue Jun 24 00:07:42 GMT fridge:/vol/src1/.snapshot/snap2/qt1 toaster:/vol/
dst4/.snapshot/snap1/qt1 Atime_differ (./subd1/dfile2)
Tue Jun 24 00:07:42 GMT fridge:/vol/src1/.snapshot/snap2/qt1 toaster:/vol/
dst4/.snapshot/snap1/qt1 Mtime_differ (./subd1/dfile2)
Tue Jun 24 00:07:42 GMT fridge:/vol/src1/.snapshot/snap2/qt1 toaster:/vol/
dst4/.snapshot/snap1/qt1 Size_differ (./subd1/dfile2)
Tue Jun 24 00:07:42 GMT fridge:/vol/src1/.snapshot/snap2/qt1 toaster:/vol/
dst4/.snapshot/snap1/qt1 Data_differ (block 0 in ./subd1/dfile2)
Tue Jun 24 00:07:51 GMT fridge:/vol/src1/.snapshot/snap2/qt1 toaster:/vol/
dst4/.snapshot/snap1/qt1 Data_differ (block 1000 in ./subd1/dfile2)
Tue Jun 24 00:07:52 GMT fridge:/vol/src1/.snapshot/snap2/qt1 toaster:/vol/
dst4/.snapshot/snap1/qt1 End (tree1_only = 0, tree2_only = 0, mismatch = 7)
```

FILES

/etc/log/treecompare
Treecompare log file for current week.

/etc/log/treecompare.[0-5] Treecompare log files for previous weeks.

SEE ALSO

usermap.cfg

NAME

na_usermap.cfg - mappings between UNIX and Windows NT accounts and users

SYNOPSIS

/etc/usermap.cfg

DESCRIPTION

The **usermap.cfg** file explicitly maps Windows NT users to the correct UNIX account and UNIX users to the correct Windows NT account. Each line in **/etc/usermap.cfg** has the format:

[*IP-qual:*] [*NT-domain* \] *NTUser* [*direction*] [*IP_qual:*] *UnixUser*

Lines are processed sequentially.

The following table describes the variables in the **usermap.cfg** file description.

IP-qual
: An IP qualifier that the node uses to match a user. You use an IP qualifier to narrow a match. *IP-qual* can be a regular IP address, a host name, a network name, or a network name with a subnet specified in dot notation.

NT-domain
: Specifies the domain to match or the domain to use for a mapped UNIX account. The default is the domain in which the node is installed.

NTUser
: Any user-type account name. If the name contains a space, put the name in quotation marks.

direction
: Restricts the direction of the mapping. By default, mappings are bidirectional. The three valid direction symbols are as follows: "=>" means NT to UNIX mapping only; "<=" means UNIX to NT mapping only; "==" means bidirectional mapping (use this to explicitly indicate a bidirectional mapping).

The **usermap.cfg** file format uses the following symbol conventions. An asterisk (*) matches any name. The null string ("") matches no name and rejects any user. You can use either spaces or tabs as separators.

Windows NT names are case-insensitive and can contain nonASCII characters within the character set in the current code page. Windows NT user names can contain spaces, in which case you must enclose the name in quotation marks. UNIX user names are case-sensitive and must be in ASCII.

This manpage is not encyclopedic. Please refer to online documentation and the System Administrator’s Guide for further information.

EXAMPLES

The following **usermap.cfg** file ...

"Bob Garg" == bobg
mktg\Roy => nobody
engr\Tom => ""
uguest <= *
***\root => ""**

maps NT user *Bob Garg* to UNIX user *bobg* and vice versa,

allows *mktg\Roy* to login, but only with the privileges of UNIX user *nobody*,

disallows login by NT user *engr\Tom*,

maps all other UNIX names to NT user *uguest*,

and disallows NT logins using the name *root* from all domains.

zoneinfo

NAME

na_zoneinfo - time zone information files

SYNOPSIS

/etc/zoneinfo

DESCRIPTION

The directory **/etc/zoneinfo** contains time zone information files used by the **timezone** command (see na_timezone(1)). They are in standard Unix time zone file format as described below.

The time zone information files begin with bytes reserved for future use, followed by six four-byte signed values, written in a "standard" byte order (the high-order byte of the value is written first). These values are, in order:

tzh_ttisgmtcnt
The number of GMT/local indicators stored in the file.

tzh_ttisstdcnt
The number of standard/wall indicators stored in the file.

