StorNext 7 Man Pages Reference Guide

6-68799-01, Rev. D

Quantum

StorNext 7 Man Pages Reference Guide
6-68799-01 Rev. D
December 2022

Introduction

Quantum recommends using the GUI to complete most StorNext tasks, but there might be situations where you prefer or need to use the command line interface (CLI) instead.

This document is a printed version of the man pages as they currently exist in StorNext.

For each command, information similar to the following is provided:

- name
- synopsis
- description
- options
- examples
- see also

Note: Storage Manager commands are only available on Linux MDC systems.

Note: On a Windows Vista system, when you run applications from the command line that require administrative privileges (such as those provided by **cygwin**), you can start the CLI application either from an elevated shell environment or a DOS shell. If you do not, CLI commands requiring administrative privileges will fail and you will receive an error message indicating that you do not have sufficient privileges to run the command.

To start the shell in elevated mode, right-click the icon for Command Prompt or Cygwin and select Administrative Mode.

Using Commands or Viewing man Pages

The **man** pages are contained within this document, and can also be accessed via the command line.

Use the following procedure to view the **man** page for a command.

1. Source the StorNext profile. Do one of the following:
   - If using the **bash** shell, at the command prompt, type:

     source /usr/adic/.profile
   - If using the **csh** or **tcsh** shell, at the command prompt, type:

     source /usr/adic/.cshrc
2. View the **man** page for a command. At the command prompt, type:

man <command>

where **<command>** is the command for which you want to view the **man** page.

3. Press **<Spacebar>** to page through the **man** page.
4. When you are finished, type **q** and press **<Enter>** to exit the **man** page.

Paging through **man** pages may work differently depending on the viewer specified by the **$PAGER** environmental variable.

Contacting Quantum

More information about StorNext is available on the Quantum Service and Support website at http://www.quantum.com/ServiceandSupport. The Quantum Service and Support website contains a collection of information, including answers to frequently asked questions (FAQs).

StorNext Appliance Upgrades

To request a StorNext software upgrade for non-Quantum MDCs , visit http://www.quantum.com/ServiceandSupport/Upgrade/Index.aspx. To request a StorNext software upgrade for StorNext Appliances, open a support ticket at: www.quantum.com/customercenter/. For further assistance, or if training is desired, contact the Quantum Technical Assistance Center.

Contacts

Quantum company contacts are listed below.

Quantum Home Page

Visit the Quantum home page at:

http://www.quantum.com

Comments

To provide comments or feedback about this document, or about other Quantum technical publications, send e-mail to:

doc-comments@quantum.com

Getting More Information or Help

StorageCare™, Quantum's comprehensive service approach, leverages advanced data access and diagnostics technologies with cross-environment, multi-vendor expertise to resolve backup issues faster and at lower cost.

Accelerate service issue resolution with these exclusive Quantum StorageCare services:

Quantum. Global Services

- **Service and Support Website** - Register products, license software, browse Quantum Learning courses, check backup software and operating system support, and locate manuals, FAQs, firmware downloads, product updates and more in one convenient location. Benefit today at:

  http://www.quantum.com/ServiceandSupport/Index.aspx

- **eSupport** - Submit online service requests, update contact information, add attachments, and receive status updates via email. Online Service accounts are free from Quantum. That account can also be used to access Quantum's Knowledge Base, a comprehensive repository of product support information. Sign up today at:

  www.quantum.com/customercenter/

For further assistance, or for training opportunities, contact the Quantum Customer Support Center:

United States	1-800-284-5101 (toll free) +1-720-249-5700
EMEA	+800-7826-8888 (toll free) +49-6131-3241-1164
APAC	+800-7826-8887 (toll free) +603-7953-3010
Other numbers available at: www.quantum.com/serviceandsupport/get-help/index.aspx#contact-support Worldwide support: http://www.quantum.com/ServiceandSupport/Index.aspx	

Worldwide End-User Product Warranty

For more information on the Quantum Worldwide End-User Standard Limited Product Warranty:

http://www.quantum.com/serviceandsupport/warrantyinformation/index.aspx

Table of Contents

StorNext Filesystem Commands

StorNext Storage Manager Commands

NAME

access.json − StorNext File System Access Control

SYNOPSIS

Linux only.

/usr/cvfs/config/access.json

DESCRIPTION

access.json file is an optional StorNext configuration file used to control access to the file system using the **snfs_access(7)** feature. The file consists of a brief header followed by one or more rules that specify whether a host (or set of hosts) is allowed access as well as the sub-tree and time of week the rule applies to. Most fields in the file have defaults and the absense of a field implies the default value allowing for a more compact file. Note that while this man-page describes the file's syntax, it is strongly recommended that the **snaccess(8)** program or a graphical user interface be used instead of editing the **access.json** file directly as any syntax errors will prevent the feature from working properly. It is very easy to forget to add or remove commas in JSON files.

HEADER

The header of the **access.json** file always contains a version value such as the following:

"version": "1.0"

In addition, the header may also include a boolean called "suspended" that indicates whether the **snfs_access(7)** feature is active. Setting "suspended" to true allows the feature to be tempoarily disabled without having to delete the **access.json** file. The default value of "suspended" is false.

"suspended": false

RULE LIST

The list of rules follows the header and is a JSON array named "access_list" Note that when access is attempted, more than one rule may apply and the last matching rule determines whether access is allowed or denied. If no rules match, acccess is allowed.

Each rule in the array is an entry consisting of the following fields.

type Either "allow" or "deny" This is a required field.

host The IP address of the system the rule applies to with an optional subnet mask denoted by a "/" and an integer. Examples include "10.101.224.1" and "10.101.224.0/24" Note that host values should correspond to metadata network. If **host** is omittied, the rule applies to all hosts.

file_systems
A JSON array consisting of the set of file systems the rule applies to. If **file_systems** is omitted, the rule applies to all file systems. Care should be taken to ensure that file system names are spelled correctly.

path A path relative to root that the rule applies to. This path must not start with a "/" Also, the path must not include the mount point. For example, if the full path is /stornext/snfs1/projects/current and the mount point is /stornext/snfs1, "projects/current" should be specified. When specifying a path, care should be taken to avoid typos and that the directory exists as the tool may not catch errors.

start_time
The time of day in 24-hour HH:MM format that determines when the rule begins to apply. If omitted, 00:00 is assumed (i.e. the rule applies at the very beginning of the day).

end_time
The time of day in 24-hour HH:MM format that determines when the rule no longer applies. If omitted, 24:00 is assumed (i.e. the rule applies until midnight).

days A JSON array consisting of the days of the week the rule applies to. if **days** is omitted, the rule applies every day.

enabled

A boolean that determine whether the rule is active. Setting this to false is a way to "comment out" an individual rule that you may way wish to use later. The default is true. As previously mentioned, the entire snfs_access(7) feature can be temporarily disabled without removing the **access.json** file by setting "suspended" to true in the header.

EXAMPLES

As mentioned previously, **snfs_access(7)** feature allows access by default. One simple way to reverse this logic is to have the first rule deny access to all hosts. Then, allowed hosts can be added in subsequent rules. For example:

```
{
    "version": "1.0",
    "access_list": [
        {
            "type": "deny"
        },
        {
            "type": "allow",
            "host": "10.65.166.51"
        },
        {
            "type": "allow",
            "host": "10.65.162.37"
        },
        {
            "type": "allow",
            "host": "10.65.180.188"
        }
    ]
}
```

If a **access.json** file is permissive (i.e. does not have an initial "deny" rule) and a host needs to have its access limited, a deny rule for that host will be needed. As a trivial example, the following **snfs_access(7)** allows access by all hosts at any time, except that the host "10.65.166.51" may only access the file systems "snfs1" and "snfs2":

```
{
    "version": "1.0",
    "access_list": [
        {
            "type": "deny",
            "host": "10.65.166.51"
        },
        {
            "type": "allow",
            "host": "10.65.166.51",
            "file_systems": [
                "snfs1",
                "snfs2"
            ]
        }
    ]
```

```
}
```

In most cases, when specifying a path, it only makes sense to also specify the file system to which it belongs as there usually is only one. For example, the following restricts access by "10.65.166.51" to the path "projects/current" on the file system "snfs1":

```
{
    "version": "1.0",
    "access_list": [
        {
            "type": "deny",
            "host": "10.65.166.51"
        },
        {
            "type": "allow",
            "host": "10.65.166.51",
            "path": "projects/current"
            "file_systems": [
                "snfs1"
            ]
        }
    ]
}
```

NOTES

If a file system is already mounted and a rule revokes access, this can cause unsaved changes to be lost without warning. Therefore, care must be taken, especially when using day and time ranges to restrict access.

FILES

/usr/cvfs/config/access.json

SEE ALSO

snfs_access(7) **snaccess**(8)

NAME

awsregions.json − StorNext Amazon Web Services (AWS) regions configuration file

SYNOPSIS

/usr/cvfs/config/awsregions.json

DESCRIPTION

The StorNext (**SNFS**) *awsregions.json* file is used to define new endpoints and region names supported by the Amazon Simple Storage Service and the Amazon Web Services Security Token Service.

Modifications to this file are necessary only if you require access to a bucket created in a region not listed in the *awsregions.json.template* file or require the use of a regional Security Token Service endpoint not listed in that file.

SYNTAX

To allow access to a new region, refer to the Amazon Web Services documentation for a list of valid endpoints and their region names. For each new endpoint to be defined, add an entry to the awsregions.json file. The entry should be of the format:

```
{
        "region":<region-value>,
        "endpoint":<endpoint-value>
}
```

where both **<region-value>** and **<endpoint-value>** are JSON strings.

Note, buckets in the newly specified region will not be accessible via the Storage Manager until after the next Storage Manager restart.

FILES

/usr/cvfs/config/awsregions.json

NAME

/usr/adic/util/cnvt2ha.sh − Conversion script for StorNext High Availability Servers (Linux only)

SYNOPSIS

cnvt2ha.sh primary

cnvt2ha.sh secondary *shared_FS peer_IP_addr*

DESCRIPTION

StorNext High Availability (HA) Server configurations that include Storage Manager (SNSM) software must have their operational data moved to a shared location. The *cnvt2ha.sh* script performs this function. It runs in different modes for the first server to be converted (Primary) versus all subsequent server conversions (Secondary). All file systems should be configured before starting the Primary conversion. The Secondary conversion is done on a fresh installation of SNSM with no file systems configured.

The *cnvt2ha.sh* script status and operations are written to the log file */usr/adic/HA/cnvt2ha.sh.log*.

Before starting conversion of the **Primary** server, the following steps must be completed:

/usr/cvfs/config/.cfgx*
: Every file-system configuration must contain one of the **HaFsType** configuration items (see **snfs_config**(5)). One and only one of them must be **HaShared** type.

SNSM Features
: Storage Manager elements should be configured including policies, libraries etc.

/usr/cvfs/config/fsmlist
: Lists all the configured CVFS file systems to be started. See **fsmlist**(4).

/usr/cvfs/config/fsnameservers
: It is recommended, but not required, that the HA Cluster Servers be the nameservers. The only requirement is that at least one nameserver is available when either of the cluster servers is powered down. See **fsnameservers**(4).

/usr/cvfs/config/ha_peer
: Contains the IPv4 or IPv6 numerical address of the peer server. See **ha_peer**(4).

/usr/cvfs/config/license.dat
: Licenses for both servers must be in this file.

User and Group IDs
: The users **tdlm** and **www**, and the group **adic** must exist on both servers with the same ID numbers.

Synchronized System Clocks
: This recommendation is to aid in log-file analysis.

No Processes in File Systems
: The CVFS file systems will be unmounted during conversion.

Before starting conversion of the **Secondary** server, the following steps must be completed:

/usr/cvfs/config/fsnameservers
: Configure the */usr/cvfs/config/fsnameservers* file identical to the Primary server. This will allow the Secondary server to mount the shared file system and copy in all the other configuration data.

OPTIONS

primary
: Perform Primary server conversion.

secondary
: Perform Secondary server conversion.

shared_FS
Name of the shared file system.

peer_IP_addr
The numerical IPv4 or IPv6 address of the Primary server.

FILES

/usr/adic/util/cnvt2ha.sh
/usr/adic/HA/cnvt2ha.sh.log
*/usr/adic/HAM/shared/**
*/usr/adic/HAM/shared/mirror/**

ENVIRONMENT VARIABLES

SNSM_HA_CONFIGURED

SEE ALSO

mount(8), **snhamgr**(8), **fsmlist**(4), **vfstab**(4), **fstab**(5), **snfs_config**(5), **init.d**(7), **chkconfig**(8)

NAME

cvadmin − Administer a StorNext File System

SYNOPSIS

cvadmin [**−H** *FSMHostName*] [**−F** *FileSystemName*] [**−M**] [**−n**] [**−x**] [**−f** *filename*]
[**−e** *command1* **-e** *command2*...]

DESCRIPTION

cvadmin is an interactive command used for general purpose administration of a StorNext File System including:

1. displaying file system and client status
2. activating a file system currently in stand-by mode
3. viewing stripe group attributes
4. enabling File System Manager (FSM) tracing
5. displaying disk and path information for the local system
6. forcing FSM failover
7. fetching FSM usage and performance statistics
8. temporarily enabling or disabling global file locking
9. generating a report of open files
10. listing currently held file locks
11. starting, restarting and stopping of daemon processes
12. resetting RPL information

OPTIONS

Invoke **cvadmin** to start the interactive session and list the running File System Managers (FSMs). (Note: StorNext system services must be started prior to running **cvadmin**. In particular, the local **fsmpm**(8) process must be active.)

The initial output shows the running FSMs grouped by the cluster in which they are located. The hostnames or IP addresses are those that are advertised to clients. When more than one location is listed, one or more these will be marked as preferred. This means that this address is on the metadata network and should be used, if possible, by a client to make a connection to the FSM. The remaining addresses can be used if a connection can not be made using a preferred address.

Note: An administrator can filter out addresses that are not to be advertised by using the **snfs_metadata_network_filter.json**(5) configuration file.

Then (optionally) use the **select** command described below to pick an FSM to connect to. Once connected, the command will display basic information about the selected file system and prompt for further commands.

Note that a few commands such as **paths**, **disks**, **start**, and **stop** require information obtained from the local **fsmpm**(8) only, so there is is no need to select an FSM prior to using them.

USAGE

−H *FSMHostName*
: Connect to the FSM located on the machine FSMHostName. By default cvadmin will attempt to connect to an FSM located on the local machine.

−F *FileSystemName*
: Automatically set the file system *FileSystemName* as the active file system in cvadmin.

-M
: When listing file systems with the **select** command, display [managed] next to each file system with DataMigration enabled. This option is currently only intended for use by support personnel.

−f *filename*
Read commands from *filename*

−e *command*
Execute command(s) and exit

−n Show numerical addresses instead of trying to determine symbolic host

−x Enable extended commands.

COMMANDS

The **cvadmin** commands can be used to display and modify the SNFS active configuration. When a modification is made, it exists only as long as the **FSM** is running. More permanent changes can be made in the configuration file. Refer to the **snfs_config**(5) man page for details. The following commands are supported.

The syntax for *file_system_name* is: *name*[**@***cluster*[*/addom*]]

activate *file_system_name* [*hostname_or_IP_address*]
Activate a file system *file_system_name*. This command may cause an FSM to activate. If the FSM is already active, no action is taken.

activate *file_system_name number_of_votes*
Quantum Internal only. Bypass the election system and attempt to activate the fsm on this node.

cluster select [*N*|[+]*cluster_name*]
Select a cluster to be used by other commands when the cluster name isn't explicitly given with the FSM name. Normally only a known cluster name can be selected, but preceding it with a '+' will force an unknown cluster name to be used. A cluster name of **none** or **0** will de-select the currently selected cluster name. Using no arguments or just **select** will show all known clusters. *Cluster_name* is [**@**]*cluster*[*/addom*].

cluster filter [*N*|*cluster_name*]
Set a filter for the output of the **select** command. When a cluster filter is set, only FSMs in that cluster will be displayed by **select**. The filter can be set to all the clusters in an administrative domain by using A cluster name of **none** or **0** will disable the filter. Using no arguments or just **filter** will show all known clusters. *Cluster_name* is [**@**]*cluster*[*/addom*].

coord [[**on**] *hostname_or_IP_address*]
Display the list of NSS coordinators on the local host or on the specified host.

debug [[+|-] *flag* [...]]
View or set the File System Manager's debugging flags. Entering the command with no *flag* will return current settings, the location of the **FSM** log file and a legend describing what each setting does. By entering the command with a *flag* list, the **FSM** Debugging Flags will be set accordingly. Each *flag* can be either a name or numeric value. Names will be mapped to their numeric value, and may be abbreviated as long as they remain unique. Numeric values are specified using a standard decimal or hexadecimal (0x) value of up to 32 bits. Using '+' or '-' enables ('+') or disables ('-') only the selected flags, leaving all other flags unchanged.

NOTE - Setting Debugging Flags will severely impact the **FSM**'s performance! Do this only when directed by a Quantum specialist.

discon Forcefully disconnect a client from the FSM. With no arguments, **discon** displays the list of connected clients. When supplied an index, **discon** will immediately disconnect the specified client. Only use when recommended by Technical Support. Also see the "repblocked" command.

disks [**refresh**]
Display the StorNext disk volumes local to the system that cvadmin is attached to. Using the optional **refresh** argument will force the fsmpm to re-scan all volumes before responding. If the fsmpm's view of the disks in any file system changes compared with the FSM's view of that client's disks as a result of the refresh, a disconnect and reconnect to the FSM will take place to resynchronise the file system state.

disks [**refresh**] *fsm*
Display the StorNext meta-data disk volumes in use by the *fsm*. If the optional **refresh** argument is used, additional paths to these volumes may be added by the fsm.

fail *[file_system_name|index_number]*
Initiate an FSM Failover of file system *file_system_name*. This command may cause a stand-by FSM to activate. If an FSM is already active, the FSM will shut down. A stand-by FSM will then take over. If a stand-by FSM is not available the primary FSM will reactivate after failover processing is complete.

files Report counts of files, directories, symlinks and other objects which are anchored by a user type inode. These include named streams, block and character device files, fifos or pipes and named sockets. If the file system is undergoing conversion to StorNext 5.0, conversion progress is displayed and counters reflect the count of converted objects.

fsmlist [*file_system_name*] [**on** *hostname_or_IP_address*]
Display the state of FSM processes, running or not. Optionally specify a single *file_system_name* to display. Optionally specify the host name or IP address of the system on which to list the FSM processes.

filelocks
Query cluster-wide file/record lock enforcement. Currently cluster-wide file locks are automatically used on Unix. Windows file/record locks are optional.

If enabled, byte-range file locks are coordinated through the FSM, allowing a lock set by one client to block overlapping locks by other clients. If disabled, then byte-range locks are local to a client and do not prevent other clients from getting byte-range locks on a file, however they do prevent overlapping lock attempts on the same client.

help (**?**)
The **help** or **?** command will display a command usage summary.

latency-test [*index_number*|**all**] [*seconds*]
Run an I/O latency test between the FSM process and one client or all clients. The default test duration is 2 seconds.

metadata
Report metadata usage. Also provide an estimate on the value of bufferCacheSize that will allow all metadata to be cached.

mdarchive [*status*|*backup*|*rebuild*|*prune*]
Run commands targeting the metadata archive associated with the selected FSM. Running mdarchive without command arguments will display the full path location of the metadata archive.

mdarchive status
The *status* command prints the progress of the current metadata archive activity, if any. If generating a new metadata archive or restoring an existing one, the percentage complete will be displayed. Otherwise, the current update backlog is displayed.

mdarchive rebuild [new]
If metadataArchiveDays is set to zero then the *rebuild* command will force the FSM to discard the existing metadata archive and generate a new one, otherwise the FSM will rebuild the existing metadata archive in a way that will preserve history across the rebuild.