tzh_leapcnt
The number of leap seconds for which data is stored in the file.

tzh_timecnt
The number of "transition times" for which data is stored in the file.

tzh_typecnt
The number of "local time types" for which data is stored in the file (must not be zero).

tzh_charcnt
The number of characters of "time zone abbreviation strings" stored in the file.

The above header is followed by **tzh_timecnt** four-byte signed values, sorted in ascending order. These values are written in "standard" byte order. Each is used as a transition time at which the rules for computing local time change. Next come **tzh_timecnt** one-byte unsigned values; each one tells which of the different types of "local time" types described in the file is associated with the same-indexed transition time. These values serve as indices into an array of structures that appears next in the file; these structures are written as a four-byte signed **tt_gmtoff** member in a standard byte order, followed by a one-byte signed **tt_isdst** member and a one-byte unsigned **tt_abbrind** member. In each structure, **tt_gmtoff** gives the number of seconds to be added to GMT, **tt_isdst** tells whether this time is during a Daylight Savings Time period and **tt_abbrind** serves as an index into the array of time zone abbreviation characters that follow the structure(s) in the file.

Then there are **tzh_leapcnt** pairs of four-byte values, written in standard byte order; the first value of each pair gives the time at which a leap second occurs; the second gives the *total* number of leap seconds to be applied after the given time. The pairs of values are sorted in ascending order by time.

Then there are **tzh_ttisstdcnt** standard/wall indicators, each stored as a one-byte value; they tell whether the transition times associated with local time types were specified as standard time or wall clock time. A local time transition specified in standard time ignores any offset due to Daylight Savings Time. On the other hand, a time specified in wall clock time takes the prevailing value of Daylight Savings Time in to account.

Finally there are **tzh_ttisgmtcnt** GMT/local indicators, each stored as a one-byte value; they tell whether the transition times associated with local time types were specified as GMT or local time.

SEE ALSO

na_timezone(1)

cifs

NAME

na_cifs - Common Internet File System (CIFS) Protocol

DESCRIPTION

The node supports the **CIFS** protocol, which is documented in an Internet Engineering Task Force (IETF) InternetDraft specification titled "A Common Internet File System (CIFS/1.0) Protocol."

CIFS is a file sharing protocol intended to provide an open cross-platform mechanism for client systems to request file services from server systems over a network. it is based on the standard Server Message Block (SMB) protocol widely in use by personal computers and workstations running a wide variety of operating systems.

SEE ALSO

na_cifs_audit(1), na_cifs_help(1),

na_cifs_restart(1), na_cifs_shares(1), na_cifs_testdc(1)

RFC 1001, RFC 1002

cli

NAME

na_cli - Data ONTAP command language interperter (CLI)

DESCRIPTION

The Data ONTAP CLI is a command language interpreter that executes commands from the Data ONTAP console. You can access the console with a physical connection, through telnet, or through the Remote LAN Manager (RLM). The commands can also be executed using rsh and ssh protocols.

You can concatenate commands together on the same line by separating the commands with semi-colons, (;).

Quoting
The quoting rules in the Data ONTAP CLI are unusual. There is no **escape** character like the backslash; however there are the following special characters:

```
& (ampersand)     - unicode indicator
# (pound sign)    - comment indicator
; (semicolon)     - command separator
’ (single quote)  - parameter wrapper
" (double quote)  - parameter wrapper
  (space)         - parameter separator
  (tab)           - parameter separator
```

When special characters are part of a command argument, the argument needs to be surrounded by quotes or the character will be treated as a special character. A single quote character needs to be surrounded by double quote characters and a double quote character needs to be surrounded by single quote characters. The other special characters can be surrounded by either single or double quotes.

EXAMPLES

The following examples show quote usage:

qtree create /vol/test_vol/’qtree 1’

The qtree **qtree 1** is created.

qtree create /vol/test_vol/’qtree#1’

The qtree **qtree#1** is created.

qtree create /vol/test_vol/"qtree’1"

The qtree **qtree’1** is created.

qtree create /vol/test_vol/’hello"’"’"1

The qtree **hello"’1** is created.

cifs shares add jふxp /vol/test_vol/home

Creates a share with a Japanese character; whereas

cifs shares add "jふxp" /vol/test_vol/home

Creates the share **jふxp**.

sysconfig; version

Executes the **sysconfig** and **version** commands.

SEE ALSO

na_rshd(8), na_source(1),

dns

NAME

na_dns - Domain Name System

DESCRIPTION

Domain Name Service provides information about hosts on a network. This service has two parts: a resolver which requests information and a nameserver which provides it.

Data ONTAP supports only the resolver. When the node needs to resolve a host address, it first looks at the **/etc/nsswitch.conf** (see na_nsswitch.conf(5)) file to get the order in which various name services are to be consulted. If the name services before DNS fail in their lookup and DNS is enabled, then the DNS name server is contacted for address resolution.

DNS can be enabled on the node by running the **setup** command (see na_setup(1)) or by manually editing the configuration files as described below. If DNS is enabled by running the **setup** command, then the DNS domain name needs to be entered.

Enabling DNS without the setup command:

1. Create the **/etc/resolv.conf** file (see na_resolv.conf(5)) with up to 3 nameservers. Each line contains the keyword **nameserver** followed by the IP address of the server. For example:

```
nameserver 192.9.200.1
nameserver 192.9.201.1
nameserver 192.9.202.1
```

2. Edit the **/etc/rc** file (see na_rc(5)) to make sure that the option specifying the DNS domain name is set and the option to enable DNS is on. For example:

```
options dns.domainname mycompany.com
options dns.enable on
```

3. Reboot the node for these changes to take effect. If the above options commands are also entered from the console, the reboot can be avoided.

Enabling DNS with the setup command:

At setup time, one can choose to enable DNS when prompted to do so. **setup** then queries for the Internet addresses of up to three DNS nameservers.

VFILER CONSIDERATIONS

When run from a vfiler context, (e.g. via the **vfiler run** command), **dns** displays DNS information about the concerned vfiler.

SEE ALSO

na_resolv.conf(5), **RFC1034**, **RFC1035**

http

NAME

na_http - HyperText Transfer Protocol

DESCRIPTION

The node supports the **HTTP/1.0** protocol, which is documented in the Internet Engineering Task Force (IETF) **RFC 1945** titled "HyperText Transfer Protocol --HTTP/1.0."

HTTP is the primary Internet protocol used for transferring documents on the World Wide Web. It is a simple ASCII text request/response protocol. An HTTP request consists of a **method,** a target Web address or **URL** (Uniform Resource Locator), a protocol version identifier, and a set of **headers.** The method specifies the type of operation. For example, the **GET** method is used to retrieve a document. The **POST** method is used to submit a form. Headers contain additional information to the request in the form of simple name-value pairs. The HTTP header section is similar to Multipurpose Internet Mail Extensions (MIME).

The GET method is the most commonly used HTTP method. GET is used to retrieve a single resource, for example, an HTML document, image file, or other type of object, or part of it. By appending an **If-modified-since** header to the GET request, the document is retrieved conditionally, based on whether it has been modified since the date specified in the header.

An HTTP response consists of a protocol version identifier, a status code, a text response status line, response headers, and the contents of the requested document.

Access for http can be restricted by the **options httpd.access** command. Please see na_protocolaccess(8) for details.

EXAMPLES

The following is an example of use of the **GET** method:

```
GET http://www.somesite.com/ HTTP/1.0
If-modified-since: Fri, 31 Dec 1999 15:45:12 GMT
```

SEE ALSO

na_httpd.group(5),

na_httpd.log(5),

na_httpd.passwd(5),

na_httpstat(1), na_protocolaccess(8)

nfs

NAME

na_nfs - Network File System (NFS) Protocol

DESCRIPTION

The node supports versions 2, 3, and 4 of the **NFS** protocol, which are documented in RFC's 1094, 1813, and 3530 respectively.

NFS is a widely used file sharing protocol supported on a broad range of platforms. The protocol is designed to be stateless, allowing easy recovery in the event of server failure. Associated with the NFS protocol are two ancillary protocols, the MOUNT protocol and the NLM protocol. The MOUNT protocol provides a means of translating an initial pathname on a server to an NFS file handle which provides the initial reference for subsequent NFS protocol operations. The NLM protocol provides file locking services, which are stateful by nature, outside of the stateless NFS protocol.

NFS is supported on both TCP and UDP transports. Support for TCP and UDP is enabled by default. Either one can be disabled by setting the **nfs.tcp.enable** or **nfs.udp.enable** options using the **options** command.