Rebuilds can be performed online while clients are active in the file system.

When metadataArchiveDays is set to a value greater than zero, requesting a *new* rebuild will cause the FSM to delete the current metadata archive before starting the rebuild. All metadata history that existed prior to the new rebuild will be lost but the rebuild will shrink the mdarchive down to its minimum size. Requesting a *new* rebuild has no effect when metadataArchiveDays is set to zero because the existing metadata archive is always deleted before starting the rebuild in that case.

WARNING: When using the *new* argument to the rebuild command, all metadata history that existed prior to the rebuild will be lost. This will prevent external applications like snaudit and snhistory from reporting events that occurred prior to the rebuild and will prevent snrecover from recovering files that were deleted prior to the rebuild.

mdarchive rebuild wait
Wait for an ongoing rebuild to complete and report status periodically.

mdarchive backup [*full|partial***]** pathname
For an FSM with a current mdarchive, request a backup copy be generated as a compressed tar file. The pathname must specify an existing directory on the FSM node and include the file name to be created. A full dump contains all the content, a partial dump contains all the content changed since the last backup.

mdarchive backup wait
Wait for an ongoing mdarchive backup to complete and report progress periodically.

mdarchive prune
The *prune* command will force the removal of expired historical data from the metadata archive. This command is ignored and a message is written to the system log file if metadataArchiveDays is set to zero in the configuration file or if it is issued while an mdarchive build, file recovery, or metadata restore is in progress.

mdarchive throttle set <threshold> <delay>
Set the threshold (number of bundles) and delay (milliseconds) for throttling incoming metadata changes to allow an update backlog to dissipate. Throttling is disabled when delay is equal to 0 which is the default. In addition to using the cvadmin mdarchive command, throttling can be set at FSM startup using the MDARCHIVE_THROTTLE_THRESHOLD and MDARCHIVE_THROTTLE_DELAY environment variables. Note: this option is not intended for general use. Only enable throttling when recommended by Technical Support. Note: the throttling capability may change or be removed in a future release.

mdarchive throttle show
Display current mdarchive throttle settings. See "mdarchive throttle set" above.

paths Display the StorNext disk volumes visible to the local system. The display is grouped by *<controller>* identity, and will indicate the "Active" or "Passive" nature of the path if the Raid Controller has been recognized as configured in Active/Active mode with AVAT enabled.

proxy [**long**]
Display Disk Proxy servers and optionally display the disks they serve for this file system.

proxy who *hostname*
The "who" option displays all proxy connections for the specified host.

qbm trace [*clientid***]**
Create a QBM internal trace file on the MDC. If *clientid* is specified (where *clientid* is the client's login id), a client trace is also also generated. This command creates an

SNQBM_xxx file in the debug directory. Currently, only Windows clients generate internal traces. This command is for Quantum use only.

qbm show *stripe-group-num*
Show the QBM allocation table for a stripe group given the stripe-group index. All bandwidth (BW) is in MB/sec. The display includes the following sub-tables:

Stripe Group (SG) Table: SG max (capacity), BW currently free, the total BW in the reserve pools, and the total over-subscripted (over capacity) amount.

Priority (Pri) Table: table class name, count of clients in the priority, total bandwidth allocated at this priority, minimum BW, maximum BW, extra (over min) BW, reserved BW used/total reserve allocated to this priority, and current BW used.

Client Entry: list number, IP address, client login number, bandwidth information - minimum, maximum and current BW, restore BW, flag (l - client asked for less, m - client asked for more, g - client given more BW over minimum BW).

qbm newconfig
Re-read the <*_file_system_name_*>_qbm.conf file and update the clients.

qos Display per-stripe group QOS statistics. Per-client QoS statistics are also displayed under each qos-configured stripe group.

quit This command will disconnect **cvadmin** from the **FSM** and exit.

ras enq [nomerge] *event "detail string"*
Generate an SNFS RAS event. If "nomerge" is specified, multiple instances of the RAS event are not merged into a single ticket. For internal use only.

ras enq [nomerge] *event reporting_FRU violating_FRU "detail string"*
Generate a generic RAS event. If "nomerge" is specified, multiple instances of the RAS event are not merged into a single ticket. For internal use only.

regapigw
Register the filesystem with the fsmpm to enable registration with an api gateway. To be used when the fsmpm is not aware of this filesystem for api gateway registration purposes. This can be confirmed using the 'snrest gateway status -d' command and noting the absence of the filesystem. The fsmpm performs the api gateway registration on behalf of the filesystem. This situation should not occur, but if it does this command allows recovery from the situation without requiring a restart of the filesystem.

repblocked
Generate a report of files that may be causing open operations to block due to an unresponsive client that is slow to acknowledge a request to change state. Such clients should be inspected to determine whether there is a reason for their lack of response. One common cause of unresponsiveness is that Xsan clients have entered sleep mode. The settings on these clients should be adjusted so they never sleep. If the client is known to be hung, the connection to the FSM can be reset using the **discon** command.

repfl Generate a report that displays the file locks currently held. Note: this command is only intended for debugging purposes by support personnel. In future releases, the format of the report may change or the command may be removed entirely. Running the repfl command will write out a report file and display the output filename.

repof Generate a report that displays all files that are currently open on each StorNext client. Running the repof command will write out a report file and display the output filename containing that report.
The information displayed for each file is: the pathname, inode number, the number of "writers" (which is the # of times this client has opened the file for write -- usually 1 or

0), and stat information.
Next a line is printed showing the open state across all clients. The open state contains: the **count** of different clients with this file open, the # of those clients that are **writers**, and which **client**, if non-zero, has the file open exclusively or that owns the **ioToken**. The open state **flags** are displayed followed by a set of strings indicating the meaning of some of the bits in the **flags** field. Some examples:

flags 0x2000: shared_read
flags 0x5000: I/O token shared_excl
flags 0x5009: revoking I/O token shared_excl
flags 0x9: revoking OLD Exclusive

If **revoking** is indicated, that means the **client**(s) has been sent a message and a response is being awaited. If **client** is zero with **revoking**, a message was sent to one or more clients in **shared_read**.
See also the **repblocked** command.

In future releases, the format of the report may change.

resetrpl [**clear**]
Repopulate Reverse Path Lookup (RPL) information. The optional *clear* argument causes existing RPL data to be cleared before starting repopulation. Note: **resetrpl** is only available when cvadmin is invoked with the **-x** option. Running **resetrpl** may significantly delay FSM activation. This command is not intended for general use. Only run **resetrpl** when recommended by Technical Support.

restartd *daemon* [**once**]
Restart the *daemon* process. For internal use only.

select [*file_system_name*|*N*]
Select an active FSM to view and modify. If no argument is specified, a numbered list of FSMs and running utilities will be displayed. If there is only one active file system in the list, it will automatically be selected.

When a running utility is displayed by the select command, it will show the following information. First the name of the file system is displayed. Following that, in brackets "[]", is the name of the utility that is running. Third, a letter indicating the access type of the operation. The options here are (W) for read-write access, (R) for read-only access and (U) for unique access. Finally, the location and process id of the running utility is displayed.

If *file_system_name* is specified, then **cvadmin** will connect to the current active FSM for that file system. If *N* (a number) is specified, cvadmin will connect to the *N*th FSM in the list. However, only active FSMs may be selected in this form.

show [*groupname*] [**long**]
Display information about the stripe groups associated with the selected file system. If a stripe group name *groupname* is given only that stripe group's information will be given. Omitting the *groupname* argument will display all stripe groups associated with the active file system. Using the **long** modifier will additionally display detailed information about the disk units associated with displayed stripe groups.

start *file_system_name* [**on** *hostname_or_IP_address*]
Start a File System Manager for the file system *file_system_name*. When the command is running on an MDC of an HA cluster, the local FSM is started, and then an attempt is made to start the FSM on the peer MDC as identified by the */usr/cvfs/config/ha_peer* file. When the optional *hostname_or_IP_address* is specified, the FSM is started on that MDC

only. The file system's configuration file must be operational and placed in */usr/cvfs/config/<file_system_name>.cfgx* before invoking this command. See **snfs_config**(5) for information on how to create a configuration file for an SNFS file system.

startd *daemon* [**once**]
Start the *daemon* process. For internal use only.

stat Display the general status of the file system. The output will show the number of clients connected to the file system. This count includes any administrative programs, such as **cvadmin**. Also shown are some of the static file-system-wide values such as the block size, number of stripe groups, number of mirrored stripe groups and number of disk devices. The output also shows total blocks and free blocks for the entire file system.

stats *client_IP_address* [**clear**]
Display read/write statistics for the selected file system. This command connects to the host FSMPM who then collects statistics from the file system client. The ten most active files by bytes read and written and by the number of read/write requests are displayed. If clear is specified, zero the stats after printing.

stop *file_system_name* [**on** *hostname_or_IP_address*]
Stop the File System Manager for *file_system_name*. This will shut down the FSM for the specified file system on every MDC. When the optional hostname or IP address is specified, the FSM is stopped on that MDC only. Further operations to the file system will be blocked in clients until an FSM for the file system is activated.

stopd *daemon*
Start the *daemon* process. For internal use only.

winidmap *__username__*
Show the MDC-mapped user id and group information for the username.

who Query client list for the active file system. The output will show the following information for each client.

Type - Type of connection. The client types are:
FSM - File System Manager(FSM) process
ADM - Administrative(cvadmin) connection
CLI - File system client connection. May be followed by a CLI type character:
S - Disk Proxy Server
C - Disk Proxy Client
H - Disk Proxy Hybrid Client. This is a client that has been configured as a proxy client but is operating as a SAN client.
Location - The client's hostname or IP address that was used by this client to connect to the FSM.
Up Time - The time since the client connection was initiated

EXAMPLES

Invoke the **cvadmin** command for file system named **snfs1**.

```
[root@dev-snc-daiquiri-n1 debug]# cvadmin -F snfs1
StorNext Administrator

Enter command(s)
For command help, enter "help" or "?".

List FSS
```

```
File System Services (* indicates service is in control of FS):
Selected Cluster: _cluster0/_addom0

Cluster: _cluster0/_addom0
 1>*fs1-man[0]           located on dev-snc-daiquiri-n1.mdh.quantum.com:52006 (pi
                         located on Qnode1:52006
                         located on 169.254.21.1:52006
                         located on 192.168.21.11:52006 [preferred]
 2> fs1-man[0]           located on dev-snc-daiquiri-n2.mdh.quantum.com:52004 (pi
                         located on 172.18.0.1:52004
                         located on 172.17.0.1:52004
                         located on Qnode2:52004
                         located on 169.254.21.2:52004
                         located on 192.168.21.12:52004 [preferred]
 3>*shared-SV1535CKH00012[0] located on dev-snc-daiquiri-n1.mdh.quantum.com:52003
                         located on Qnode1:52003
                         located on 169.254.21.1:52003
                         located on 192.168.21.11:52003 [preferred]
 4> shared-SV1535CKH00012[0] located on dev-snc-daiquiri-n2.mdh.quantum.com:52003
                         located on 172.18.0.1:52003
                         located on 172.17.0.1:52003
                         located on Qnode2:52003
                         located on 169.254.21.2:52003
                         located on 192.168.21.12:52003 [preferred]

Cluster: cluster1/_addom0
 5>*shared[0]            located on jk-rh6sp6-1.mdh.quantum.com:52003 (pid 8650)
 6> shared[0]            located on jk-rh6sp6-2.mdh.quantum.com:52003 (pid 19417)
 7>*snfs1[0]             located on jk-rh6sp6-1.mdh.quantum.com:52004 (pid 8651)
 8> snfs1[0]             located on jk-rh6sp6-2.mdh.quantum.com:52004 (pid 19431)

Select FSM "snfs1"

 Created           :    Wed Sep 26 12:27:43 2018
 Active Connections:    2
 Fs Block Size     :    4K
 Msg Buffer Size   :    8K
 Disk Devices      :    6
 Stripe Groups     :    3
 Fs Blocks         :    2684342720 (10.00 TB)
 Fs Blocks Free    :    2679776316 (9.98 TB) (99%)


snadmin (snfs1@cluster1/_addom0) >
```

Show all the stripe groups in the file system;

```
snadmin (snfs1@cluster1/_addom0) > show
Show stripe groups (File System "snfs1@cluster1/_addom0")

Stripe Group 0 [sg0]  Status:Up,MetaData,Journal,Exclusive
  Total Blocks:536868544 (2.00 TB)  Reserved:0 (0.00 B) Free:536841775 (2.00 TB)
(99%)
  MultiPath Method:Rotate
```

```
        Primary  Stripe [sg0]  Read:Enabled  Write:Enabled

    Stripe Group 1 [sg1]  Status:Up
      Total Blocks:1610605632 (6.00 TB)  Reserved:1082880 (4.13 GB) Free:1607753562
    (5.99 TB) (99%)
      MultiPath Method:Rotate
        Primary  Stripe [sg1]  Read:Enabled  Write:Enabled

    Stripe Group 2 [sg2]  Status:Up
      Total Blocks:1073737088 (4.00 TB)  Reserved:1082880 (4.13 GB) Free:1072022754
    (3.99 TB) (99%)
      MultiPath Method:Rotate
        Primary  Stripe [sg2]  Read:Enabled  Write:Enabled


    snadmin (snfs1@cluster1/_addom0) >
```

Display the **long** version of the **sg2** stripe group;

```
    snadmin (snfs1@cluster1/_addom0) > show sg2 long
    Show stripe group "sg2" (File System "snfs1@cluster1/_addom0")

    Stripe Group 2 [sg2]  Status:Up
      Total Blocks:1073737088 (4.00 TB)  Reserved:1082880 (4.13 GB) Free:1072022754 (
      MultiPath Method:Rotate
      Stripe Depth:2  Stripe Breadth:64 blocks (256.00 KB)
      Affinity Key:af1
      Realtime limit IO/sec:0 (~0 mb/sec) Non-Realtime reserve IO/sec:0
        Committed RTIO/sec:0 Non-RTIO clients:0 Non-RTIO hint IO/sec:0
      Disk stripes:
        Primary  Stripe [sg2]  Read:Enabled  Write:Enabled
          Node 0 [t1_0008]
          Node 1 [t1_0009]


    snadmin (snfs1@cluster1/_addom0) >
```

FILES

/usr/cvfs/config/.cfgx*

SEE ALSO

cvfs(8), **snfs_config**(5), **snfs_metadata_network_filter.json**(5), **fsmpm**(8), **fsm**(8), **mount_cvfs**(8), **sgmanage**(8)

NAME

cvaffinity − set, get, or delete the affinity of a file or directory

SYNOPSIS

cvaffinity [-r] -s key *filename [filename...]*

cvaffinity [-r] -l *filename [filename...]*

cvaffinity [-r] -d *filename [filename...]*

DESCRIPTION

cvaffinity can be used to set an affinity for a specific stripe group on a file or directory, or list the current affinity. An affinity is created in a stripe group through the file system configuration, see **snfs_config**(5). It is a name, up to eight (8) characters, describing a special media type. Use **cvadmin**(8) to see what affinity sets are assigned to the configured stripe groups.

If the affinity does not exist on any of the stripe groups, a set will fail. When allocating space for a file with an affinity, if the affinity does not exist for any of the stripe groups, or if the stripe groups with the affinity cannot be used to satisfy an allocation request, then the allocation will occur on the non-exclusive stripe groups. If there is no non-exclusive stripe groups, an **ENOSPC** is returned. See also **AffinityPreference** in **snfs_config**(5).

If a file does not have an affinity, it cannot have space allocated to it on stripe groups that have any affinities and are configured to be exclusive.

A common way to use the affinity capability, is to create a directory with **cvmkdir**(1) and have all files and directories below that directory inherit the affinity. Also, files can be created and have space pre-allocated with an affinity using **cvmkfile**(1).

For automatic affinity mapping for certain files, see **autoAffinity** and **autoAffinities** in **snfs_config**(5) and **snfs.cfgx**(5).

OPTIONS

-s *key* Set the given *key* to be the Affinity Key of the given file or directory. This key must be configured as an Affinity in the stripe group section of the file system configuration. Use **cvadmin**(8) to see the affinities in this file system. For files with an Affinity, new blocks allocated to that file are placed on a storage pool with the specified Affinity. For directories with an Affinity new files created in that directory inherit the Affinity from the directory.

-l This option says to just list the affinity for the specified file and exit.

-d This option says to delete the affinity from the specified file or directory, if one exists.

-r This option says to run recursively for directories.

filename

Specifies the file or directory operated on. Multiple *filename* arguments are permitted.

EXAMPLES

List the affinity on the file */usr/clips/foo*.

```
rock # cvaffinity -l /usr/clips/foo
```

Set this file or directory to use the stripe group that has the **jmfn8** affinity type.

```
rock # cvaffinity -s jmfn8 /usr/clips/filename
```

Remove the affinity from the */usr/clips/mydir*, if one is currently assigned.

```
rock # cvaffinity -d /usr/clips/mydir
```

NOTES

With regard to symbolic links, the recursive behavior of **cvaffinity** is similar to the Linux "find" command when run with defaults. That is, when symbolic links are encountered, the specified operation is performed on the object pointed to by the link even when the object resides outside of the hierarchy covered by the

top-level argument. However, **cvaffinity** will not recurse into directories pointed to by a symbolic link.

File system objects that are neither regular files nor directories such as device files and unix domain sockets cannot have affinities. If such objects are specified on the command line, an error will be generated. However, if such objects are encountered as part of a recursive operation, they are simply ignored without an error being generated.

The return value will be non-zero if any error is encountered for any argument. This includes any errors encountered while recursively traversing a directory.

SEE ALSO

snfs_config(5), **cvadmin**(8)

NAME

cvcp − StorNext Copy Utility

SYNOPSIS

cvcp [*options*] *Source Destination*

DESCRIPTION

cvcp provides a high speed, multi-threaded copy mechanism for copying directories and **tar** images onto and off of a StorNext file system. The utility uses IO strategies and multi-threading techniques that exploit the SNFS IO model.

cvcp can work in many modes;

Directory-to-directory copies of regular files.
Directory copy of regular files to a Vtape virtual sub-directory.
Single File-to-File copy.

Tar formatted data stream to a target directory.
Tar formatted data stream to a target Vtape virtual sub-directory.
Single file or directory copy to a tar formatted output stream.

In terms of functionality for regular files, **cvcp** is much like the **tar**(1) utility. However, when copying a directory or **tar** stream to a **Vtape** virtual directory, **cvcp** can rename and renumber the source images as they are being transferred. The files in the *Source* directory must have a decipherable numeric sequence embedded in their names.

The **cvcp** utility was written to provide high performance data movement, therefore, unlike utilities such as rsync, it does not write data to temporary files or manipulate the target files' modification times to allow recovery of partially-copied files when interrupted. Because of this, **cvcp** may leave partially-copied files if interrupted by signals such as SIGINT, SIGTERM, or SIGHUP. Partially-copied target files will be of the same size as source files; however, the data will be only partially copied into them.

OPTIONS

The *Source* parameter determines whether to copy a single file , use a directory scan or to use **stdin** for a **tar** extraction. If *Source* is a dash (-), then the standard input will be interpreted as a **tar** stream. Any other *Source* must be a directory or file name.

Using **cvcp** for directory copies is best accomplished by **cd**'ing to the *Source* directory and using the dot (.) as the *Source*. This has been shown to improve performance since fewer paths are searched in the directory tree scan.

The *Destination* parameter determines the target file, directory, or tar stream. If *Destination* is a dash (-), then all data will be sent to the standard output of the program in a standard **tar** stream. This stream can be directed, using the '>' parameter, to a regular file or to an output device such as a tape. Note that only one of *Source* or *Destination* may be a **tar** stream, not both.

USAGE

-a Archive mode. Preserve the original permissions, owner/group, modification times and links. This is the same as options w, x, y and z.