SEE ALSO

na_nfsstat(1), na_exports(5),

nis

NAME

na_nis - NIS client service

DESCRIPTION

The NIS client service provides information about hosts, user passwords, user groups and netgroups on a network. In NIS terminology, each of the above is referred to as the map and the specific information being looked up is called the key. For example, the hosts map is like the **/etc/hosts** file; it provides a translation from host names to IP addresses. The NIS service typically has two parts: a client component which requests information and a name server which provides it.

Data ONTAP supports only the NIS client. When the node needs to resolve a key in a given map, it looks at the **/etc/nsswitch.conf** (see na_nsswitch.conf(5)) file to figure out the order in which the various databases should be consulted. For example, in case of the hosts map the lookup order may be **file, nis, dns**. This means that the node will first consult the **/etc/hosts** file. If the host name is not found in the local file, it will then try the NIS service. If the host name is still not found, then it will attempt a DNS lookup.

The NIS client can be enabled on the node by running the **setup** command (see na_setup(1)) or by manually editing the configuration files as described below. If NIS is enabled by running the **setup** command, then the NIS domain name needs to be entered.

Enabling NIS without the setup command:

1. Edit the **/etc/rc** file (see na_rc(5)) to make sure that the option specifying the NIS domain name is set and the option to enable NIS is on. For example:

```
options nis.domainname mycompany.com
options nis.enable on
```

2. Reboot the node for these changes to take effect. If the above options commands are also entered from the console, the reboot can be avoided. If the options are entered via the console only, they are not saved across a reboot.

Enabling NIS with the setup command:

At setup time, one can choose to enable NIS when prompted to do so. **setup** then queries for the NIS domain name.

SEE ALSO

na_resolv.conf(5), na_nsswitch.conf(5).

pcnfsd

NAME

na_pcnfsd - (PC)NFS authentication request server

DESCRIPTION

pcnfsd provides a personal computer NFS client with the authentication services. This release supports versions 1 and 2 of the PCNFSD protocol.

When **pcnfsd** receives an authentication request, it will register the user by validating the user name and password and returning the corresponding UID and primary GID pair, and the secondary group set for PCNFSD version 2.

It will look up the user in the **/etc/shadow** file, or the **passwd.adjunct** NIS map, if present, to find the user's password. It will look up the user in the **/etc/passwd** file, or the **passwd.byname** NIS map, to find the user's UID and primary GID, and to find the user's password if there is no **/etc/shadow** file or **passwd.adjunct** NIS map.

For a PCNFSD version 2 request, it will scan the **/etc/group** file, or the **group.byname** NIS map, to find all the groups of which the user is a member. It will look up the user in the **auto.home** NIS map, if NIS is enabled, to find the user's home directory; if NIS is not enabled, no home directory will be returned.

FILES

/etc/passwd
: This file should be in the format used on many flavors of UNIX (SunOS 4.x and later, 4.4BSD, System V Release 4 and later, and others).

/etc/group
: This file should be in the format used on many flavors of UNIX (SunOS 4.x and later, 4.4BSD, System V Release 4 and later, and others).

/etc/shadow
: This file should be in the format used on many flavors of UNIX (SunOS 5.x and later, System V Release 4 and later, and others).

SEE ALSO

na_nis(8)

BUGS

When the call fails, **pcnfsd** doesn't fake by setting the UID and the GID to acceptable values. Passwords that have been encrypted using Kerberos are not supported.

protocolaccess

NAME

na_protocolaccess - Describes protocol access control

DESCRIPTION

Protocol access control defines a method to restrict access to the node on a protocol-by-protocol basis. For example, the command **options rsh.access host=admin** restricts access to rsh to a host named admin. Access can be restricted by host name, IP address, and/or network interface name.

USAGE

The syntax is as follows:

options *protocol.access access_spec* [**AND** | **OR** [**(**] *access_spec* **[ )**] ...]

protocol is currently one of the following: rsh, telnet, ssh, httpd, httpd.admin, snmp, ndmpd, snapmirror, or snapvault.

access_spec is composed of keywords and their values. Currently the following keywords and values are defined:

host [=|!=] *host spec*
netgroup [=|!=] *netgroup spec*
if [=|!=] *network interface spec* **all**
none
legacy
*****

host spec is a comma separated list consisting of either a host name, an IP address, or an IP address with a netmask. Valid host name is a string and cannot contain the following characters: "=", "(", ")", "!", "*", and "," . An IP address is of the format *aa.bb.cc.dd* . If the IP address contains a netmask, then the format is: *aa.bb.cc.dd/mm* where mm represents the number of bits from the left.

network interface spec is a comma separated list of one or more network interface names. Valid network interface names can be obtained from the **ifconfig -a** command.

netgroup spec is a comma separated list consisting of names of one or more netgroups(group of hosts).

The access specs may be and'ed and or'ed by the keywords **AND** and **OR** respectively. The keywords **AND** and **OR** are not case-sensitive.

Operational precedence is from left to right. Parentheses may be used to force operational order.

The keyword *all* is used to allow access to all. The keyword *none* is used to allow access to none. The *legacy* keyword is used to specify previous behavior. For example, the legacy behavior of telnet is to use trusted.hosts, while the legacy behavior of rsh is to allow all.

The *access spec* can be a "*" which matches all. This is the same as the *all* keyword. If the *access spec* is a "-", then all access is denied. This is the same as the *none* keyword.

The difference between setting the host value to an IP address or a host name becomes apparent when the matching occurs. IP addresses are matched before the connection is made. If access is denied, the connection is not made and the client times out. Therefore, specifiyng the IP address lessens the impact of denial of service attacks. Host names are matched after the connection is made, and therefore the client is informed that access is denied.

If **httpd.admin.access** is not set to *legacy*, then **trusted.hosts** is ignored for httpd.admin. If **telnet.access** is not set to *legacy*, then **trusted.hosts** is ignored for telnet. If **snapmirror.access** is not set to *legacy*, then the **/etc/snapmirror.allow** file is ignored for snapmirror destination checking.

EXAMPLES

Here are some protocol access control examples:

Allow an NDMP server to accept control connection request from any client.

options ndmpd.access legacy

Allow remote shell access for only one host named gnesha.zo.

options rsh.access "host = gnesha.zo"

Allow access for Telnet subnet 10.42.69.

options telnet.access host=10.42.69.1/24

Allow ssh access for hosts abc and xyz when on network interface e0.

options ssh.access "host=abc,xyz AND if=e0"

Allow access to SNMP for network interfaces e0, e1, and e2.

options snmp.access if=e0,e1,e2

Do not allow access to HTTPD for network interface e3.

options httpd.access "if != e3"

Allow access to administrative HTTPD from for two hosts.

options httpd.admin.access host=champagne,tequilla

Disallow all access to Telnet.

options telnet.access "host=-"

Set httpd.admin to use previous trusted.hosts access

options httpd.admin.access legacy

Point SnapMirror to the (deprecated) **/etc/snapmirror.allow** file to check access to sources from other nodes.

options snapmirror.access legacy

Allow a SnapVault server to accept any client requests.

options snapvault.access all

Allow telnet access for all hosts in the netgroups admin_hosts and it_hosts. Both netgroups admin_hosts and it_hosts are defined in /etc/netgroup.

options telnet.access "netgroup = admin_hosts,it_hosts"

Allow telnet access for all hosts except those in the netgroup admin_hosts. Netgroup admin_hosts is defined in /etc/netgroup.

options telnet.access "netgroup != admin_hosts"

Note: quotes are needed around access specifications that include blanks.

SEE ALSO

na_snmpd(8), na_netgroup(5)

rlmaccess

NAME

na_rlmaccess - Describes SSH access control to the RLM.

DESCRIPTION

The access control functionality for the Remote LAN Module (RLM) provides a method to restrict SSH access to the RLM.

USAGE

The syntax is as follows:

options rlm.ssh.access host_spec

host_spec is defined as:

host[=|!=]host_list
all
none
*****

host_list is a comma-separated list consisting of either a host name, an IP address, or an IP address with a netmask. A valid host name is a string and cannot contain the following characters: "=", "(", ")", "!", "*", and ",". The IP address can be either an IPv4 address or IPv6 address. An IPv4 address is of the format aaa.bbb.ccc.ddd. If the IP address contains a netmask, then the format is: aaa.bbb.ccc.ddd/mm where mm represents the number of bits from the left. An IPv6 address is of the format aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh. If the IPv6 address contains a prefixlen, then the format is: aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh/mm where mm represents the number of bits from the left.

The default value is * - everyone is allowed access to the RLM.

The keyword **all** is used to grant access to all hosts. The keyword **none** is used to allow access to none(SSH connections cannot be made to the RLM).