-A If specified, will **turn off** the pre-allocation feature. This feature looks at the size of the source file and then makes an ALLOCSPACE call to the file system. This pre-allocation is a performance advantage as the file will only contain a single extent. It also promotes file system space savings since files that are dynamically expanded do so in a more coarse manner. Up to 30% savings in physical disk space can be seen using the pre-allocation feature. **NOTE:** Non-SNFS file systems that do not support pre-allocation will turn pre-allocation off when writing. The default is to have the pre-allocation feature **on**.

-b *buffers*

Set the number of IO buffers to *buffers*. The default is two times the number of copy threads started(see the **-t** option). Experimenting with other values between 1 and 2 times the number of copy

threads may yield performance improvements.

-B When specified, this option disables the bulk create optimization. By default, this optimization is used in certain cases to improve performance. In some circumstances, the use of bulk create can cause cvcp to return errors if the destination file system is StorNext and the FSM process exits ungracefully while a copy is in progress. The use of the **-B** option avoids this potentiality at the cost of performance. The effect on performance will depend on whether bulk create is being disabled for other reasons as well as the size of the files with the impact being more observable when small files are copied.

-c When specified, if **cvcp** fails to copy a file it reports an error and continues.

-d Changes directory-to-directory mode to work more like **cp -R**. Without **-d**, **cvcp** copies the files and sub-directories under *Source* to the *Destination* directory. With **-d**, **cvcp** first creates a sub-directory called *Source* in the *Destination* directory, then copies the files and sub-directories under *Source* to that new sub-directory.

-k *buffer_size*
Set the IO buffer size to *buffer_size* bytes. The default buffer size is 4MB.

-l If set, when in directory to directory mode, follow symbolic links instead of copying the symbolic link.

-n If set, do not recurse into any sub-directories.

-p *source_prefix*
If set, only copy files whose beginning file name characters match *source_prefix*. The matching test only checks starting at character one.

-s The **−s** option forces allocations to line up on the beginning block modulus of the stripe group. This can help performance in situations where the I/O size perfectly spans the width of the stripe group's disks.

-t *num_threads*
Set the number of copy threads to *num_threads*. The default is 4 copy threads. This option may have a significant impact on speed and resource consumption. The total copy buffer pool size is calculated by multiplying the number of buffers(**-b**) by the buffer size(**-k**). Experimenting with the **-t** option along with the **-b** and **-k** options are encouraged.

-u Update only. If set, copies only when the source file is newer than the destination file or the destination file does not exist. The file modification time check uses a granularity of one second on Windows and microseconds on other platforms. This makes it possible for a slightly newer source file to not be copied over an older destination file even though **-u** is used. **-u** cannot be used with tar files.

-v Be verbose about the files being copied. May be specified twice for extreme verbosity.

-w If set, when in file to file mode, copy a symbolic link instead of following the link.

-x If set, ignore **umask**(1) and retain original permissions from the source file. If the super-user, set sticky and setuid/gid bits as well.

-y If set, preserve ownership and group information if possible.

-z If set, retain original modification times.

EXAMPLES

Copy directory *abc* and its sub-directories to directory */usr/clips/foo*. This copy will use the default number of copy threads and buffers. The total buffer pool size will total 32MB (8 buffers @ 4MB each).

Retain all permissions and ownerships. Show all files being copied.

```
rock% cvcp -vxy abc /usr/clips/foo
```

Copy the same directory the same way, but only those files that start with **mumblypeg**.

```
rock# cvcp -vxy -p mumblypeg abc /usr/clips/foo
```

Copy a single file *def* to the directory */usr/clips/foo/*

```
rock# cvcp def /usr/clips/foo
```

Copy from a **DLT** tar tape into */usr/clips/testdump*.

```
mt -f /dev/dlt rewind
rock% cvcp - /usr/clips/testdump </dev/dlt
```

Copy a file sequence in the current directory prefixed with **secta**. Place the files into the **Vtape** */usr/clips/n8 yuv* sub-directory. Use the verbose option.

```
rock% cvcp -v -p secta . /usr/clips/n8/yuv
```

Do the same thing but from a tar file.

```
rock% cvcp -p secta - /usr/clips/n8/yuv < /usr/clips/pic.tar
```

Copy directory */usr/clips/pictures* to a **tar** file named */usr/clips/pic.tar*

```
rock# cd /usr/clips/pictures
rock# cvcp . - > /usr/clips/pic.tar
```

Copy the file **HugeFile** to a DLT device as a **tar** stream.

```
rock# mt -f /dev/dlt rewind
rock# cvcp HugeFile - >/dev/dlt
```

CVCP TUNING

cvcp can be tuned to improve performance and resource utilization. By adjusting the **-t**, **-k** and **-b** options cvcp can be optimized for any number of different environments.

-t *num_threads*

Increasing the number of copy threads will increase the number of concurrent copies. This option is useful when copying large directory structures. Single file copies are not affected by the number of copy threads.

-b *buffers*

The number of copy buffer should be set to a number between 1 and 3 times the number of copy threads. Increasing the number of copy buffers increases the amount of work that is queued up waiting for an available copy thread, but also increases resource consumption.

-k *buffer_size*

The size of the copy buffer may be tuned to fit the I/O characteristics of a copy. If files smaller than 4MB are being copied performance may be improved by reducing the size of copy buffers to more closely match the source file sizes.

NOTE: It is important to ensure that the resource consumption of **cvcp** is tuned to minimize the effects of system memory pressure. On systems with limited available physical memory, performance may be increased by reducing the resource consumption of **cvcp**.

SEE ALSO

cvfs(8) **tar**(1)

NAME

cvdb − StorNext Client File System Debugger

SYNOPSIS

cvdb [*options*]

DESCRIPTION

cvdb provides a mechanism for developers and system administrators to extract debugging information from the *StorNext File System* (*SNFS*) client filesystem. It can be used by system administrators to change the level of system logging that the client filesystem performs. There is also a switch to retrieve various statistics.

USAGE

cvdb is a multi-purpose debugging tool, performing a variety of functions. A rich set of options provide the user with control over various debug and logging functions. The main features of **cvdb** are as follows:

Control debug logging.
Control level and verbosity of *syslog* **logging.**
Retrieve statistics.

OPTIONS

-G unix=*unixfile* **core=***corefile* [**ptrsize=**{**32**|**64**}]
Extract the debug log from a crashdump. The **ptrsize** keyword is optional, and defaults to the pointer size of the machine. Note: the **-G** option is not available on all platforms.

−g Retrieve the debug log from a running system. The log pointers are reset after this command, so that the next invocation of **cvdb -g** will retrieve new information from the buffer.

−C Continuously snap the trace. (Only useful with the **-g** option.)

−S *stopfile*
Stop snapping the trace when the file *stopfile* appears. (Only useful when also using the **-g** and **-C** options.)

-D *msec*
Delay *msec* milliseconds between trace snaps. The default is 1000 msec or one second. (Only useful when also using the **-C** and **-g** options.)

−F Save the trace output to files named *cvdbout.000000*, *cvdbout.000001*, etc. instead of writing to standard output. These files will appear in the current working directory. (Only useful when also using the **-C** and **-g** options.)

−n *cnt* After writing *cnt* files, overwrite the cvdbout out files starting with *cvdbout.000000*. This will essentially "wrap" the trace output.

−N *name*
Use *name* instead of **cvdbout** for the cvdb output files. (Only useful when also using the **-C**, **-g**, and **-F** options.)

−d Disable debug logging. This is the initial (start-up) default.

−e Enable debug logging. Disabled by default. Note: care should be taken when enabling logging in a production environment as this can significantly reduce file system performance.

-m modules=*bitvector* **logmask=***bitvector*
Specify the trace points for a given module or modules.

−l List the current trace points and their mask values.

-L List the available trace/debug points.

-s syslog={**none**|**notice**|**info**|**debug**}
Set the *syslog* logging value. The default at mount time is **notice**. See **mount_cvfs**(8) for more information.

-R size=[*nbytes*[**k**|**m**|**g**]]
Resize the the debug log. By default, the size of the log is 4MB. The minimum allowed size is 32768 bytes.

-p Get profile information. (Windows only.)

-P Toggle enabling/disabling of performance monitoring. (Windows only.)

-Q Set QOS callback failure test mode. (Windows only.)

-v Be verbose about the operations.

-i Print various statistics about the directory cache. If enabled and configured, the directory cache contains a number of buffers of directory contents. This cache is shared by all mounted StorNext file systems. Without **-v**, the following are printed:

The number of directory buffers currently cached and the maximum number allowed.

The number of times a buffer has been "hit" in the cache.

The number of times a cache search missed and required an RPC to the MDC.

The number of times a read of the directory re-used the LAST buffer that was used on the previous read of the same directory (similar to a cache hit but doesn't probe the cache).

The number of times a read of a directory specified the EOF offset.

The number of times the directory cache for a specific directory was invalidated. For example, if the directory contents changed after it was read and a subsequent read directory was done thereby causing the invalidation.

If **-v** is also specified, **-i** displays more statistics. Note that there are 2 hashes in the directory cache: one for all buffers and one by directory and file system.

The number of entries in the hash used to find dir cache buffers.

The # of searches using the directory cache buffer hash.

The total # of probes searching the directory cache for buffers. This can be larger than searches in the hash since multiple buffers may hit the same hash bucket.

The maximum probes after hitting a particular hash bucket (for buffers).

The maximum probes in the hash by directory and file system.

-b Print various statistics about each buffer cache. The only other option that can be used with this is **-v**. There are buffer caches per **cachebufsize**, see **mount_cvfs**(8). For each buffer cache, the following is printed:

# of mounted file systems using this buffer cache

# of buffers and total memory used

# of cache hits (and percentage)

# of cache misses (and percentage)

# of checks for write throttling to prevent over use by one file system. Write throttles only occur when more than 1 file system is using the cache.

# of times writes were throttled

If the **-v** option is also used with **-b**, the following additional statistics are printed for each buffer cache:

buffercachecap, see **mount_cvfs**(8)

buffercachewant (internal, means thread is waiting for a buffer)

bufhashsize (internal, # of entries in hash used to search buffers)

bcdirtycnt (internal, # of buffers with "dirty" data queued in cache)

dirty_ndone (internal, bcdirtycnt + buffers being written)

flusheractive (internal, flag indicating buffer flusher is active)

deferredflush (internal, # of buffers deferred after files are closed)

dirtywaiters (internal, # of threads waiting due to throttling)

rsvd max (internal, maximum amount of reserved space seen)

non-zero rsvd min (internal, minimum amount of reserved space seen > 0)

successful rsvd requests (internal, # of times reserved space was needed)

failed rsvd requests (internal, # of times reserved space not available)

-B Print buffer cache statistics using a curses based display that refreshes every second. Statistics are maintained separately for reads and writes, for each cache segment, and each mount point. Statistics labeled **Cumulative** are those representing the totals since the command was invoked or since the last reset. Those labeled **Current** represent the change in the last one second, roughly corresponding to the display refresh interval.

Two keystrokes are interactively recognized on systems supporting curses. A **q**, quit, will cause the display to terminate. An **r**, reset, will reset the cumulative counters to zeros.

The **-B** option is intended to be used to to analyze performance of the buffer cache with various applications, I/O subsystems, and various configuration parameters.

The refreshing display is supported on clients that have a curses capability. Other clients will produce a line oriented output with similar content.

A deadman timer will terminate the display after 30 seconds with no file systems mounted. This is to avoid hanging during file system shutdown.

-x Print distributed LAN proxy client and server statistics. The only other options that can be used with this are **-X** and **-f**. The proxy statistics are collected at both the client and server ends of each proxy connection. The client will have a connection entry for each path to a proxy server for each proxy client file system. A proxy server will have a connection entry for each path to each client which has the file system mounted.

Note: The distributed LAN proxy options are only available on platforms which support the distributed LAN client or server.

The following information is displayed for each proxy connection:

Client/Server System ID This IP address identifies the remote host.

Client IP Addr The IP address of the Client side of the connection.

Server IP Addr The IP address of the Server side of the connection.

Read Bytes/Sec Measured recent read performance of the connection.

Write Bytes/Sec Measured recent write performance of the connection.

FS Read Bytes/Sec Measured recent read performance for all connections for this file system.

FS Write Bytes/Sec Measured recent write performance for all connections for this file system.

Queued I/O Outstanding I/O (backlog) for this connection. The backlog is meaningful for client side connections only.

-X *option*

Dump statistics for each path in comma separated value (CSV) format. (Only useful with the **-x** option.) The following *options* are available:

1 Dump remote endpoint IP address and backlog in bytes. This option is only relevant for client mounts.

2 Dump remote endpoint IP address and read bytes per second.

3 Dump remote endpoint IP address and write bytes per second.

-f *fsname*

Specifies the file system name associated with an action option. For proxy statistics(**-x** option), filter on connections for the given file system. This parameter is required for the read/write statistics (**-y** or **-Y**) option.

-U **NOTE: Not intended for general use. Only use when recommended by Quantum Support as a fault injection tool.**

This option resets the network connection to the proxy peer for all proxy connections on all file systems for which this node is either a proxy client or gateway. This simulates an unexpected network disconnect and reconnect. It is intended to test the robustness of the error handling and reconnect logic in the StorNext DLC proxy client and gateway systems.

-y, **-Y** Display the read/write statistics for the file system specified with the **-f** option (required). If **-Y**, also clear the stats.

-Z **NOTE: Not intended for general use. Only use when recommended by Quantum Support as a fault injection tool.**

This option resets the network connection to the file system manager for all active file systems. This simulates an unexpected network disconnect and reconnect. It is intended to test the robustness of the error handling and reconnect logic in the StorNext file system.

-z **NOTE: Not intended for general use. Only use when recommended by Quantum Support as a performance measuring tool. Setting this option could result in data corruption, loss of data, or unintended exposure of uninitialized disk data!!**

This option turns on the DEVNULL capability and only applies to linux clients. Once enabled this option will continue to be enabled until reboot. When this option is enabled, all I/O for files with the DEVNULL affinity is not performed at the lowest level. The code paths are all executed including the allocation of space, but the data is not read or written to disk. Instead, writes simply complete the I/O and return and reads zero out the "read" buffer and complete the I/O.

Files without the DEVNULL affinity are unaffected by this setting.

Before attempting to use this capability, make sure no one is already using DEVNULL as an affinity on any file system the client has access too. Then, modify the file system configuration file, **snfs_config(5)**, for the file system under test to contain DEVNULL as an affinity on at least one stripe group that can hold data. Next, restart the fsm. Then, use **cvmkdir(1)** with -k DEVNULL to create a directory to hold files to be used for this test. Finally, enable the feature with this option, **cvdb -z**.

DEBUG LOGGING

Developing code that runs in the kernel is very different than programming a user-level application. To assist plugin developers who may not be familiar with the kernel environment, SNFS provides a simple "tracepoint like" debugging mechanism. This mechanism allows developers to use printf-like statements to assist in debugging their code.

To use the debugging facility, each module (typically a ".c" file), must declare a structure of type *ModuleL-*

ogInfo_t. This structure defines the name of the module as it will appear in the debug statements, and indicates the debug level that is in effect for that module.

```
ModuleLogInfo_t  MyLogModule =
        { "mymodule_name", DEBUGLOG_NONE};
```

To use the facility, each module must call the *AddLogModule()* routine. This is typically done when the module is first initialized (in the xxx_start() routine for a plugin). When logging is no longer required (as when the plugin is unloaded), the module should call *RemoveLogModule()* to free up the system resources.

Logging is not enabled by default. To enable logging at any time, specify the **enable** flag (**-e**)

```
shrubbery %h: cvdb -e
```

To disable logging, specify the **disable** flag.

```
shrubbery %h: cvdb -d -v
Disabling debug logging
```

The level of debugging is controlled via a 64-bit mask. This allows each module to have 64 different, discrete trace/log points. If the log point is enabled when the code is executed, the trace point will be dumped to the circular buffer.

A complete listing of all the pre-defined trace points can be obtained via:

```
rabbit %h:   cvdb -L
Trace points:
    cvENTRY      0x0001
    cvEXIT       0x0002
    cvINFO       0x0004
    cvNOTE       0x0008
    cvWARN       0x0010
    cvMEM        0x0020
    cvNUKE       0x0040
    cvLOOKUP     0x0080
    cvGATE       0x0100
    cvSTRAT      0x0200
    cvRWCVP      0x0400
```

These trace points would then be used to control the verbosity of logging. Using the example above, if the cvEXIT and cvINFO trace points are enabled, then only those trace points would be dumped to the log.

To enable the trace points, the first step is to determine the ID of the module. This is done with the **list** command.

```
shrubbery %h: cvdb -l
Module 'cvfs_memalloc'  module 0x000001 logmask 0x0000000000000000
Module 'cvfs_fsmsubr'   module 0x000002 logmask 0x0000000000000000
Module 'cvfs_fsmdir'    module 0x000004 logmask 0x0000000000000000
Module 'cvfs_fsmvfsops' module 0x000008 logmask 0x0000000000000000
Module 'cvfs_fsmvnops'  module 0x000010 logmask 0x0000000000000000
Module 'cvfs_sockio'    module 0x000020 logmask 0x0000000000000000
Module 'cvfs_subr'      module 0x000040 logmask 0x0000000000000000
Module 'cvfs_vfsops'    module 0x000080 logmask 0x0000000000000000
Module 'cvfs_vnops'     module 0x000100 logmask 0x0000000000000000
Module 'cvfs_dmon'      module 0x000200 logmask 0x0000000000000000
Module 'cvfs_rwlock'    module 0x000400 logmask 0x0000000000000000
Module 'cvfs_rw'        module 0x000800 logmask 0x0000000000000000
```

```
Module 'cvfs_fsmtokops' module 0x001000 logmask 0x0000000000000000
Module 'cvfs_extent'    module 0x002000 logmask 0x0000000000000000
Module 'cvfs_plugin'    module 0x004000 logmask 0x0000000000000000
Module 'cvfs_disk'      module 0x008000 logmask 0x0000000000000000
```

To enable the cvENTRY and cvEXIT trace points of the plugin, rwlock, vnops, and memalloc routines, use the **modules** command.

```
shrubbery %h: cvdb -m modules=0x4501 logmask=3
```

The bit masks are additive, not replacement. This means that modules and trace points you do not specify are unaffected. To turn on all debugging on all trace points, specify minus one (-1).

```
shrubbery %h: cvdb -m modules=-1 logmask=-1
```

Once the module has been added to the system, log messages will then be dumped into a 1 meg circular buffer. Modules may find it convenient to declare a macro in each file so that the form of log messages will be the same in each file. For example, the following macro definition and following log function would dump information to the log buffer if the trace point is enabled:

```
#define LOGINFO         (&MyLogModule)

LogMsg(LOGINFO, cvEXIT, "Plugin read return error %d bytes %llx",
            error, num_bytes);
```

To extract the messages from the log on a running system, use the **-g** option of **cvdb**.

To extract the messages from the log from a crashdump, you must use the **-G** option, and specify the name of the *unix* file and the name of the memory core file.

```
cvdb -G unix=unix.21 core=vmcore.21.comp > /tmp/logbuf
```

Note: The **-G** option is not available on all platforms and the location and names of the "unix" and "core" files will vary.

SYSLOG

The StorNext client file system can log certain events so that they show up on the system console and in the system log, */var/adm/SYSLOG*. The verbosity of messages can be controlled via the *syslog* parameter. The default is to log all messages. See **syslogd**(1M) for more information of setting up system logging.

There are four log levels: *none*, *notice*, *info*, and *debug* . The levels are prioritized so that the **debug** level is the most verbose; setting the level to **none** will turn off logging completely. The events that are logged at each level are as follows:

notice

- reconnection with the FSM.

info

- all **notice** messages, plus
- socket daemon termination

debug

- Currently unused

The log level is set to *debug* by default.