The host_spec can be a "*" which matches all. This is the same as the **all** keyword. If the host_spec is a "-", then all access is denied. This is the same as the **none** keyword.

EXAMPLES

Here are some RLM SSH access control examples:

Granting RLM SSH access to only one IP address, 10.42.69.20.

options rlm.ssh.access host=10.42.69.20

Granting RLM SSH access to all hosts with prefix matching 3FFE:81D0:107:2082.

options rlm.ssh.access host=3FFE:81D0:107:2082::1/64

Disallow all access to the RLM.

options rlm.ssh.access none

Granting RLM SSH access to only two hosts, identified by their host names.

options rlm.ssh.access host=champagne,tequilla

Granting RLM SSH access to any hosts in the 10.42.69.0 subnet.

options rlm.ssh.access host=10.42.69.1/24

Allowing all IP addresses and hosts to access the RLM via SSH.

options rlm.ssh.access all

SEE ALSO

na_options(1),

rmt

NAME

na_rmt - Remote magtape protocol module

SYNOPSIS

/etc/rmt

DESCRIPTION

/etc/rmt is a special command that can be used by remote computers to manipulate a magnetic tape drive over a network connection; for example, the UNIX **dump** and **restore** commands often can either use **/etc/rmt** to access a remote tape, or have **rdump** and **rrestore** variants that can do so. **/etc/rmt** is normally run by the **rshd** daemon (see na_rshd(8)) as a result of a remote machine making a request to **rshd** to do so.

The **/etc/rmt** command accepts requests specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. This protocol is provided by **rmt** commands on many UNIX systems, although UNIX systems may support more commands and may give more different error codes.

All responses are in ASCII and in one of two forms. Successful commands have responses of:

Anumber\n

number is an ASCII representation of a decimal number. Unsuccessful commands are responded to with:

Eerror-number\nerror-message\n

error-number is one of:

2 (ENOENT)
The tape device specified in an open request did not have a valid syntax.

6 (ENXIO)
The tape device specified in an open request does not exist.

5 (EIO)
An I/O error occurred when performing the request.

25 (ENOTTY)
An invalid tape operation was specified in a ‘‘perform special tape operation’’ request.

error-message is a (UNIX-style) error string for the error specified by *error-number*.

The protocol is comprised of the following commands, which are sent as indicated - no spaces are supplied between the command and its arguments, or between its arguments, and **\n** indicates that a newline should be supplied:

Odevice\nmode\n
Open the specified *device* using the indicated *mode*. *device* is a tape name of the form described in na_tape(4) and *mode* is an ASCII representation of a decimal number specifying how the tape is to be opened:

0
read-only

1
write-only

2
read-write

If a device had already been opened, it is closed before a new open is performed.

Cdevice\n
Close the currently open device. The *device* specified is ignored.

Lwhence\noffset\n
Performs no operation, and returns the value of *offset*; UNIX-style **lseek** operations are ignored on NetApp node tape devices, just as they are on tape devices on many UNIX systems.

Wcount\n
Write data onto the open device. If *count* exceeds the maximum data buffer size (64 kilobytes), it is truncated to that size. **/etc/rmt** then reads *count* bytes from the connection, aborting if a premature end-offile is encountered. The response value is the number of bytes written if the write succeeds, or -1 if the write fails.

Rcount\n
Read *count* bytes of data from the open device. If *count* exceeds the maximum data buffer size (64 kilobytes), it is truncated to that size. **/etc/rmt** then attempts to read *count* bytes from the tape and responds with **Acount-read\n** if the read was successful; otherwise an error in the standard format is returned. If the read was successful, the data read is then sent.

Ioperation\ncount\n
Perform a special tape operation on the open device using the specified parameters. The parameters are interpreted as ASCII representations of the decimal values. *operation* is one of:

0
write end-of-file marker

1
forward space *count* files

2
backward space *count* files

3
forward space *count* tape blocks

4
backward space *count* tape blocks

5
rewind the tape

6
rewind and unload the tape

The return value is the *count* parameter when the operation is successful.

Any other command causes **/etc/rmt** to close the connection.

DIAGNOSTICS

All responses are of the form described above.

SEE ALSO

na_rshd(8)

rquotad

NAME

na_rquotad - remote quota server

DESCRIPTION

The node supports the remote quota service that allows NFS clients to determine their quota allocation on the server.