BUSY UNMOUNTS

Occasionally, it will be impossible to unmount the SNFS file system even when it appears that all processes are no longer using the file system. The problem is that the processes are most likely in the *zombie* state; while they do not show up in **ps**, then can be found using **crash**. Usually, these processes are waiting on a lock in the SNFS file system, or waiting for a response from the FSM.

DEBUG LOGGING EXAMPLES

To enable logging:

cvdb -e

To disable logging:

cvdb -d

To retrieve (get) log information on a running system:

cvdb -g > cvdbout

To continuously retrieve log information on a running system, snapping the trace once per second:

cvdb -g -C > cvdbout

To continuously retrieve log information on a running system, snapping the trace once every two seconds and stopping when the file named **STOP** appears:

cvdb -g -C -D 2000 -S STOP > cvdbout

To continuously retrieve log information on a running system, and save the output to files named *cvdbout.000000*, *cvdbout.0000001*, etc. and wrapping after 100 files have been written:

cvdb -g -C -F -n 100

To continuously snap traces named */tmp/snap.000000*, */tmp/snap.000001*, etc.:

cvdb -g -C -F -N /tmp/snap

To retrieve log information from a crashdump:

cvdb -G unix=*name_of_unix_file* **core=***name_of_core_file*

To list all the modules and their enabled trace points:

cvdb -l

To set trace points in individual modules:

cvdb -m modules=*bitmask_of_modules* **logmask=***tracepoints***.**

To resize the log to 12 megabytes:

cvdb -R 12m

To dump out all the pre-defined trace points:

cvdb -L

SEE ALSO

syslogd(1M), **umount**(8), **cvdbset**(8)

NAME

cvdbset − A program to control cvdb tracing.

SYNOPSIS

cvdbset [*options*]

DESCRIPTION

cvdbset is a tool for system administrators to control **cvdb**(8) tracing information from the *StorNext File System* (SNFS) client file system.

The level of tracing emitted can be controlled on a per module basis. The set of modules for which tracing is enabled is called the trace set. The level of tracing can be refined further by specifying a set of tracepoints (such as entry/exit points). The set of enabled tracepoints is called the logmask.

Warning: enabling tracing can have a substantial performance impact.

cvdbset can be used to:

List all the current client modules in the trace set.

Add all modules to the trace set.

Define the trace set.

Add selected modules to trace set

Remove selected modules from the trace set

Set the logmask for a set of modules in the trace set.

Resize the logging buffer

Start/stop continuous tracing

Disable tracing

OPTIONS

no options

Display the whether tracing is enabled/disabled, the size of the logging buffer, the modules in the trace set, and their corresponding logmasks.

all Enable tracing of all modules. Once cvdbset with a list of modules is invoked, some modules are turned off. **cvdbset all** sets all modules for tracing. When used with **+** or **-**, add or remove all modules.

[**:**]*module1* [**:**]*module2 ...*

When invoked with a list of modules, cvdbset first disables all modules. Then, it enables exactly the given list of modules. To see all modules that can be enabled, use the **cvdbset -l** command/option. If the module name is preceded by a **:**, all modules containing the module name will be affected.

+ [**:**]*module1* [**:**]*module2 ...*

When invoked with a plus sign (+) as the first argument followed by a list of modules, the given list of modules is added to the current trace set. If the module name is preceded by a **:**, all modules containing the module name will be affected.

− [**:**]*module1* [**:**]*module2 ...*

When invoked with a minus sign (-) as the first argument followed by a list of modules, the given list of modules is removed from the current trace set. If the module name is preceded by a **:**, all modules containing the module name will be affected.

-h Display a help message and exit.

-c Enable continuous cvdb tracing. The trace log will be retrieved once per second and placed in files named cvdbout.000001, ...

-d Disable cvdb tracing.

-g Dump the current trace buffer to standard out.

-l Display whether logging is enabled, the buffer size, and the logmask for all modules.

-L Display the list of all available tracepoints for use with the -t option.

-r *mb* Resize the trace buffer to *mb* megabytes.

-t *tracepoint*
For the indicated modules, enable tracing only for the indicated tracepoints. Multiple **-t** options can be supplied. Use the **-L** option to **cvdbset** to see a listing of tracepoints.

EXAMPLES

To see what modules are in the trace set and their logmasks, the command **cvdbset** with no parameters is used. Here is the output from this command at start-up.

```
Debug logging is DISABLED, Bufsize 4194304
Currently set masks:
Module '   proxy_clnt' module 0x0000000000000001   logmask 0xffffffffffffffff
Module '        cvdir' module 0x0000000000000002   logmask 0xffffffffffffffff
Module '       cvdisk' module 0x0000000000000004   logmask 0xffffffffffffffff
Module '         cvnc' module 0x0000000000000008   logmask 0xffffffffffffffff
Module '       cvpath' module 0x0000000000000010   logmask 0xffffffffffffffff
Module '      portmap' module 0x0000000000000020   logmask 0xffffffffffffffff
Module '       cvsock' module 0x0000000000000040   logmask 0xffffffffffffffff
Module '       cvsubr' module 0x0000000000000080   logmask 0xffffffffffffffff
Module '       dmigfs' module 0x0000000000000100   logmask 0xffffffffffffffff
Module '         dmig' module 0x0000000000000200   logmask 0xffffffffffffffff
Module '         dmon' module 0x0000000000000400   logmask 0xffffffffffffffff
Module '       extapi' module 0x0000000000000800   logmask 0xffffffffffffffff
Module '       extent' module 0x0000000000001000   logmask 0xffffffffffffffff
Module '        fsmat' module 0x0000000000002000   logmask 0xffffffffffffffff
Module '       fsmcom' module 0x0000000000004000   logmask 0xffffffffffffffff
Module '      fsmdmig' module 0x0000000000008000   logmask 0xffffffffffffffff
Module '     fsmproxy' module 0x0000000000010000   logmask 0xffffffffffffffff
Module '      fsmrtio' module 0x0000000000020000   logmask 0xffffffffffffffff
Module '     fsmtoken' module 0x0000000000040000   logmask 0xffffffffffffffff
Module '       fsmvfs' module 0x0000000000080000   logmask 0xffffffffffffffff
Module '     fsmvnops' module 0x0000000000100000   logmask 0xffffffffffffffff
Module '     memalloc' module 0x0000000000200000   logmask 0xffffffffffffffff
Module '    proxy_con' module 0x0000000000400000   logmask 0xffffffffffffffff
Module '       quotas' module 0x0000000000800000   logmask 0xffffffffffffffff
Module '        recon' module 0x0000000001000000   logmask 0xffffffffffffffff
Module '         rtio' module 0x0000000002000000   logmask 0xffffffffffffffff
Module '        rwbuf' module 0x0000000004000000   logmask 0xffffffffffffffff
Module '      rwproxy' module 0x0000000008000000   logmask 0xffffffffffffffff
Module '       rwlock' module 0x0000000010000000   logmask 0xffffffffffffffff
Module '           rw' module 0x0000000020000000   logmask 0xffffffffffffffff
Module 'slidingbucket' module 0x0000000040000000   logmask 0xffffffffffffffff
Module '    sockinput' module 0x0000000080000000   logmask 0xffffffffffffffff
Module '    proxy_srv' module 0x0000000100000000   logmask 0xffffffffffffffff
Module '   proxy_subr' module 0x0000000200000000   logmask 0xffffffffffffffff
Module '       vfsops' module 0x0000000400000000   logmask 0xffffffffffffffff
Module '        vnops' module 0x0000000800000000   logmask 0xffffffffffffffff
Module '         perf' module 0x0000001000000000   logmask 0xffffffffffffffff
Module '     md_cvdir' module 0x0000002000000000   logmask 0xffffffffffffffff
Module '    md_cvsock' module 0x0000004000000000   logmask 0xffffffffffffffff
```

```
Module '    md_cvsubr' module 0x0000008000000000   logmask 0xffffffffffffffff
Module '      md_dmon' module 0x0000010000000000   logmask 0xffffffffffffffff
Module '    md_fsmcom' module 0x0000020000000000   logmask 0xffffffffffffffff
Module '  md_memalloc' module 0x0000040000000000   logmask 0xffffffffffffffff
Module '    md_rwlock' module 0x0000080000000000   logmask 0xffffffffffffffff
Module '        md_rw' module 0x0000100000000000   logmask 0xffffffffffffffff
Module '   md_rwproxy' module 0x0000200000000000   logmask 0xffffffffffffffff
Module '  md_socksubr' module 0x0000400000000000   logmask 0xffffffffffffffff
Module '    md_vfsops' module 0x0000800000000000   logmask 0xffffffffffffffff
Module '     md_vnops' module 0x0001000000000000   logmask 0xffffffffffffffff
Module '    sh_cvsubr' module 0x0002000000000000   logmask 0xffffffffffffffff
Module '    sh_fsmcom' module 0x0004000000000000   logmask 0xffffffffffffffff
Module ' sh_sockinput' module 0x0008000000000000   logmask 0xffffffffffffffff
Module '     sh_vnops' module 0x0010000000000000   logmask 0xffffffffffffffff
```

To enable tracing for selected modules:

cvdbset md_vnops rw fsmvnops fsmtoken fsmdmig

This enables tracing for only these five modules and prints the output:

```
Setting md_vnops.
Setting rw.
Setting fsmvnops.
Setting fsmtoken.
Setting fsmdmig.
cvdb -m modules=0x0001000020148000 logmask=0xffffffffffffffff
```

If an argument is preceded by a colon (:), then any module whose name contains the argument as a substring is included.

To enable the md_vnops tracing module and all of the proxy-related modules:

cvdbset md_vnops :proxy

This displays the following output:

```
Setting md_vnops.
Setting proxy_clnt.
Setting fsmproxy.
Setting proxy_con.
Setting rwproxy.
Setting proxy_srv.
Setting proxy_subr.
Setting md_rwproxy.
cvdb -m modules=0x0001200308410001 logmask=0xffffffffffffffff
```

To add rwbuf and vnops modules to the current tracing set with the cvENTRY and cvEXIT tracepoints enabled:

cvdbset -t cvENTRY -t cvEXIT + rwbuf vnops

This displays the following output:

```
Adding rwbuf.
Adding vnops.
cvdb -m modules=0x0000000804000000 logmask=0x0000000000000003
```

To remove rwbuf and vnops from the current tracing set:

cvdbset - rwbuf vnops

This displays the following output:

```
Clearing rwbuf.
Clearing vnops.
cvdb -m modules=0x0000000804000000 logmask=0x0000000000000000
```

The special module **all** can be used with both the **+** and **-** options to add/remove all modules from the trace.

After tracing is enabled, **cvdbset -g** can be used to retrieve the trace. When desired, **cvdbset -d** can be used to disable tracing.

Various **cvdb**(8) command/options can be used for even finer control of tracing. See **cvdb**(8) for more details.

I/O PERFORMANCE ANALYSIS

The 'perf' trace module is very useful to analyze I/O performance, for example:

cvdbset perf

Then **cvdbset -g** will display info like this:

```
PERF: Device Write 41 MB/s IOs 2 exts 1 offs 0x0 len 0x400000 mics 95589 ino
PERF: VFS Write EofDmaAlgn 41 MB/s offs 0x0 len 0x400000 mics 95618 ino 0x5
```

The 'PERF: Device' trace shows throughput measured for the device I/O. It also shows the number of I/O's that it was broken into and number of extents (sequence of consecutive filesystem blocks).

The 'PERF: VFS' trace shows throughput measured for the read or write system call and significant aspects of the I/O including:

```
Dma  - DMA
Buf  - Buffered
Eof  - File extended
Algn - Well formed DMA I/O
Shr  - File is shared by another client
Rt   - File is real time
Zr   - Hole in file was zeroed
```

Both traces also report file offset, I/O size, latency (mics), and inode number.

Sample use cases:

1) Verify I/O properties are as expected.

The VFS trace can be used to ensure that the displayed properties are consistent with expectations, for example, well formed, buffered vs. DMA, shared/non-shared, or I/O size. If a small I/O is being performed DMA then performance will be poor. If DMA I/O is not well formed then it requires an extra data copy and may even be broken into small chunks. Zeroing holes in files has a performance impact.

2) Determine if metadata operations are impacting performance.

If VFS throughput is inconsistent or significantly less than Device throughput then it may be caused by metadata operations. In that case it would be useful to display 'fsmtoken', 'fsmvnops', and 'fsmdmig' traces in addition to 'perf'.

3) Identify disk performance issues.

If Device throughput is inconsistent or less than expected then it may indicate a slow disk in a stripe group or that RAID tuning is necessary.

4) Identify file fragmentation.

If the extent count 'exts' is high then it may indicate a fragmentation problem. This causes the device I/O's to broken into smaller chunks which can significantly impact throughput.

5) Identify read/modify/write condition.

If buffered VFS writes are causing Device reads then it may be beneficial to match I/O request size to a multiple of the ’cachebufsize’ (default 256KB or 1024K, see **mount_cvfs**(8)). Another way to avoid this is by truncating the file before writing.

SEE ALSO

cvdb(8)

NAME

/etc/init.d/cvfs − Initialization script for StorNext File System components (Unix only)

SYNOPSIS

service cvfs {**start**|**stop**|**restart**|**fullstop**}

DESCRIPTION

The StorNext File System (SNFS) can be controlled by the system **init**(8) mechanism. For more information on the init.d services reference the **init**(8) and **chkconfig**(8) man pages.

This initialization script responds to the normal **start**, **stop** and **restart** commands. This allows the StorNext File System to be started and stopped at any time, and the **fullstop** option unloads CVFS kernel modules.

Note: Take care not to trigger an HA reset action on a server when it has been configured for High Availability (HA). For more information about HA reference the **snhamgr**(8) man page.

See the **init**(8) or **chkconfig**(8) man pages for additional startup and shutdown options.

On Solaris clients, StorNext File System Services have been integrated into the Solaris service management facility, **smf**. The **cvfs** script is located in */usr/cvfs/bin*.

Starting SNFS Services

The **service cvfs start** command should be executed only after local file systems are mounted and the network is initialized. The following steps are executed:

- CVFS kernel modules are loaded.
- Base SNFS services are started. This includes the SNFS portmapper service and any needed HBA drivers.
- Any SNFS metadata servers included in the **fsmlist**(4) file are started automatically.
- SNFS file systems included in the systems **fstab**(5) or **vfstab**(4) file are mounted.
- StorNext Storage Manager (SNSM) components are started if they are installed.

Stopping SNFS Services

The **service cvfs stop** command should be executed before the network is brought down and before the local filesystems are unmounted. The following steps are executed:

- SNSM components are stopped.
- All SNFS file systems are unmounted.
- All SNFS server components are shut down.

The kernel modules are not unloaded unless the **fullstop** option is used.

LIMITATIONS

Only the **Linux** and **Unix** platforms are supported for utilizing /etc/init.d/cvfs

FILES

/etc/init.d/cvfs

SEE ALSO

mount(8), **snhamgr**(8), **fsmlist**(4), **vfstab**(4), **fstab**(5), **init.d**(7), **chkconfig**(8)

NAME

cvfsck − Check and Recover a StorNext File System

SYNOPSIS

cvfsck [*options*] [*FsName*] [*FsPath*]

DESCRIPTION

The **cvfsck** program can check and repair StorNext file system metadata corruption due to a system crash, bad disk or other catastrophic failure. This program also has the ability to list all of the existing files and their pertinent statistics, such as inode number, size, file type and location in the file system.

If the file system is active, it may only be checked in a *Read-only* mode. In this mode, modifications are noted, but not committed. The **−n** option may be used to perform a read only check as well.

The file system checking program must be run on the machine where the File System Services are running.

cvfsck reads the configuration file and compares the configuration against a saved copy that is stored in the metadata. It is important that the configuration file (see **snfs_config**(5)) accurately reflect the current state of the file system. If you need to change a parameter in a current configuration, save a copy of the configuration first or make sure */usr/cvfs/data/*FsName*/config_history/*.*cfgx.*<TIMESTAMP> already has a recent copy. Once the configuration file has been validated with the metadata version, if the configuration file is different and **cvfsck** is not in read-only mode, the new configuration is stored in the metadata and the previous version is written to */usr/cvfs/data/*FsName*/config_history/*.*cfgx.*<TIMESTAMP>.

After validating the configuration file, **cvfsck** reads all of the metadata, checks it for any inconsistencies, and the file system is repaired to resolve these issues or if in read-only mode, any problems are reported.

By default, modifications are first written to a file in the local file system instead of the SNFS disks. All fixes are made to this local file, including journal replay. When all problems are fixed and the run is complete, the user is asked if the changes should be copied to the actual SNFS disks. If the user responds "y", the changes are made. An answer of "n" indicates that the file system should not be changed. This allows the user to easily gauge the extent of problems with a file system before committing to the repair. The user can override this behavior with the **−n**, **−y**, and **-−T** options.

OPTIONS

-4 If there are files with unconverted or partially converted xattr chains that contain xattrs greater than 4KiB in length, destroy the oversized xattrs so conversion can continue. Use with caution.

-A Scan directories for name collisions that would occur on a case-insensitive file system. Note that the FSM must be stopped when using this option.

−a This option can only be used with **−f** and is used to tell **cvfsck** to print totals (all). When used, a line is printed after each stripe group showing how many free space fragments exist for that stripe group. In addition, at the end of the run, this options prints the grand total of free space fragments for all stripe groups.

−B *separator*
Use *separator* instead of comma (,) for the character used to partition fields when the **-x** option is specified.

−c *pathname*
Provide a specific path to a configuration file that is to be used, overriding the implicit location. This option is used when **cvupdatefs** invokes **cvfsck** as a sub-process to insure that the file system meta data is consistent prior to doing a capacity or stripegroup expansion.

−d Internal debug use. This option dumps a significant amount of data to the standard output device.

−e Report statistics for extents in each file. This reporting option enables all the same file statistics that the **-r** flag enables. In addition, the **-e** flag enables statistic reporting for each extent in a file. All extent data is displayed immediately following the parent file's information. See the **-r** flag description for file statistics output. The extent stats are output in the following order; *Extent#, Stripe group, File relative block, Base block, End block* No checking is done. This flag implies **-r** and **-n** flags. No tracing is enabled for this report option.

−E Erase i.e. "scrub" on disk free space. Cvfsck will write zeros over all free space on the disk. It works in conjunction with the **-P** option that reports the last block actually scrubbed in case of a crash during a scrub operation. This is intended for Linux.

−f Report free space fragmentation. Each separate chunk of free allocation blocks is tallied based on the chunk's size. After all free chunks are accounted for, a report is displayed showing the counts for each unique sized free space chunk. Free space fragmentation is reported separately for each stripe group. The free space report is sorted from smallest contiguous allocation chunk to largest. The "Pct." column indicates percentage of the stripe group space the given sized chunks make up. The "(sum)" column indicates what percentage of the total stripe group space is taken up by chunks smaller than, and equal to the given size. The "Chunk Size" gives the chunk's size in file system blocks, and the "Chunk Count" column displays how many instances of this sized chunk are located in this stripe group's free space. For more information on fragmentation see the **snfsdefrag**(1) and **sgdefrag**(8) pages. No checking is done. Implies **-n** flag. See also **-a** that is used to get more output.

−F This option causes **cvfsck** to make use of the compressed cache even when the configured value of bufferCacheSize is less than or equal to 1GB. It also sizes the cache to hold all metadata which can dramatically improve performance for aged file systems having large file counts. This option can cause **cvfsck** to use a lot of memory, so it is advisable to first obtain an estimate using the **−q** option.

−g Print journal recovery log. With this flag cvfsck reports contents of the metadata journal. For debugging use only. Implies **-n** flag.

−i Print inode summary report. With this flag cvfsck scans the inode list and reports inode statistics information then exits. This includes a breakdown of the count of inode types, hard links, and size of the largest directory. This is normally reported as part of the 'Building Inode Index Database' phase anyway but with this flag cvfsck exits after printing the inode summary report and skips the rest of the operations. This allows the inode summary report to run pretty fast. Implies **-n** flag.

−G Exit immediately after cvfsck completes on Windows systems. Without this flag the Windows command terminal will wait for a key to be pressed before exiting. This flag has no effect on non-Windows systems.

−j Execute journal recovery and then exit. Running journal recovery will ensure all operations have been committed to disk, and that the metadata state is up to date. It is recommended that cvfsck is run with the **-j** flag before any read-only checks or file system reports are run.

−J Dump raw journal to a file named jrnraw.dat and then exit. For debugging use only.

−K Forces the journal to be cleared and reset. **WARNING:** Resetting the journal may introduce metadata inconsistency. After the journal reset has been completed, run cvfsck to verify and repair any metadata inconsistency. Use this option with extreme caution.

−l This option will log any problems to the system log. **NOTE**: This flag may be deprecated in future releases.

−L This option forces all orphaned inodes (valid inodes which are not linked in to the directory tree) to be reattached in the lost+found directory. If this option is not present, cvfsck examines the RPL attribute on the inodes and tries to reattach them to the directory that used to hold them. In either usage, it tries to name the inodes using the name in the RPL attribute. If there is no RPL attribute, the inode number is used as a name. If that name already exists, the inode will be reattached using that name followed by a dash and a random number.

−M Performs simple checks that attempt to determine whether a new metadata dump is needed. If the checks find that a dump is needed, cvfsck will exit with status 1 and print an explanation. If the checks do not find that a dump is needed, cvfsck will exit with status 0. If an error occurs while performing the checks, cvfsck will print an explanation and exit with status 2. This option is useful only on managed file systems. Note: these checks are not exhaustive, and, in some cases, cvf-

sck will exit with status 0 when a new dump is actually required.

−m *size*

This option is used to specify the amount of memory in bytes to be used for the internal cache used to hold inode information. For larger file systems, this can improve the performance of **cvfsck**. Note that the memory estimate produced using the **-q** option will be increased by the amount specified with this option. The 'k', 'm', and 'g' extensions are recognized for this option. For example, **-m 2g** can be used to specify 2GB.

−n This option allows a file system to be checked in a **read-only** mode. Modifications are written to a file in the local file system instead of the SNFS disks. All fixes that would be made if **cvfsck** was run without the **−n** option are made to this local file, including journal replay. When the run is complete, the local file is thrown away. The file system itself is never changed.

−O If cvfsck is run on a file system while the FSM for that file system is active, cvfsck runs in shared mode. This means that it runs in read-only mode and only a small subset of the usual checking is performed. This is because the FSM changing the file system may confuse a full cvfsck and cause problems. The **−O** option causes cvfsck to perform full (read-only) checking anyway. Strange behavior may be observed.

−p *StripeGroupName*

This option provides a method for deleting all files that have blocks allocated on the given stripe group. All files that have at least one data extent on the given stripe group will be deleted, even if they have extents on other stripe groups as well. *WARNING:* Use this option with extreme caution. This option could remove files that the user did not intend to remove, and there are no methods to recover files that have been deleted with this option.

−q This option causes cvfsck to generate and estimate for disk and memory requirements and then exit. Any other options that will get used when performing the actual check should also be specified to improve estimate accuracy. For example, if the intent is to run **cvfsck -m2g -F** *FsName*, then to generate the estimate, run **cvfsck -q -m2g -F** *FsName*. Note that the base memory requirements will typically be around 600MB, so this should be taken into account when also using the **-m** option.

−Q This option causes cvfsck to print qustat statistics just before exiting.

−P Report progress of an Erase operation. This flag enables the writing of a file in */usr/cvfs/debug* of the last block on a given strip group that has been scrubbed. The files are created on a stripe group by stripe group basis as */usr/cvfs/data/cvfsck*_<FsName>*_sg*<StripeGroupOrdinal>. This is intended for Linux use.

−r This report option shows information on file state. Information for each file is output in the following order. *Inode#, Mode, Size, Block count, Extent count, Stripe groups, Affinity, Path* No tracing is enabled for this report option.

−R This option helps repair a file system which had cvmkfs accidentally run on it. First, cvfsck restores file system state which was saved by cvmkfs in */usr/cvfs/debug/*FsName.*cvmkfs*. Then, it continues as usual to fix any other problems it may encounter. The COW layer treats the restoration of saved state the same as any other file system modification. This option is only useful if the accidental cvmkfs is detected before the file system is mounted and changed. Using it at any other time is not advised. If unsure, please contact customer support.

−s *StripeGroupName*

THIS FUNCTIONALITY IS ONLY SUPPORTED ON MANAGED FILE SYSTEMS

Provides a method for restoring data on the given stripe group from a stored copy. **cvfsck** will truncate all files with data on the given stripe group. As such, all data blocks on that stripe group will be inaccessible and subsequent access of these files will trigger data retrieval from a stored copy. All data will be lost for files that do not have any stored copies. The **cvfsck** output will indicate whether or not the ALL_COPIES_MADE flag is set in the SNEA attribute for each truncated file. Files outside of all managed directories that have data on the given stripe group will be delet-

ed from the file system. *NOTE:* Use this option with extreme caution because it can result in permanent data loss.

−T *directory*
This option specifies the directory where all temporary files created by cvfsck will be placed. If this option is omitted all temporary files will be placed in the system's default temporary folder. NOTE: cvfsck does honor the use of TMPDIR/TEMP environment variables.

−t This option is used to check the work of the **-U** option on thin provisioned devices in the given file system. It causes **cvfsck** to print in **sn_dmap(1) -v** format, from the file system's perspective, an idea of what should be unmapped and mapped on each thin provisioned device in the given file system. One can then somewhat compare the **sn_dmap(1)** output with the **cvfsck -t** output as follows: any "mapped" space indicated by **sn_dmap(1)** must also be mapped in the **cvfsck** output. But, unmapped space from **sn_dmap(1)** may show up as mapped in the **cvfsck** output since the space has been allocated but not yet written. Going the other way, any mapped space from **cvfsck** may or may not be mapped from **sn_dmap**, depending if the allocated space was actually written. Any unmapped space (immediately after running **cvfsck -U**) must also appear as unmapped from **sn_dmap(1)**. In addition, this option verifies this final condition (checking ummapped space) printing lines where unmapped space is incorrectly still mapped. If all is copacetic, the following output appears for each device: *checked* **NNN** *unmapped entries with 0 errors* where **NNN** is the number of unmapped pieces checked. This currently only works on Linux and is intended as a debugging tool for quality assurance and development.

−U This option is for use with thin provisioned devices in the given file system. It causes UNMAPS or TRIMs operations for all file system free space. **Cvfsck** will issue the appropriate UNMAP/TRIM device operations for every free chunk in the file system. See also the **−t** option. This currently only works on Linux and only with Quantum QXS series storage.

−v Use verbose reporting methods.

−W This option causes cvfsck to always clean up any orphaned "Wopens" inodes that may have been generated when an earlier metadata restore from the metadata archive was performed using an older version of StorNext. Normally, cvfsck will only clean up these inodes if other metadata inconsistencies are detected prior to the orphan inode phase.

−x Report statistics. No checking is done. Implies **-e**,**-r** and **-n** flags. All values are in decimal. Data is output in this order: *Inode#, Mode, Size, Block Count, Affinity, Path, Extent Count, Extent Number, Stripe group, File Relative Block, Base, End, Depth, Breadth* By default, fields are comma separated. However, the separating character can be changed using the -B option. See also the -z option. No tracing is enabled for this report option.

−X (Engineering use only.) Free all inodes in extended attribute chains. Extended attributes present in these inodes will be deleted.

−y Fix any problems found in the file system without prompting for confirmation. The default behavior is to display the extent of the changes that will be made and prompt for whether or not to make the changes. The fixes are first made to a file in a file on the local file system (specified by **−T**). When all fixes are complete, they are copied into the actual SNFS disks.

−Y Same behavior as **-y** except that the changes are not buffered through the local file system as they are by default.

−z Enclose the Path field displayed by -x with double quotes. If the Path itself contains double quotes, replace each of them with two double quote characters.

−Z Remove all NT Security Descriptors from the file system. This is useful when ACLs are being abandoned to allow the use of the unixPermBits Security Model. This option should only be used when recommended by Quantum support. All StorNext systems must first unmount the file system prior to running cvfsck -Z since it modifies the metadata causing the FSM to prevent currently mounted clients from reconnecting. Running cvfsck -Z can take a long time for large file systems because all inodes have to be scanned for security descriptors. Also, since the metadata is updated

by cvfsck -Z, if the **metadataArchive** parameter is set to true in the file system configuration file, a new metadata archive will be generated when the FSM is restarted. Note that when running cvfsck -Z, the file system must be configured such that the securityModel is NOT **acl**, and enforceAcls, quotas, and windowsSecurity are all disabled either explicitly or by setting the securityModel to **unixpermbits**. After running cvfsck -Z, unix permissions on files and directories should be updated if needed.

FsName

Specifies a file system to check. Otherwise all file systems on this system will be displayed for selection.

FsPath Forces the program to use *FsPath/data* instead of */usr/cvfs/data* to locate the file systems.

EXIT VALUES

cvfsck will return one of the following condition codes upon exit.

```
0 - No error, no changes made to the file system
1 - Inconsistencies encountered, changes have been
    made to the file system
  - A read-only cvfsck will return 1 if journal replay is needed.
  - A read-only cvfsck will only print the needed fixes and not
    commit changes to the metadata.
2 - Fatal error, cvfsck run aborted
3 - Name collisions found, no repair needed
4 - Name collisions found, file system successfully repaired
```

NOTES

It is strongly recommended that the user should not run cvfsck with the **−y** or **−Y** options until the extent of any metadata corruption is known.

Unless running **cvfsck** in read-only mode, the file system should be unmounted from all machines before a check is performed. In the event that repairs are required and **cvfsck** modifies metadata, it will report this at the end of the check. If this occurs, any machines that continue to mount the file system should be rebooted before restarting the file system.

In order to ensure minimum run-time **cvfsck** should be run on an idle FSS server. Extraneous I/O and processor usage will severely impact the performance of **cvfsck**.

CRC checks are now done on all Windows Security descriptors. Windows Security Descriptors with inconsistent CRC's are removed causing affected files to inherit permissions from the parent folder.

Cvfsck limits the number of trace files to 100. It starts removing the oldest trace file if the max number of trace files in */usr/cvfs/data/*FsName*/trace* is exceeded before a new file is created.

NOTE: On large file systems cvfsck may requires 100s of megabytes or more of local system disk space for working files. Please refer to the StorNext documentation to ensure minimum system requirements are met.

If the output of -x is to be used with Excel, consider the use of the -z option so that lines having pathnames containing commas can be parsed. If the output of -x is to be used with Unix tools such as awk, Perl, or Python, consider using the -B option with a field separator such as '|' or similar that does not appear as a character in a pathname.

FILES

*/usr/cvfs/data/**
*/usr/cvfs/data/*FsName*/config_history/*.cfgx.*<TIMESTAMP>
/usr/cvfs/config/.cfgx*

SEE ALSO

snfs_config(5) **cvmkfile**(1), **cvupdatefs**(8), **cvadmin**(8), **sgdefrag**(8), **snfsdefrag**(1)

NAME

cvfsd − StorNext File System Daemon

SYNOPSIS

Internal Kernel Thread

DESCRIPTION

Cvfsd is a server daemon that is launched by the *StorNext File System* (*SNFS*) **mount_cvfs**(8) command. It is an internal kernel thread and is used for network communication to the *File System Manager*. Multiple **cvfsd** threads are launched for each SNFS file system. The number of **cvfsd** threads can be modified. See **mount_cvfs**(8) for details.

ENVIRONMENT

Quantum Internal Use Only - only on Solaris.

WATCHER_NODETACH

If set, **cvfsd** will not detach from its parent process. Used start **cvfsd** under the control of a debugger.

WATCHER_NORESTART

If set, **cvfsd** will not be restarted automatically after an unsuccessful exit or crash.

LIMITATIONS

Only the **Linux** and **Unix** platforms are supported with the cvfsd daemon

SEE ALSO

cvfs(8), **mount_cvfs**(8)

NAME

cvfsdb − StorNext File System debugging tool

SYNOPSYS

cvfsdb *FsName*

DESCRIPTION

The **cvfsdb** command is a tool for debugging a StorNext file system.

WARNING: Quantum Internal use only. The **cvfsdb** command can easily damage a StorNext file system, and should only be used under the direction of customer support.

OPTIONS

FsName

The file system to debug.

COMMANDS

The **cvfsdb** command is an interactive program, that contains builtin help on commands and usage.

help [*command*]

Display help information. If *command* is omitted, the **help** command will display a list of commands that **cvfsdb** can understand. If *command* is provided, command specific help will be given.

exit, **q**, **quit**

Exit **cvfsdb**.

The output of any command can be redirected using:

command **|** *shell_command*

Redirect the output of *command* to *shell_command* using **popen**(3).

command **>** *file*

Redirect the output of *command* into *file*, which will be overwritten if it exists.

command **>>** *file*

Append the output of *command* to *file*.

SEE ALSO

popen(3)

NAME

cvfsid − Display SNFS System Identifier

SYNOPSIS

/usr/cvfs/bin/cvfsid [**-?Ghnl**]

DESCRIPTION

cvfsid provides a mechanism for displaying the SNFS identifier for the executing system. For customers using client-based licensing, SNFS identifiers are used to generate individual client licenses. This identifier string is submitted to Quantum Technical Support for license authorization keys. See the installation instructions for additional information on SNFS licensing.

OPTIONS

−h, **−?** Display help

−G Gather mode. *NOTE*: Not intended for general use. Only use when recommended by Quantum Support.

−l List the local host's Authorizing IDs, IP addresses, and MACs. (Linux only.)

−n Display the network interface information in a compact, machine readable form. (Linux only.)

When executed without options, cvfsid prints the information required to generate a license for the host on which it is executed. Simply execute the program on each participating system, and either Email or Fax the identifiers to Quantum Technical Support for authorization keys.

After the license keys are received cut-and-paste them into the file */usr/cvfs/config/license.dat* on the system that runs the CVFS File System Manager.

FILES

/usr/cvfs/config/license.dat

SEE ALSO

cvfs(8), **snfs_config**(5), **SNFS Installation Instructions**

NAME

cvfs_config − StorNext File System Configuration File

DESCRIPTION

This man page is deprecated - see **snfs_config**(5) for details on the StorNext File System Configuration File

SEE ALSO

snfs_config(5), **snfs.cfgx**(5), **snfs.cfg**(5),

NAME

StorNext File System Failover - How To Configure and Operate

DESCRIPTION

The StorNext File System uses a single File System Manager (*FSM*) process per file system to manage metadata. Since this is a single point of failure, the ability to configure an additional hot-standby FSM is supported. This redundant configuration is called High Availability (HA). An HA cluster comprises two identically configured server-class computers operating as metadata controllers (MDC). Either MDC in an HA cluster can serve as the *primary* MDC for the purposes of configuring the cluster and for running the processes that provide the Stornext Storage Manager (SNSM) features. The alternate MDC is called *secondary*.

All SNSM HA clusters must have one (HaShared) unmanaged Stornext file system dedicated to configuration and operational data that is shared between the MDCs. The MDC running the active HaShared FSM is the *primary* MDC by definition. The *primary* MDC runs the active FSMs for all the managed file systems (HaManaged), as well as the HaShared file system, and it runs all the management processes together on one MDC. In the event that an HaManaged FSM process fails, another FSM process for that file system will be started and activated on the primary. There are no redundant FSM processes on the secondary MDC for HaManaged file systems. Non-managed file systems (HaUnmanaged) can be active on either MDC. There is a redundant standby FSM ready to take control through the activation protocol for each HaUnmanaged file system.

HA cluster configurations guard against data corruption that could occur from both MDCs simultaneously writing metadata or management data by resetting one of the MDCs when failure conditions are detected. HA resets allow the alternate MDC to operate without risk of corruption from multiple writers. HA reset is also known as *Shoot Myself in the Head* (SMITH) for the way that resets are triggered autonomously. HA resets occur when an active FSM fails to update the arbitration control block (ARB) for a file system, which prevents the standby from attempting a takeover, but also fails to relinquish control. HA reset also occurs when the active HaShared FSM stops unless the file system is unmounted on the local server, which ensures that management processes will only run on a single MDC.

There are three major system components that participate in a failover situation. First, there is the FSM Port Mapper daemon, **fsmpm**(8). This daemon resolves the TCP access ports to the server of the file system. Along with this daemon is the Node Status Server daemon (*NSS*). This daemon monitors the health of the communication network and the File System Services. The third component is the *FSM* that is responsible for the file system metadata.

Whenever a file system driver requests the location of a file system server, the *NSS* initiates a quorum vote to decide which of the FSMs that are standing by should activate. The vote is based on an optional priority specified in the FSM host configuration list, **fsmlist**(4), and the connectivity each server has to its clients. When an elected FSM is given the green light, it initiates a failover protocol that uses an arbitration block on disk (ARB) to take control of metadata operations. The activating server *brands* the file system by writing to ARB block, essentially taking ownership of it. It then re-checks the brand twice to make sure another server has not raced to this point. If all is correct, it lets the server take over. The new server re-plays the file system journal and publishes its port address to the local FSM Port Mapper. Once these steps are taken, clients attempting connection will recover their operations with the new server.

SITE PLANNING

In order to correctly configure a failover capable StorNext system, there are a number of things to consider. First, hardware connectivity must be planned. It is recommended that servers have redundant network connections. In order to failover, the metadata must reside on shareable storage.

CONFIGURATION

This section will show how to set up a StorNext configuration in a way that will support failover.

File System Name Server Configuration

The **fsnameservers**(4) files should have two hosts described that could manage the File System Name Services. This is required to ensure that the name service, and therefore the NSS voting capabilities, do not have a single point of failure. It is recommended that these server machines

also be named as the name servers. It is important to note that the **fsnameservers** list be consistent and accurate on all of the participating SAN clients. Otherwise some clients may not correctly acquire access to the file system. In other words, be sure to replicate the **fsnameservers** list across all SNFS clients.

FSM List

Each line in the FSM list file **fsmlist**(4) describes a single file system name. An entry in this file directs the **fsmpm** process to start an **fsm** process with a configuration file of the same name.

File System Configuration

GUI supported configuration is done by completely configuring a single MDC, and then the configuration is copied to the other MDC through the HaShared file system. By-hand configurations must be exactly the same on both MDCs.

License Files

License files must also be distributed to each system that may be a server.

OPERATION

Once all the servers are up and running they can be managed using the normal **cvadmin**(8) command. The active servers will be shown with an asterisk (*****) before it. Server priorities are shown inside brackets. DO NOT start managed FSMs on the secondary server by hand as this violates the management requirement for running all of them on a single MDC. When a managed FSM will not start reliably, a failover can be forced by the snhamgr command on the primary MDC as follows:

```
snhamgr force smith
```

FILES

/usr/cvfs/config/license.dat
/usr/cvfs/config/fsmlist
/usr/cvfs/config/fsnameservers

SEE ALSO

cvadmin(8), **snfs_config**(5), **cvfsck**(8), **fsnameservers**(4), **fsm**(8), **fsmpm**(8)

NAME

cvgather − Compile debugging information for a StorNext File System

(The Windows version of cvgather contains different features and options. See the StorNext Windows "Gather Debugging Info (cvgather.exe)" help page for more information.)

SYNOPSIS

cvgather -f *FsName* [**-sukxr**] [**-o** *OutputFile*] [**-n** *NumberOfCvlogs*] [**-U** *UserCore*] [**-K** *KernelCore*] [**-p** *SnfsPath*]

DESCRIPTION

The **cvgather** program is used to collect debug information from a file system. This creates a tar file of the system's *StorNext File System* debug logs, configuration, version information and disk devices.