SEE ALSO

na_quota(1)

BUGS

The rquota protocol doesn’t support group or tree quotas.

rshd

NAME

na_rshd - remote shell daemon

DESCRIPTION

The node has UNIX-compatible remote shell capability that enables you to execute certain node commands from a UNIX command line or shell script. It also enables you to use a remote shell application on a PC to run node commands.

The value of **rsh.access** controls access to the node, and is set by **options rsh.access.** See na_protocolaccess(8) for more details. This value is checked prior to the authentication mechanisms discussed below.

The **/etc/hosts.equiv** file controls authentication to the node remote shell. The hosts and users (on those hosts) listed in the **/etc/hosts.equiv** file are automatically authenticated. This means that the node accepts remote shell commands via rsh from these hosts and users.

An alternative authentication mechanism for rshd is to have the client use rsh with a *-l* option that specifies the admin_name and password in the form of *-l admin_name:password*. Both the admin_name and password are created with the node's **useradmin** command.

EXAMPLE

The following example shows how to run the **version** command from a trusted host named *adminhost* through a remote shell:

adminhost% **rsh -l root toaster version**

The following example shows how to run the **sysconfig -r** command with a password *rpass42* from an untrusted host named *ahost* through a remote shell:

ahost% **rsh -l root:rpass42 toaster sysconfig -r**

To see a list of node commands that can be executed, enter:

adminhost% **rsh -l root toaster "?"**

SEE ALSO

na_options(1), na_protocolaccess(8)

snmpd

NAME

na_snmpd - snmp agent daemon

DESCRIPTION

The node supports an SNMP version 1 (RFC 1157) compatible agent that provides support for both the MIB-II (RFC 1213) management information base for TCP/IP based internets as well as a Data ONTAP Custom MIB.

A number of user configurable options for the SNMP agent can be set and queried from the console using the **snmp** command (see na_snmp(1)).

Due to weak authentication in SNMP version 1, SetRequest commands that allow the remote setting of configuration variables have been disabled.

Access for snmp can be restricted by the **options snmp.access** command. Please see na_protocolaccess(8) for details.

MIB-II

Under MIB-II, information is accessible for the **system, interfaces, at, ip, icmp, tcp, udp** and **snmp** MIB-II groups. The transmission and egp groups are not supported.

The **coldStart, linkDown, linkUp** and **authenticationFailure** traps are implemented. Traps are configured using the **snmp** command.

DATA ONTAP CUSTOM MIB

The Data ONTAP Custom MIB provides a means to obtain detailed information about many aspects of node operation via SNMP. The Custom MIB can be obtained from Network Appliance's FTP site at **ftp://ftp.netapp.com/pub/netapp/mib/netapp.mib**, or by requesting the MIB on a floppy disk from Network Appliance Technical Support.

The following is a summary of the top-level groups in the Custom MIB and the information they contain:

product
Product-level information such as the software version string and system ID.

sysStat
System-level statistics such as CPU uptime, idle time and aggregate kilobytes received and transmitted on all network interfaces.

nfs
Statistics like those displayed by the **nfsstat** command (see na_nfsstat(1)), including statistics for each client if perclient NFS statistics have been enabled using the **nfs.per_client_stats.enable** option (see na_options(1)). The per-client NFS statistics are indexed by client IP addresses.

quota
Information related to disk quotas, including the output of the quota report command (see na_quota(1)). To access quota information, quotas must be turned on.

filesys
Information related to the file system, including the equivalent of the **maxfiles** and **df** commands, and some of the information from the **snap list** command (see na_df(1), na_snap(1)).

raid
Information on RAID equivalent to the output of the **sysconfig -r** command (see na_sysconfig(1)).

HA CONSIDERATIONS

In takeover mode, SNMP agents can continue to access the MIBs on both nodes in an HA pair. However, the counters reported by SNMP are combined counters from both nodes. For example, in takeover mode, the SNMP agent can report the number of packets sent or received by both nodes, but you cannot determine from the number how many packets are sent or received on each node.

You can have an application on the network management station set an alarm when a node has been taken over. The SNMP variable to check is the **netapp.netapp1.sysStat.cf.cfSettings** variable. If this variable is set to **thisNodeDead**, the node has been taken over.

SEE ALSO

na_options(1), na_snmp(1), na_protocolaccess(8)

spaccess

NAME

na_spaccess - Describes SSH access control to the SP.