The **cvgather** program will collect client debug information on client machines and server information on server machines, as well as portmapper information from all machines. System log files as well as SNFS log files are included. At the users option, **cvgather** also collects core files from user space utilities, such as the fsm, and also from the operating system kernel, when available. This information provides Quantum technical support staff with enough information to deal with most problems encountered by SNFS users.

USAGE

When the operator encounters an error using SNFS and wishes to send debugging information to Quantum technical support, the **cvgather** utility may be run. The following command arguments and options affect the behavior of **cvgather**.

-f *FsName*
: Specify the name of the file system for which debugging information should be collected. Some information is universal to all installed file systems, while some is unique to each file system.

-k
: Collect the core file from the operating system kernel. This option is not supported on Linux. The **-k** option collects the kernel core from the default location for the machine's operating system. To collect the kernel core from another location use **-K**.

-K *KernelCore*
: Collect the kernel core file from any file. You must specify the full filename as well as the path.

-n *NumberOfCvlogs*
: Specify the number of cvlog files to include in the tarball. If this option is not selected, 4 will be used. This is the default number of cvlogs used by the fsm.

-o *OutputFile*
: Specify the name of the output file. This name is appended with a _timestamp and the suffix '.tgz'. The timestamp is appended to the filename to allow for the existence of multiple tar files. If this option is not selected, the name of the file system will be used as a default.

-p *SnfsPath*
: Specify the file path to the SNFS install directory. If this option is not selected, the path */usr/cvfs* will be used as a default.

-s
: Gather symbol information without core files.

-u
: Collect the core file from user executables, such as the fsm. By default, if they exist, cvgather will pick up a file named "core" and the the most recently modified "core.*" file on systems that support core file names with extensions. The **-u** option collects core files from the 'debug' directory in the SNFS directory. To collect user core files from another location or core files with with non-standard names use **-U**.

-U *UserCore*
: Collect the user core file from any file. You must specify the full filename as well as the path.

−r
: Show numerical addresses instead of trying to determine symbolic host

-x Exclude files that are collected by pse_snapshot. Note that this option is intended to be used by pse_snapshot only and not for general use. The behavior of this option may change without warning.

When **cvgather** is run it will create a tar file, that can be simply e-mailed to Quantum technical support for evaluation. It is recommended that the tar file be compressed into a standard compression format such as compress, gzip, or bzip2.

NOTES

IMPORTANT: **cvgather** creates a number of temporary files, thus must have write privileges for the directory in which it is run. These files, as well as the output tar file can be very large, especially when the kernel core file is included, thus adequate disk space must be available.

Several important log files are only accessible by the root user, thus it is important that **cvgather** be run with root privileges to gather the entire range of useful information.

FILES

/usr/cvfs/config/.cfgx*
/usr/cvfs/debug/cvfsd.out
/usr/cvfs/debug/nssdbg.out
/usr/cvfs/debug/fsmpm.out
/usr/cvfs/data/<file_system_name>*/log/cvlog**

LIMITATIONS

Only the **Linux** platform is supported with **cvgather**

The Windows version of **cvgather** contains different features and options. See the StorNext Windows "Gather Debugging Info (cvgather.exe)" help page for more information.

SEE ALSO

cvdb(8), **cvversions**(1), **cvfsid**(8) **cvlabel**(8)

NAME

cvlabel − Label StorNext Disk Devices (LUNs)

SYNOPSIS

cvlabel -l [**−agsv**] [**−F** *filter*]

cvlabel -L [**−agv**] [**−F** *filter*]

cvlabel -j [**−av**] [**−F** *filter*]

cvlabel -c [**−T]** [**−F** *filter*]

cvlabel -C *format* [**−F** *filter*]

cvlabel -x

cvlabel [**−frRvw**] [**−q** *tag_q_depth*] *label_list*

cvlabel [**−fw**] **−u** *VolumeName*

cvlabel [**−fw**] **−U** *DeviceName*

cvlabel −D *VolumeName*

DESCRIPTION

cvlabel is used when configuring the *StorNext File System* disks. One host that has visibility to all the storage area network disk devices must create a list of disk labels, their associated device names and optionally the sectors to use. The **mount_cvfs**(8) process uses the volume labels to determine which disk drive is to be used for *SNFS* stripe group nodes. The label name that is written to a disk device must match the **[Disk ...]** name in the *File System Manager (FSM)* configuration. See **snfs_config**(5) for details of the FSM configuration file.

It is recommended to first use **cvlabel** with the **−l** or **−L** option. This option will present all of the usable disk devices found on the system. It will try to identify the volume label and display the results. This will help determine what disk drives are visible to the client.

The next step is to create the *label_list* file. Use */usr/cvfs/examples/cvlabels.example* as a template for your file. Or, use **cvlabel** with the **−c** option, in which case **cvlabel** will write on stdout the list of all devices found in a format compatible with a *label_list* file.

Once a *label_list* file has been generated it must be edited to match the desired SNFS label updates. All LUNs included in the *label_list* file that are not allocated to the *StorNext File System* should be removed from the *label_list* file to prevent accidental overwriting of existing data. Once all updates to the *label_list* are complete cvlabel should be run using this file to apply label changes to the indicated LUNs.

A final option for creating a label file is to use the **−C** option with a format string. This behaves the same as the **−c** option, except the format string is used to build template labels. The format string uses a printf like syntax where % followed by a letter is replaced by information obtained from the storage. The available format strings are **%B** size in sectors, **%L** lun number, **%C** controller id and **%S** serial number. Care should be taken to use a format which generates unique names for devices before using the output to label them.

Certain RAID devices require special handling. Cvlabel uses the raid strings inquiry table to determine which devices require special handling. The default table (displayed with the **-R** option), can be overridden by a user supplied file */usr/cvfs/config/raid-strings.* Note: the **-R** option is not intended for general use and may be deprecated in the future. Only use when recommended by Quantum Support.

OPTIONS

-l, **-L** Use the **−l** option (short format) or the **−L** option (long format) to list usable disk devices on the system.

-j Use the **−j** option (JSON format) to list usable disk devices on the system in a machine and human readable format.

−u *VolumeName*
Use the **−u** *VolumeName* option to unlabel the specified volume.

−U *DeviceName*
The **−U** *DeviceName* option is similar to the **-u** option, except that the path to the device special file is used instead of the label name.

-s When used in conjunction with the **−l** option, the **−s** option prints the disk device **serial #**, which can be used to distinguish the difference between **duplicate labels** and **multiple paths**.

-g When used in conjunction with the **-l** or **-L** options, the **-g** option also prints GUID information for EFI-labeled disks. The GUID includes a timestamp and the MAC address of the node that created the label.

-a When used in conjunction with the **−l** or **−L** options, the **−a** option also prints unusable disk devices, along with a description of why they are unusable. This is usually due to a lack of OS support for large LUNs or an unsupported disk label format.

−F *filter*
When used in conjunction with the **−c**, **−C**, **−l** or **−L** options, the **−F** *filter* option will only list devices whose inquiry string contains the *filter* string.

-v The **-v** option prints more information about the labeling process. Multiple **−v** options accumulate, providing more information often used for debugging the label process.

-q The **-q** option can be used during labeling to set the Command Tag Queue Depth. By default, the Depth is set to 16.

-f The **−f** option forces labeling and you will not be asked for confirmation before labeling (or unlabeling) a disk device. **WARNING:** errors in the SNFS label_list file can cause data loss.

-c The **−c** option outputs a cvlabel format template file to stdout. This template file will reflect all disk devices visible to the local system. Use this template to build a cvlabel file. **WARNING: Be sure to edit the template file to remove all devices which you do not want labeled.**

−T The **−T** option can be used in conjunction with the **−c** option to facilitate conversion of labels from the old VTOC format to the new EFI format. The output will be similar to the ordinary **−c** output, but devices that do not need conversion or cannot be safely converted will be output as comment lines, along with explanatory text. Only convertible devices are output normally.

−D *VolumeName*
The **−D** *VolumeName* option can be used to dump the label for *VolumeName* in ascii to stdout. Examining this output is useful when debugging labels.

−r The **−r** option can be used to force a disk to be relabeled, even if there are no changes to the label information. Normally such disks are skipped.

−R The **−R** option can be used to display the default raid strings inquiry table. Note that EFI labels are not supported on IRIX systems for older releases of the Xsan File System.

−i The **−i** option is no longer supported. Labels should be in EFI format.

−w The **−w** option tells cvlabel to wait for the completion of the disk scan that is requested after a disk label has been written or a volume has been unlabeled. The disk scan requests that the file system server update its internal device tables and the **−w** option ensures that the operation has been completed. Note that a disk scan may take a number of seconds on a large SAN or a SAN that is experiencing device errors.

***WARNING* Use this program with extreme caution! Modifying a system disk's volume label may result in irreparable harm to your system. It may render the system inoperable and force you to repair the volume using the boot maintenance program. Only label disk devices which you are sure are to be used for the StorNext File System's storage area network.**

FILE FORMAT

You may use the */usr/cvfs/examples/cvlabels.example* file as a template.

A label entry consists of two or three parameters on a single line. White space and comment lines are allowed. Comment lines are designated by using a pound sign (**#**) as the first non−white space character of the line.

The *label_list* file format is as follows:

<SNFS_label_name> <operating_system_device_name> [<sectors> [<type>]]

Where:

<SNFS_label_name>

The **<SNFS_label_name>** parameter is the name of the disk as described in the **FSM** configuration file. The parameter must match a **[Disk <SNFS_label_name>]** entry.

<operating_system_device_name>

The **<operating_system_device_name>** is the device name of the complete disk device.

NOTE: operating system device names may change after reboots and will differ per system. Always configure SNFS label files, and label devices in the same session.

On **Windows** systems, the devices start as **PhysicalDrive0** and increment up to the number of drives configured.

<sectors>

The **<sectors>** parameter is the number in 512-byte sectors that matches the **[DiskType ...]** configuration in the FSM configuration file. This is required for disks that must be configured smaller than their actual size. For example, MPIRE video disks must be under−configured to eliminate using the last zone of the disk. If **<sectors>** is not specified or is specified as **−**, then the **cvlabel**(8) program will use the entire available volume.

<type>

The **<type>** parameter is used to override the default label type, or to change the label type for a disk that already has a label. The value must be **EFI** for changing a **VTOC** label to an **EFI** label.

EXAMPLES

List all the disk devices in a system.

```
rock # cvlabel -L
/dev/sda   [ATA     ST500NM0011     PA08] MBR  Controller 'default', Serial '500
/dev/mapper/mpathai [Quantum StorNext QX     H205] SNFS-EFI "dexter1d1"  Control
[...]
/dev/mapper/mpathap [DotHill DH4730          H205] unknown  Controller '208000C0
[...]
/dev/mapper/mpathan [Quantum QXS             G22x] SNFS-EFI "snfs_data_bh-5600-1_
```

Then create a template label file:

```
rock # cvlabel -c >label_list
```

The output file will include an entry for the 'unknown' disk:

```
CvfsDisk_UNKNOWN /dev/mapper/mpathao # host 0 lun 3 sectors 18554669023 sector_si
```

Edit the *label_list* file, changing **CvfsDisk_UNKNOWN** to the desired label name:

```
CvfsDisk_39 /dev/mapper/mpathao
```

Now label the disk devices. Your *label_list* file must be specified on the command line.

```
rock # cvlabel label_list
```

WARNING This program will over-write volume labels on the devices specified in the file *label_list*.

After execution, the devices will only be usable by the StorNext File System. You will have to re-partition the devices to use them on a different file system.

Do you want to proceed? (Y / N) -> y

/dev/mapper/mpathap [DotHill DH4730 H205] unknown Controller '208000C0FF155519', Serial '600C0FF0001
Do you want to label it SNFS-EFI - Name: CvfsDisk_39 Sectors: 18554669023 (Y / N) -> y
New Volume Label -Device: /dev/mapper/mpathap SNFS Label: CvfsDisk_39 Sectors: 18554669023.

Done. 1 source lines. 1 labels.

The labels are done. List the disk devices again.

```
rock # cvlabel -L
/dev/sda   [ATA     ST500NM0011     PA08] MBR  Controller 'default', Serial '500
/dev/mapper/mpathai [Quantum StorNext QX     H205] SNFS-EFI "dexter1d1"  Control
[...]
/dev/mapper/mpathap [DotHill DH4730          H205] SNFS-EFI "CvfsDisk_39"  Contro
[...]
/dev/mapper/mpathan [Quantum QXS             G22x] SNFS-EFI "snfs_data_bh-5600-1_
```

Generate a label file of all LSI storage which uses the controller serial number and lun numbers as components of the labels.

```
rock # cvlabel -C CVFS_%S_%L -F LSI > label_list
```

Display to stdout the default raid strings inquiry table.

```
rock # cvlabel -R
# Raid inquiry string table
# Controls interpretation of raid mode pages based on inquiry strings
#
# Allowed types:
# LSI           LSI (Engenio) Raid in AVT mode
# Clariion      Clariion (EMC) Raid in Auto trespass mode
# Seagate       Dual port Seagate JBODs
# JBOD          No special handling (Real JBOD or RDAC driver)
# Quantum StorNext QX
# Quantum QXS

# String 1       String 2             Raid Type

  "DGC"         ""                    Clariion
  "ENGENIO"     ""                    LSI
  "IBM"         "1722-600"            LSI
```

```
"IBM"          "1742-900"            LSI
"IBM"          "1814"                LSI
"IBM"          "Universal Xport"     LSI
"LSI"          "VirtualDisk"         JBOD
"LSI"          "MegaRAID"            JBOD
"LSI"          "ProFibre"            JBOD
"LSI"          "Universal Xport"     LSI
"NETAPP"       "Universal Xport"     LSI
"ENGENIO"      "Universal Xport"     LSI
"LSI"          ""                    LSI
"SGI"          "TP9300"              LSI
"SGI"          "TP9400"              LSI
"SGI"          "TP9500"              LSI
"SGI"          "TP9700"              LSI
"SGI"          "IS600"               LSI
"SGI"          "IS500"               LSI
"SGI"          "IS400"               LSI
"SGI"          "IS300"               LSI
"STK"          "FLEXLINE"            LSI
"STK"          "OPENstorage"         LSI
"STK"          "Universal Xport"     LSI
"STK"          "BladeCtlr"           LSI
"Quantum StorNext QX"    ""          QX
"Quantum QXS"    ""                  QX
"SEAGATE"      ""                    Seagate
```

Use the default raid strings inquiry table to seed a user-defined table.

```
rock # cvlabel -R > /usr/cvfs/config/raid-strings
```

NOTES

Some operating systems require a reboot after a disk is labeled or relabeled. It is recommended that SNFS nodes are rebooted after new labels are written or existing labels are updated.

FILES

/usr/cvfs/examples/cvlabels.example
/usr/cvfs/examples/example.cfgx
/usr/cvfs/config/raid-strings

SEE ALSO

cvfs(8), **snfs_config**(5), **mount_cvfs**(8)

NAME

cvmkdir − Create a StorNext Directory with an Affinity

SYNOPSIS

cvmkdir [**−k** *key*] *dirname*

DESCRIPTION

The **cvmkdir** command creates a *StorNext File System* directory and attaches an affinity parameter (*key*) to it. If no option is used and the directory exists, the **cvmkdir** command displays the assigned affinity. Once an affinity is assigned to a directory, it cannot be altered. If no *key* is specified and the directory does not exist, the directory will not be created.

An affinity may be dissociated from a directory by specifying an empty key (e.g., "").

See **snfs_config**(5) for details about affinities to stripe groups.

OPTIONS

−k *key* Specify to the file system what affinity (*key*) to associate with the directory. All new sub-directories and files created beneath this directory inherit its affinity. If the affinity is changed or removed only files or directories created after the change are affected.

dirname

The path of the directory to be created.

SEE ALSO

cvmkfile(1), **cvaffinity**(1), **snfs_config**(5)

NAME

cvmkfile − Create a pre-allocated file

SYNOPSIS

cvmkfile [**-eprswz**] [**−k** *key*] *size*[**k**|**m**|**g**|**t**] *filename*

DESCRIPTION

cvmkfile can be used to pre-allocate a file on the StorNext file system. This is useful and preferable when preparing a file for use in a real-time or streaming environment as the entire file is represented in only one file system extent. Additionally, a file can be placed onto a specific stripe group by specifying the *key* value, which is used as the affinity locator. See **snfs_config**(5) for more details about affinities.

WARNING: This will destroy all existing data for the specified file unless the **−e** option is used.

OPTIONS

−e The **−e** option tells cvmkfile not to clobber an existing file, just expand or verify the requested space. The default behavior is to unlink and re-create an existing file (see **WARNING** above).

−k *key* The **−k** *key* optionally tells the file system where to place the data file. If an Affinity Key is specified, the file is placed on stripe groups that are specified to support this key. If there is no stripe group with the key specified, then the file is placed in non-exclusive data pools. If there are no non-exclusive data pools, then ENOSPC (no space) is returned.

−p The **−p** option forces the allocation and any subsequent expansions to be fitted "perfectly" as multiples of the **PerfectFitSize** configuration parameter. The allocation extent will always line up on and be a perfect multiple of the number of blocks specified in **PerfectFitSize**.

−r The **−r** option causes the file to be written with pseudo-random data. This can take a significant amount of time.

−s The **−s** option forces the allocation to line up on the beginning block modulus of the stripe group. This can help performance in situations where the I/O size perfectly spans the width of the stripe group's disks.

−w The **−w** option sets the file size to be equal to *size*. Without this option the blocks are allocated but the size is set to zero. **NOTE:** Unless the -z option is used, the new file will contain undefined data. Using the -w option is not recommended unless absolutely needed, and beware that it could cause some write operations to become read-modify-write operations.

−z The **−z** option causes the file to be physically zeroed out. This can take a significant amount of time.

size[**k**|**m**|**g**|**t**]

The *size* argument specifies the number of bytes, kilobytes(**k**), megabytes(**m**), gigabytes(**g**), terabytes(**t**) to allocate for the file. Multiple extents will be allocated if there is insufficient contiguous available space to satisfy the requested amount. In the event that there is not enough space to satisfy the request, the file size will still reflect the requested *size* value if the **−w** option is specified.

filename

The file to be created.

EXAMPLES

Make a file of one gigabyte with zero length. Allocate it on a stripe group that has specified the affinity key **6100_n8**.

```
rock # cvmkfile -k 6100_n8 1g foobar
```

SEE ALSO

snfs_config(5), **cvmkdir**(1)

NAME

cvmkfs − Initialize a StorNext File System

SYNOPSIS

cvmkfs [**-GF**] [**-a** *key*] [**-n** *ninode*[**k**|**m**|**g**]] [**-r**[**-e**]**[**-m**]]** [**-Q**] [**-R** *[date:]time*] [**-X]** [*file_system_name*]

DESCRIPTION

cvmkfs will initialize a *StorNext* (SNFS) file system optionally using *file_system_name* as the name. If no name is supplied, a list of file systems configured will be presented. Active file systems may not be re−initialized. The user will be prompted for a confirmation before initializing the file system.

WARNING: This will destroy **ANY** existing file system data for the named *SNFS* file system!

OPTIONS

-a *key* Set the affinity of the root directory to *key*.

-e When remaking a managed file system in preparation for restoring all metadata from a metadata archive, the -e option specifies that the FSM should restore all user file extents. When this option is not specified, files are truncated which results in them being restored from backup. Use this option when the metadata disks must be restored but all disks containing user data are intact. This option can only be used in conjunction with the -r option and is ignored when restoring unmanaged file systems.

This option also causes thin provision unmapping work to be skipped for all stripegroups that can contain user data.

-G Bypass "Press return to continue..." type prompts. These prompts are useful on Windows systems to give the user a chance to read the error message before the window disappears.

-F Force. This option has been deprecated and replaced with **-X**. It will cause the same action as that option.

-f Failure mode - do not fail if there is a configuration mismatch or other serious abnormal condition detected. Note: This option is **not** intended for general use. Use only if instructed by Quantum support. Incorrect use may result in an unusable file system.

-m When using the -r option to remake a file system in preparation for a metadata restore from the metadata archive, cvmkfs will issue an error message and exit without modifying the file system if the stripe groups are defined to hold both metadata and user data. It does this because it is possible for the restore procedure to inadvertently allocate disk space for metadata that conflicts with user data, resulting in file corruption. The -m option can be used in conjunction with the -r option to override this behavior and force cvmkfs to remake the file system despite the risk of corruption. Use this option only if instructed by Quantum support.

-n *ninode*[**k**|**m**|**g**]]
Pre-allocate *ninode* inodes.
NOTE: This option has been deprecated.

−Q This option causes cvmkfs to print qustat statistics just before exiting.

-R [date:]time
Remake the file system in preparation for restoring all metadata as it existed at the given date and time. The format for the date:time argument is yyyy-mm-dd:hh:mm:ss, for example, "-R 2016-08-24:08:00:00". If the date is not specified, then today is assumed. This option is only valid for managed file systems when *metadataArchiveDays* is set to a non-zero value in the configuration file and it cannot be used with the -e option to restore file extents. This "historical" restore will truncate all files, forcing all data to be restored from backup.

WARNING: It is highly recommended that Quantum Technical Support be contacted before using this option. If used improperly, data could be lost.

-r Remake the file system in preparation for restoring all metadata from a metadata archive. This option can only be used when *metadataArchive* is set to true in the configuration file and a metadata

archive exists that is current as of the last time the corresponding FSM was stopped.

The remake option can be useful for disaster recovery or for metadata and journal stripe group reconfiguration.

For a managed file system, the default behavior is to truncate all of the user data files with the expectation that they have been backed up to another media such as tape. The files will be reloaded when next accessed or through other storage manager actions. It is possible to override this behavior by specifying -e on a managed file system. In this case the same cautions as specified below for unmanaged file systems apply.

For an unmanaged file system, there is no backup copy of the user data. The -e option can be specified, but it is ignored and is forced on. The metadata that is restored contains the disk addresses of the user data. This means that all stripe groups that contain user data must be left completely intact. Therefore, all thin provision unmap work is skipped for all stripegroups that can contain user data.

The following statements apply to both managed and unmanaged file systems. The metadata and journal stripe groups are remade from scratch. This allows the underlying storage on these stripe groups to be replaced and stripe group attributes to be changed. Metadata stripe groups can be converted to data stripe groups. New stripe groups can be added. The journal stripe group can change.

WARNING: It is highly recommended that Quantum Technical Support be contacted before using this option. If used improperly, data could be lost or corrupted.

-T Normally on linux, **cvmkfs** opens all devices in the configuration file to check for thin provisioned devices. This is done to unmap/trim any prior mappings on those devices that are eliminated by this cvmkfs command. This "thin provision work" can be bypassed using the **-T** option.

-U Do a check for disks that are included in the file system that is being made to see if they are currently in use in another file system that is visible to the cluster. In some configurations, this may take a long time. If there are disks in use, the operation is aborted.

-X Use expert mode to automatically answer all prompts for verification. This is useful for running cvmkfs as part of a script or automated test. The failure option can be used instead, but with the failure option no configuration transformation validation is done and is therefore not recommended. With the *-X* option, all of the normal checks are performed and if an error is detected, the command exits with appropriate message and status.

FILES

*/usr/cvfs/data/**

SEE ALSO

cvfs(8), **snfs_config**(5)

NAME

cvpaths − StorNext File System Disk Discovery Filter

SYNOPSIS

/usr/cvfs/config/cvpaths

DESCRIPTION

The *StorNext File System* (SNFS) *cvpaths* file is an optional configuration file used to control and/or override the normal SNFS behavior of scanning system standard directory locations during the *disk discovery* phase that occurs during a **cvlabel** run, or from the **fsmpm** at boot/initialization time.

Normally, the directories and device name patterns scanned are:

Platform	Directory	Device Name Pattern
Linux	/dev	/dev/sd[a-z] /dev/sd[a-z][a-z] /dev/sd[a-z][a-z][a-z] /dev/md* /dev/nvme[0-9]n[0-9] /dev/nvme[0-9]n[0-9][0-9] /dev/nvme[0-9][0-9]n[0-9] /dev/nvme[0-9][0-9]n[0-9][0-9] /dev/nvme[0-9][0-9][0-9]n[0-9] /dev/nvme[0-9][0-9][0-9]n[0-9][0-9]
	/dev/mapper	/dev/mapper/mpath[0-9] /dev/mapper/mpath[0-9][0-9] /dev/mapper/mpath[0-9][0-9][0-9] /dev/mapper/mpath[a-z] /dev/mapper/mpath[a-z][a-z] /dev/mapper/mpath[a-z][a-z][a-z]
Windows	<N/A>	PhysicalDrive0 PhysicalDrive1 PhysicalDrive2 *(etc.)*
macOS (Xsan)	/dev	/dev/rdisk[0-9] /dev/rdisk[0-9][0-9] /dev/rdisk[0-9][0-9][0-9]

If a *cvpaths* file exists in */usr/cvfs/config*, then the contents of the *cvpaths* file will explicitly control which devices and/or directories will be evaluated during disk discovery. On non-Windows systems, if the *cvpaths* file is executable, then it will be executed expecting it to be a shell script that will produce the *cvpaths* syntax on standard output, otherwise it will simply be read as input.

SYNTAX

The format rules for a line in the *cvpaths* file is:

Any line beginning with "#" is considered a comment line.

Any token beginning with "#" is considered to be a comment up to the end of the line.

Blank/empty lines are ignored.

A keyword=value syntax is used.

Groups of related keyword phrases can span multiple lines.

Note, the parser capability is limited, and does not allow for any white space around the equal ("=") opera-

tor, although white space, and commas, are tolerated in all other places.

There are several keywords:

```
directory=
wildcard=
device=
usage=
hba=
lun=
capacity=
geometry=
verify=
blockdev
```

The **directory**=*path* and **wildcard**=*glob* directives do not require any of the other keywords.

The directory specified by the **directory**=*path* directive will be traversed in a manner similar to the default **disk discovery** scan mechanism. The **wildcard**=*glob* directive is used to specify a glob pattern, see **glob**(7), to match and scan for device pathnames which are then examined in the same manner as the default disk discovery scan mechanism. The directory directive is not supported on Windows. The wildcard directive is not supported on Windows or Apple OSX platforms.

Assuming that DM Multipath is in use and the Linux is presenting device names with alphabetic suffixes, the following example would add EMC powerpath devices to the Linux device scan:

```
wildcard=/dev/mapper/mpath[a-z]
wildcard=/dev/mapper/mpath[a-z][a-z]
wildcard=/dev/mapper/mpath[a-z][a-z][a-z]
wildcard=/dev/emcpower*
```

A **device**=*path* directive begins a group of keywords related to the device located at **path**.

For example:

```
device=/dev/mapper/mpathg
```

would describe exactly one disk/raid device to be scanned during disk discovery.

The device **path** is the name of the "special file". The following example describes the normal default behavior on a Windows system:

```
device=PhysicalDrive0
...
device=PhysicalDriveN
```

Where N is the number of the last physical drive accessible by the Windows system.

NOTE!

Enumerating specific device paths presumes that the same disk/raid will always appear in the host system’s hardware/device graph with the same exact name.

In most cases, this can only be accomplished by utilizing **persistent binding** methods related to the specific disk driver package.

A **verify**=*labelname* keyword may be used to verify that the device located at **path** contains the SNFS label *labelname*, for example:

```
device=/dev/mapper/mapthg verify=CvfsDisk9
```

The device named must refer to a device path that describes the entire disk. For example, on Linux systems, you should use **/dev/sdc** rather than **/dev/sdc1**.

Normally, SNFS determines from the raid controller whether a path should be considered **Active** or **Passive**.

The **usage=**[**Active**|**Passive**] keyword may be used to override the normal determination of **Active** or **Passive** path usage. The default mode is **Active**.

The **capacity=***sectors* keyword may be used to override the normal determination of the number of sectors supported by the device.

The **geometry=***cyl/tpc/spt/bps* keyword may be used to override the normal determination of the physical geometry of the device where:

cyl	is the total # of cylinders
tpc	is the # of tracks per cylinder
spt	is the # of sectors per track
bps	is the # of bytes per sector

Certain device drivers use non-conventional names, or do not support standard methods of HBA & LUN identification.

If the device driver name does not follow the host system's convention of providing HBA & LUN information, then the **hba=***#* and **lun=***#* keywords may be used to provide that information.

For example:

```
device=/dev/emcpower3 verify=CvfsDisk_30 usage=Active hba=6 lun=2
```

would configure a disk path externalized as **/dev/emcpower3**, assigning the HBA id of 6, and LUN # 2.

This line could also be written as:

```
device=/dev/emcpower3, verify=CvfsDisk_30, usage=Active, hba=6, lun=2

-or-

device=/dev/emcpower3
    verify=CvfsDisk_30,
    usage=Active,
    hba=6,
    lun=2
```

The HBA id is used by the multi-path code to collect devices together according to which host HBA is used for access.

The actual value of the number is not critical, what is important is that all disks/raids configured through a specific host HBA should be assigned a consistent number that is unique to that host HBA path.

The LUN number is important if the raid controller is one of the controllers recognized by SNFS as capable of Automatic Volume Transfer, and Active/Passive path declaration. The LUN # is used to index into specific raid controller mode pages.

A Linux example, forcing certain paths to be held back for **Standby** usage:

```
device=/dev/sdg usage=Active verify=CvfsDisk_30
device=/dev/sdh usage=Passive verify=CvfsDisk_31
```

A Linux example, restricting the search to a non-standard directory

```
directory=/dev/rdcs    # DataCore
```

A Linux example, providing HBA & LUN information:

```
device=/dev/emcpowerf hba=4 lun=2 verify=CvfsDisk_30
device=/dev/emcpowerg hba=4 lun=3 verify=CvfsDisk_31
```

Generally, a SCSI interface is required for StorNext. On Linux, however, some devices with only a block device interface can be used. The keyword **blockdev** can be added to a line to force the block device interface to be used instead of the raw SCSI interface.

```
device=/dev/vgca0_vha1 blockdev
```

Note that in previous releases of StorNext, NVMe devices required a cvpaths file. However, by default, these devices are now automatically discovered and their device files are assumed to be block devices. So, generally, using a cvpaths file is not required to use NVMe or NVMe-oF disks.

Also note that Linux supports two forms of multipathing for NVMe-oF devices: native NVMe multipathing and DM Multipath. When the former is enabled, StorNext will directly access the /dev/nvme* device files. When the latter is enabled, StorNext will instead use device files in /dev/mapper. Therefore, when using **wildcard** and **device** entries for NVMe disks, it is important to use a /dev/mapper prefix when DM Multipath is enabled, and to use a pattern containing a /dev/nvme prefix, otherwise.

For Linux, StorNext also supports ram disk devices. Ram disk devices use the block interface but have additional special handling requirements. A Linux ram disk device can only be used as a local disk. Because it cannot be shared, a Linux ram disk device may only be useful as a metadata device. Because the data in a ramdisk is not persistent across boots, a Linux ram disk may only have practical application for development.

A Linux ram disk is identified by its path name starting with **/dev/ram**. The following example shows two ways to configure a Linux ram disk.

```
device=/dev/ram0

-or-

wildcard=/dev/ram*
```

Linux also supports Ceph Rados block devices and XEN virtual disks, XEN virtual disks are the presentation used by Amazon Elastic Block Storage (EBS). Note that EBS volumes are not shareable and can only be accessed from a single EC2 instance at once. Ceph Rados block devices can in theory be shared between hosts if configured without caching.

Ceph devices are recognized based on the device special file starting with **/dev/rdb**. XEN virtual disks are recognized by the name **/dev/xvd**. As with ram devices the device or wildcard specifiers can be used to include them in the device scan.

FILES

/usr/cvfs/config/cvpaths

SEE ALSO

cvfs(8), **snfs_config**(5), **fsm**(8), **fsmpm**(8)

NAME

cvtune − A program to examine and/or dynamically modify options of a running stornext client. The changes will be lost when stornext is restarted or the filesystem is unmounted.

SYNOPSIS

cvtune -a [**−F** *filesystem*] [**−r** *auto_dma_read_len*] [**−w** *auto_dma_write_len*]

cvtune -b [**−S** *cachebufsize*] [**−l** *bufferlowdirty*] [**−h** *bufferhighdirty*]

DESCRIPTION

cvtune is a tool for system administrators to examine and/or dynamically modify options for *StorNext* clients such as *StorNext File System* (SNFS) mount options and buffer cache values.

A file system can be mounted with options auto_dma_read_len and auto_dma_write_len, which specify the minimum transfer size used for performing direct DMA I/O instead of using the buffer cache for well-formed reads. These values are specified in megabytes and can be modified for an individual file system. These changes are temporary until the next mount of the file system.

A file system can be mounted with the buffer cache options bufferlowdirty and bufferhighdirty, which specify the levels at which buffer cache flushing start and stop. These values can be modified for a particular buffer cache. Note that the buffer cache may be shared across file systems which use the same cachebufsize. The buffer cache is identified by the cachebufsize in kilobytes. The bufferlowdirty and bufferhighdirty are in megabytes. These changes are temporary until stornext is restarted.

WARNING: Modifying these options can have a **substantial** performance impact. Only use when recommended by Quantum Support.

cvtune can be used to:

List the current values for auto_dma_read_len and auto_dma_write_len for a file system.

Modify the current values for auto_dma_read_len and auto_dma_write_len for a file system.

List the current values for bufferlowdirty and bufferhighdirty for a particular cachebufsize.

Modify the current values for bufferlowdirty and bufferhighdirty for a particular cachebufsize.

OPTIONS

-a Display or modify the values of auto_dma_read_len and auto_dma_write_len for the file system.

-b Display or modify the values of bufferlowdirty and bufferhighdirty for the buffer cache.

-F *file system*
Identify the file system by name.

-h *bufferhighdirty*
The value in megabytes (MB) at which the background buffer flushing is initiated. Background buffer flushing is initiated at bufferhighdirty and continues until bufferlowdirty is reached.

-l *bufferlowdirty*
The value in megabytes (MB) at which the background buffer flushing is terminated. Background buffer flushing is initiated at bufferhighdirty and continues until bufferlowdirty is reached.

-S *cachebufsize*
Identify the buffer cache in kilobytes (KB).

EXAMPLES

To examine the current buffer cache settings for the buffer cache containing 1024k buffers.

cvtune -b -S 1024

To set the current buffer cache settings for bufferhighdirty and bufferlowdirty to 768M and 700M respectively. The buffer cache is identified by a cachebufsize of 1024k.

cvtune -b -S 1024 -h 768 -l 700

To set the current buffer cache settings for bufferhighdirty and bufferlowdirty to 500M and 400M respectively for the buffer cache using 256k cachebufsize.

cvtune -b -S 256 -h 500 -l 400

To set the current buffer cache settings for bufferhighdirty and bufferlowdirty to 500M and 400M respectively for the first buffer cache found. This can be used when there is 1 filesystem mounted and thus only 1 buffer cache.

cvtune -b -h 500 -l 400

To examine the current auto_dma_read_len and auto_dma_write_len for the snfs1 file system.

cvtune -a -F snfs1

To set the current auto_dma_read_len and auto_dma_write_len for the snfs1 file system to 2M for both.

cvtune -a -F snfs1 -r 2 -w 2

SEE ALSO

mount_cvfs(8)

NAME

cvupdatefs − Commit a StorNext File System configuration change

SYNOPSIS

cvupdatefs [**-bdfFGhlnv**] [**-c** *pathname*] [[**-M**] **-R** *NewFsName*] [*FsName*] [*FsPath*]

DESCRIPTION

The **cvupdatefs** program is used to commit a configuration change to a StorNext file system. Possible configuration changes include stripe group list modification as well as file system journal modification.

The file system update program must be run on the machine that the File System Manager (FSM) is running on. This utility reads the configuration file and compares the configuration file against the current on-disk metadata configuration. If there are differences between the configuration and the on-disk metadata, the utility will display what changes need to be made to bring the file system metadata up to date.

NOTE: All metadata modification must be made on a stopped file system. It is recommended that the file system is stopped and **cvfsck**(8) has been run before making any changes to a file system configuration. Maintaining a backup of the original file system configuration file is also strongly recommended.

When a successful update is completed, the new configuration file is stored in the on-disk metadata and the previous one is saved in */usr/cvfs/data/*<file_system_name>*/config_history/*.*cfgx.*<TIMESTAMP>

OPTIONS

−b Build info - log the build information.

-c *pathname*
Provide a specific path to the previous configuration file that is to be used. This option is used to force **cvfsck** to be run as a sub-process to insure that the file system meta data is consistent prior to doing a capacity or stripegroup expansion, or any journal changes.

-d Debug - use to turn on internal debugging only.

-F Force. This option has been deprecated and replaced with **-y**. It will cause the same action as that option.

-f Failure mode - do not fail if there is a configuration mismatch or other serious abnormal condition detected. Note: This option is **not** intended for general use. Use only if instructed by Quantum support. Incorrect use may result in an unusable file system.

−G Pause - pause the program after displaying the exit status (Windows only.)

−h Help - print the synopsis for this command.

-l Log - log when the update finished.

-n Read-only - set metadata to read-only mode.

-M Allow managed file systems with -R option.

-R *NewFsName*
Rename - Provide a new file system name to rename an existing unmanaged file system. Use with **-M** to rename a managed file system.

The existing config file will be renamed, and the existing data directory containing logs will be migrated to the new name. See the section below for further details about using this option.

-s Slice rebuild - Rebuild the free slice trees to their optimal sizes. When extending the LUNs in a stripegroup, by default it will just add enough additional slices to hold the additional free space. When the **-s** option is given, it will instead rebuild the slice trees, which usually results in larger slices. If the LUNs have been previously extended, this option will allow the slice trees to be rebuilt without extending the LUNs.

-T When adding stripe groups, do not perform TRIM/UNMAP. This option is Linux-specific.

-U When a stripegroup is added, do a check for disks that are included in the stripegroup that is being added to see if they are currently in use in another file system that is visible to the cluster. In some configurations, this may take a long time. If there are disks in use, the operation is aborted.

−v Verbose - turn on verbose reporting methods.

−y Yes - Bypass the prompt and answer yes to the basic warning about proceeding. If the prompt warning is for an unusual condition, this option will not bypass that prompt.

−W Do not use copy-on-write (COW) when applying changes.

Once the file system configuration has been changed to reflect the stripe group or journal changes the **cvupdatefs** utility may be run. When **cvupdatefs** is run it will display a listing of stripe groups which will be modified, followed by a prompt. If this list accurately reflects the changes made to the configuration file then answering 'yes' at the prompt will allow the utility to make the needed changes.

Once the utility has completed, the file system may be started again. After starting the file system, the 'show' command in **cvadmin**(8) may be used to verify the new stripe groups. The 'show' command will list all of the stripe groups on the file system, including the newly created stripe group(s). Also, if the location of the file system journal has changed this too will be reflected by the cvadmin command 'show'.

WARNINGS

It is very important that the consistency of the file system be correct before cvupdatefs is run. If the file system has a bad state cvupdatefs could introduce data corruption. It is recommended that cvfsck is executed on the file system before any changes are made. If cvfsck does not finish with a clean file system do not make any configuration changes until the file system is clean.