DESCRIPTION

The access control functionality for the Service Processor (SP) provides a method to restrict SSH access to the SP.

USAGE

The syntax is as follows:

options sp.ssh.access host_spec

host_spec is defined as:

host[=|!=]host_list
all
none
*****

host_list is a comma-separated list consisting of either a host name, an IP address, or an IP address with a netmask. A valid host name is a string and cannot contain the following characters: "=", "(", ")", "!", "*", and ",". The IP address can be either an IPv4 address or IPv6 address. An IPv4 address is of the format aaa.bbb.ccc.ddd. If the IP address contains a netmask, then the format is: aaa.bbb.ccc.ddd/mm where mm represents the number of bits from the left. An IPv6 address is of the format aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh. If the IPv6 address contains a prefixlen, then the format is: aaaa:bbbb:cccc:dddd:eeee:ffff:gggg:hhhh/mm where mm represents the number of bits from the left.

The default value is * - everyone is allowed access to the SP.

The keyword **all** is used to grant access to all hosts. The keyword **none** is used to allow access to none(SSH connections cannot be made to the SP).

The host_spec can be a "*" which matches all. This is the same as the **all** keyword. If the host_spec is a "-", then all access is denied. This is the same as the **none** keyword.

EXAMPLES

Here are some SP SSH access control examples:

Granting SP SSH access to only one IP address, 10.42.69.20.

options sp.ssh.access host=10.42.69.20

Granting SP SSH access to all hosts with prefix matching 3FFE:81D0:107:2082.

options sp.ssh.access host=3FFE:81D0:107:2082::1/64

Disallow all access to the SP.

options sp.ssh.access none

Granting SP SSH access to only two hosts, identified by their host names.

options sp.ssh.access host=champagne,tequilla

Granting SP SSH access to any hosts in the 10.42.69.0 subnet.

options sp.ssh.access host=10.42.69.1/24

Allowing all IP addresses and hosts to access the SP via SSH.

options sp.ssh.access all

SEE ALSO

na_options(1),

syslogd

NAME

na_syslogd - Logs system messages.

DESCRIPTION

The **syslogd** daemon logs system messages to the console, log files and other remote systems as specified by its configuration file, **/etc/syslog.conf**. The **syslogd** daemon reads its configuration file when it starts up during the boot procedure, or within 30 seconds after the **/etc/syslog.conf** file is modified. For information about the format of the configuration file, see na_syslog.conf(5).

If **/etc/syslog.conf** does not exist the **syslogd** daemon will output all log messages of priority **info** or higher to the console and to the file **/etc/messages**. To prevent **/etc/messages** from getting too large, the **syslogd** daemon will rotate the contents of **/etc/messages** through the files **/etc/messages.0** through **/etc/messages.5**. This rotation is done once a week. So the log messages of the current week will be saved in the file **/etc/messages** and the message logs of the six weeks prior to that are saved in the files **/etc/messages.0** through **/etc/messages.5**.

To prevent large numbers of repeated messages being logged, the **syslogd** daemon will follow the first instance of a repeated message with the number of times the message was repeated. If a message is repeated over a long time period, the **syslogd** daemon will wait for increasingly longer intervals before logging the number of repeats. The repeat notification interval starts at 30 seconds and moves quickly to 20 minutes.

FILES

/etc/syslog.conf
The configuration file. **/etc/syslog.conf.sample** A sample configuration file.

/etc/messages
Message log file for current week.

/etc/messages.[0-5]
Message log for prior weeks.

HA CONSIDERATIONS

In takeover mode, the failed node logs syslog messages to its own **/etc/messages** file and to the **/etc/messages** file on the live node. The live node logs its syslog messages only to its own **/etc/messages** file.

Because the **/etc/messages** file on the live node contains syslog messages from two nodes, the node uses node names in the syslog messages to indicate the node from which the syslog message originated.

For example, if **toaster1** takes over **toaster2**, a message from **toaster2** is logged to the **/etc/messages** file on **toaster1**, and the message can be similar to the following:

```
Wed May  6 18:57:52 GMT [toaster2/toaster1]: raid_disk_admin]: Volume vol7 has been added to the system.
```

If the name of the failed node is unknown, the string ‘‘partner’’ is printed instead of a node name.

SEE ALSO

na_syslog.conf(5)