ADDING A STRIPE GROUP

The first step in adding stripe groups is to modify the file system's configuration file to reflect the desired changes. For notes on file system configuration format refer to **snfs_config**(5). In addition to adding StripeGroup configuration entries, associated Disk and DiskType entries for any new disks must be included.

Currently the ordering of stripe groups in the configuration file and in the metadata must match. Thus, when adding new stripe group configuration entries to the configuration file they must always be added to the end of the StripeGroup configuration section. **cvupdatefs** will abort if a new stripe group is detected anywhere but the end of the file.

INCREASING THE STRIPE DEPTH OF AN EXISTING STRIPE GROUP

Warning: This option is not recommended and its use is deprecated. Adding a new stripe group is the recommended way to expand capacity of a file system.

The stripe depth is the number of disks in the stripe group and is a key factor in the amount of parallel I/O that can be accomplished. This choice should ideally be made before the file system is created, thus eliminating the need for cvupdatefs to modify this value by adding disks to the stripe group. Consult the StorNext File System Tuning Guide for information on configuring for optimal file system performance.

Warning: When a stripe group is populated with file data, adding disks will increase free space fragmentation of the stripe group proportional to the amount of pre-existing file data. It is important to avoid fragmentation, which severely impacts performance and functionality of the file system. If the stripe group contains little or no file data, expansion will not result in free space fragmentation. The snfsdefrag utility can be used to relocate pre-existing file data to a different stripe group.

When new disks are added to an existing stripe group the new disks must exactly match the existing disks in size. All new disks must be added to the end in the disk list in the configuration file StripeGroup section.

New disks cannot be added to a stripe group containing metadata or journal. A new stripe group must be added if additional capacity or performance is needed for metadata or journal operations. The cvupdatefs utility can be used to relocate the journal to a new stripe group.

MODIFYING FILE SYSTEM JOURNAL CONFIGURATION

cvupdatefs will also detect changes in the journal configuration and modify the metadata accordingly. Journal changes include moving the journal to a new stripe group and increasing or decreasing the size of the journal.

JournalSize
: (Located in the Global section) Modifying this value will change the size of the on-disk journal.

Journal
: (Located in the Stripe Group section) Setting this entry to yes will place the on-disk journal on the given stripe group.

NOTE:

There may only be one journal stripe group per file system.

REMOVING A JOURNAL-ONLY STRIPE GROUP

For Linux MDCs, if a stripe group has only the journal attribute, i.e. no metadata and no userdata, and the journal is moved to another stripe group, the former journal-only stripe group is left with no attributes pertaining to content type. If it is desired that this stripe group be retired and the disks used for other purposes, you can set the status to down after the journal is moved. Note that the status must be up during the journal move operation because the journal recovery must be executed prior to moving the journal.

The behavior is similar on Windows MDCs, except that there is no explicit userdata attribute in the ASCII config file. This means that with no journal and no metadata, userdata is assumed. If the desire is to retire the former journal-only stripe group, care should be taken to not run the file system after moving the journal off of the stripe group. Set the status to down immediately after moving the journal and before starting the FSM.

CORRECTING MISCONFIGURED STRIPE GROUPS

cvupdatefs has a limited ability to address configuration errors. For example, if a stripe group was added but the configuration file shows incorrect disk sizes, this option could be used to rewrite that stripe group. Metadata and Journal stripe groups cannot be rewritten. In addition, data only stripe groups that may be overwritten must be empty.

The types of changes that can be made to a stripe group are as follows

```
1) Resize disk definitions in a stripe group
2) Modify stripe breadth in a stripe group
3) Modify the disk list in a stripe group
```

Warning: Always use this option with extreme caution. Configuration errors could lead to data loss.

RENAMING A FILE SYSTEM

Warning: Renaming a file system that is managed requires additional steps documented elsewhere. These are in the StorNext documentation center. When following those instructions, the -M option must be used when invoking this command with the -R option. Otherwise, renaming a file system is only allowed on an unmanaged file system. Without the -M option, if **cvupdatefs**(8) detects that the file system is managed, it will print an error message and exit without doing the rename.

The **-R** option for renaming a file system should be used with care, as there are several things that get modified as part of this process. Before renaming a file system, it is highly recommended that **cvfsck**(8) be run prior to renaming the file system. The file system must be unmounted on all SAN and DLAN clients, and the file system stopped, see **cvadmin**(8). If a client has the file system mounted when it is renamed, the client might need to be rebooted in order to unmount the old file system name. On Windows, use the Client Configuration Tool to unmount file system before renaming it.

The file system that is being renamed will have been configured in one of three modes: non-HA, HA or manual HA, and how it was configured will change how to rename the file system.

Non-HA mode
: There are no extra steps needed when renaming a file system that is not in HA mode.

HA mode

When a file system is being used in HA mode, prior to running the rename command on the primary, on the secondary the */usr/cvfs/data/*FsName directory should be manually renamed to */usr/cvfs/data/*NewFsName. When the rename command is then run on the primary, the HA sync processes will propagate all the other configuration changes to the secondary. Wait for the HA sync to complete before continuing.

Manual HA mode

In manual HA mode, the rename command should be run on both MDCs. When run on the second MDC, **cvupdatefs**(8) will recognize that the name in the ICB has been changed, but will proceed if *NewFsName* is the same as the name in the ICB. In manual HA mode there is no need to manually rename */usr/cvfs/data/*FsName since that will happen as part of running **cvupdatefs -R** on the second MDC.

After changing the name of a file system, the change needs to be manually reflected in the */etc/fstab*, */etc/vfstab* or */etc/vstab* files on all the clients before they remount the file system and the corresponding directories renamed or created. Windows StorNext SAN and DLAN Clients mounts will need to be remapped. Run the Client Configuration Tool to re-map the mount with new file system name.

For any client that is operating as a StorNext File System Proxy Client, check to see if it has a */usr/cvfs/config/dpserver.*FsName file. If it does, it will need to be renamed to */usr/cvfs/config/dpserver.*NewFsName.

If something goes wrong during the rename operation, **cvupdatefs**(8) will revert any partial changes, but it is still possible that in some corner cases it will not be able to fully revert the changes, and manual intervention will be required. Files that are modified and/or renamed during the rename operation include:

*/usr/cvfs/data/*FsName
*/usr/cvfs/data/*NewFsName
*/usr/cvfs/config/*FsName.*cfgx*
*/usr/cvfs/config/*NewFsName.*cfgx*
/usr/cvfs/config/fsmlist

as well as the ICB in the file system itself. The OS dependent files that need to be manually updated include:

/etc/fstab
/etc/vfstab
/etc/vstab
Windows registry via the Windows Client Configuration Tool

ENABLING CASE INSENSITIVE

If a change in the file system configuration is detected such that case insensitive is being enabled, **cvupdatefs** invokes **cvfsck** as a sub-process to check for name collisions. If name collisions are detected, the update operation will be aborted. It is strongly recommended that **cvfsck -A** be run prior to attempting the change using **cvupdatefs**.

EXIT VALUES

cvupdatefs will return one of the following condition codes upon exit.

```
0 - No error, no changes made to the file system
1 - No error, changes have been made to the file system
2 - Configuration or file system state error, no changes made
3 - ICB error, improper file system found, no changes made
4 - Case conversion found name collisions, no changes made
```

NOTES

IMPORTANT: It is highly recommended to run **cvfsck**(8) prior to making any configuration changes.

By default, cvupdatefs uses a copy-on-write (COW) store and applies changes to metadata at the very end. This is beneficial for performance and allows easier recovery if any issues are encountered that prevent successful completion. When COW is enabled, temporary space is typically consumed from /tmp or similiar,

depending on the platform. However, the temporary directory can be set using the TMPDIR environment variable. As noted above, COW can be disabled using the -W option in which case no temporary space is used.

FILES

/usr/cvfs/config/.cfgx*
/usr/cvfs/data/<file_system_name>*/config_history/*.cfgx.*<TIMESTAMP>

SEE ALSO

snfs_config(5), **cvfsck**(8), **cvadmin**(8)

NAME

cvversions − Display StorNext client/server versions

SYNOPSIS

cvversions [-h] [-F *type* **] [[***id***:]***file* **...]**

DESCRIPTION

cvversions will display the revision, build level and creation date for the *File System Manager (FSM)* and client subsystems of the *StorNext File System*.

This information should be submitted to Quantum Technical Support when contacting them about any problems found in the file system.

OPTIONS

-h Print a help/usage message and exit.

-F *type* If *type* is **text**, cvversions emits version information in text format (this is the default). If *type* is **json**, cvversions emits limited version information in a parsable JSON format.

file *file* is searched for a StorNext version string and, if found, is printed in addition to the standard version strings. If present in the argument, *id* is substituted for *file* in the program output.

USAGE

Simply execute the program and record the information shown.

SEE ALSO

StorNext File System Release Notes

NAME

deviceparams − StorNext Linux device turning parameters

SYNOPSIS

/usr/cvfs/config/deviceparams

DESCRIPTION

The StorNext File System (SNFS) **deviceparams** file provides a way to tune the various queueing parameters of the linux block devices. The file can specify values to override the default maximum I/O size submitted to devices and to select the elevator algorithm used.

Note: Changing Linux elevator parameters can have a severe negative performance impact when inappropriate values are specified. Use this file only when recommended by Quantum Support.

SYNTAX

If a **deviceparams** file exists in the SNFS 'config' directory it is used to tune the various queueing parameters of the devices used for SNFS disk I/O. The format of the **deviceparams** file consists of <parameter>=<value> pairs that can be prefixed with the ssd=1 if the parameter is specific to SSD devices or ssd=0 if the parameter is for regular disk devices. Comments starting with pound-sign (#) are skipped.

```
[ssd=[0|1]] scheduler=[none|mq-deadline|bfq|kyber]
[ssd=[0|1]] nr_requests=<depth>
[ssd=[0|1]] max_sectors_kb=<value>

Note: On RHEL 7 'no elevator' scheduler was 'noop' and mq-deadline scheduler
was 'deadline'. Other choices on RHEL 7 are anticipatory and cfq.
```

Where <*depth*> is the maximum depth of the request queue for each block device and <*value*> is the maximum size of I/O the elevator should send to the device in kbytes. Note that other factors such as the physical fragmentation of memory buffers may restrict I/O to a lower limit than that specified. Using hugetlb pages in an application is one way to overcome this.

The parameters in the **deviceparams** file are applied when the disk devices are scanned. A **cvadmin** -e "disks refresh" command is needed to apply new parameters to a running StorNext client. Because these parameters change the Linux block device characteristics, each Linux StorNext client will need a **deviceparams** file.

The I/O scheduler choices may vary by Linux distribution. Currently, the more popular Linux distributions have four choices for I/O schedulers; they are **none, mq-deadline, kyber**, and **bfq**. Each scheduler then has additional tunable parameters that can affect I/O performance. There is no guarantee that the schedulers and their tunable parameters will be available in future releases.

There are no Linux man pages describing these tunable parameters. There are however documents included with the Linux source (Documentation/block) and many papers that can be found online which discuss various attributes of the "Linux IO Scheduler". Again, the **deviceparams** file should be used only when working with Quantum Support.

EXAMPLE

In many instances Quantum has found improved performance when using the **mq-deadline** and **deadline** schedulers. In addition, increasing the block queue depth has been seen to improve performance.

Red Hat 8 example:

scheduler=mq-deadline
nr_requests=256

Red Hat 7 example:

scheduler=deadline
nr_requests=4096

To select different parameters for SSD drive you can use the ssd=[0|1] prefix. The following Red Hat 8 ex-

ample does apply these changes:

```
# Set scheduler to none on SSD and mq-deadline on regular
# StorNext devices.
# Set the max_sectors_kb to 1024 on SSD.
# Set the nr_requests to 256 on all StorNext devices.
ssd=1 scheduler=none max_sectors_kb=1024
ssd=0 scheduler=mq-deadline
nr_requests=256
```

FILES

/usr/cvfs/config/deviceparams
/usr/cvfs/examples/deviceparams.example

NAME

disk_license − Disk License Tool

SYNOPSIS

disk_license [**−hHqrsvX**] [**−d**n][**−f***filename*][**−H**][−F*format*]

disk_license [**−i***diskcatalog***]**

DESCRIPTION

disk_license is used to interrogate disk drives, StorNext disk licenses and the Quantum Disk Catalog. It can report current disk licensing status and create reports for manually requesting disk licenses on an MDC.

All the disks in all of an MDC's file systems are categorized into Quantum Branded, Quantum Certified, and Uncertified (all other) as defined in the quantum_disk_catalog.dat database.

The Disk Usage licenses are free, but aid Quantum support in understanding the usage and requirements of a site.

OPTIONS

-c <*filename*>
Use catalog <*filename*> instead of the default disk-license-request report file.

-C Only print the Disk Catalog to standard out.

-d <*n*> Print additional details on a per−file−system basis. Larger values for <*n*> yield more details, up to a point.

-f <*filename*>
Send reports to the file <*filename*>.

-F *format*
Print the output in format <*format*>. Supported formats are: *ascii*, *xml*, and *json*.

-h Print the command's usage message and quit.

-H Display capacity values in human readable, terabyte values (only for ASCII format).

-q Quiet mode. Suppress status messages to standard out.

-R <*filename*>
Input a Disk License Request report from <*filename*>. This option is not available on some platforms.

-r Print a Disk License Request report to standard out.

-s Print a Disk Usage Summary report to standard out.

-u Update the XML file specified with the **-R** option to the current version and format.

-v Print additional internal details as the command collects the information to categorize the MDC's disks. Additional **v** options increase the amount of detail.

-X Update the default Disk License Request file. When Disk Usage is over licensed capacity, create a periodic RAS message.

-i *diskcatalog*
Import the specified disk catalog when it has a newer generation number than the currently installed disk catalog.

FILES

/usr/cvfs/config/quantum_disk_catalog.dat
/usr/cvfs/config/license.dat
/usr/cvfs/config/.cfgx*

Default Disk License Request report file:
/usr/cvfs/debug/quantum_disk_license_report.xml

SEE ALSO

NAME

domainsid − StorNext File System Domain SID File

SYNOPSIS

/usr/cvfs/config/domainsid

DESCRIPTION

The *StorNext File System* (SNFS) *domainsid* file is an optional MDC configuration file used to set the domain SID to be used with ACLs when the Security Model is set to "acl" in a file system configuration file and Unix Identity Mapping is set to "algorithmic"

Note that this file should only be deployed in very specific use cases. For example, if an environment uses Open Directory, the *domainsid* file should contain the assigned Domain SID. This can be determined by running the following command on a Mac:

$ dsmemberutil getsid -U *username*

where *username* is the name of any regular user account in Open Directory. This will return a string such as the following:

S-1-5-21-2553502104-2799725507-638401443-3106

The Domain SID is the string without the trailing RID so in this example, it has the value **S-1-5-21-2553502104-2799725507-638401443** The following command may be run on the MDC to set this domain SID.

mdc# echo S-1-5-21-2553502104-2799725507-638401443 > /usr/cvfs/config/domainsid

After configuring the *domainsid*, file systems must be restarted on the FSM to have it take effect.

Note: Improper configuration of this file may lead to files having invalid ACLs and permissions not being enforced properly.

FILES

/usr/cvfs/config/domainsid

SEE ALSO

cvfs(8), **snfs_config**(5),

NAME

dpserver − StorNext File System Proxy Client Configuration

SYNOPSIS

/usr/cvfs/config/dpserver

/usr/cvfs/config/dpserver.*FsName*

DESCRIPTION

The StorNext File System (SNFS) **dpserver** file is a configuration file used to control the SNFS Stornext Distributed LAN Server (also called Proxy Server or Gateway) on Linux systems. This file is required in order to start a Proxy Server and is consulted when a **mount** command specifies the Proxy Server option.

The **sndpscfg** command is normally used to generate and maintain **dpserver** files on non-Windows systems - see **sndpscfg**(8) for details. To view and adjust the Proxy Server settings on Windows systems, use the LAN Client/Gateway tab in the Client Configuration tool instead.

SYNTAX

At minimum, the **dpserver** file specifies the network interfaces to use for Proxy Server. It can also be used to override various Proxy Server tuning parameters.

There can be both file-system-specific **dpserver.***FsName* files and a default **dpserver** file. If a file-system-specific **dpserver.***FsName* file exists, it will be used in preference to the default **dpserver** file.

The format rules for a line in the **dpserver** file are:

Any line beginning with "#" is considered a comment line.

Blank/empty lines are ignored.

There are several keywords:

```
interface ifname [address ipaddr]
transfer_buffer_size_kb n
transfer_buffer_count n
server_buffer_count n
tcp_window_size_kb n
daemon_threads n
server_conn_count n
```

The keywords are interpreted as follows:

The **interface** keyword specifies the name of a network interface (e.g., **eth0**) to use for Proxy Client traffic. If a NIC has been assigned only one IP or IPv6 address, then only the interface name needs to be specified. If the NIC has been assigned more than one address (multiple IP addresses, multiple IPv6 addresses, or both IP and IPv6 addresses), then the **address** keyword and one IP or IPv6 address should also be specified. If a single **interface** entry exists for the NIC and the address is omitted, the "main" address assigned to the interface is used and other assigned addresses are ignored. Finally, if an **interface** entry exists with the address omitted and another interface entry exists for the same NIC containing an address, the specified address is used.

At least one **interface** keyword must be specified in the file in order for a Disk Proxy Client Server to be started.

The remaining keywords are used to override the default values for tunable parameters. Note that these values are propagated from the Proxy Servers to the Proxy Clients , and thus can affect the behavior of both. Note also that not all tuning parameters affect all platforms.

The optional **transfer_buffer_size_kb** keyword specifies the size in Kilobytes of the socket transfer buffers used for Proxy Client I/O. The default value is 256 and values between 32 and 4096 are allowed.

The optional **transfer_buffer_count** keyword specifies the number of socket transfer buffers used per connection for Proxy Client I/O. Note that this parameter is not used on Linux Proxy Servers or Clients. How-

ever, it is used by Windows Proxy Clients, and the value is passed to them by Linux Proxy Servers. The default value is 16 and values between 4 and 128 are allowed.

The optional **server_buffer_count** keyword specifies the number of I/O buffers that will be allocated per network interface on the Proxy Server. The default value is 24 and values between 4 and 512 are allowed.

The optional **tcp_window_size** keyword specifies the size in Kilobytes of the TCP window used for Proxy Client I/O connections. The default value is 0 and values between 0 and 16384 are allowed. The setting of 0 has a special meaning, which is that no change is made to the default system value. This allows Linux autotuning to adjust the receive buffer size and TCP window size dynamically for each connection. Quantum recommends this setting when autotuning is enabled, which is the default for recent Linux versions.

The optional **daemon_threads** keyword specifies the number of kernel threads on the server that will be used to service Proxy Client I/O requests. The default value is 8 and values between 2 and 256 are allowed.

The optional **server_conn_count** keyword controls how many connections a client makes to each gateway. This allows the use of multiple TCP links over a single physical subnet rather than having to use multiple subnets and physical NICs. The default is 1 and up to 16 connections may be configured. This parameter requires updated logic on the MDC, gateways and DLC clients. If any of these is backrev code, the client will only use one connection per network.

HA ENVIRONMENTS

If you choose to configure the Distributed LAN Server on a StorNext cluster running in High Availability (HA) mode, each HA node must have its own dpserver files detailing the NICs on that node. The **dpserver** files are not synchronized between HA pairs.

If the Distributed LAN Server is configured after converting to HA, the file system(s) running as Distributed LAN servers must be unmounted and mounted again to service DLC requests.

In StorNext startup and failover situations, the VIP is dynamically associated with a physical address on the Primary server. Do not use VIP interfaces when setting up the dpserver configuration file, or it will not be available when the node is running as Secondary. The physical interface and IP address should be used in this situation.

EXAMPLE CONFIGURATION FILE

A very basic dpserver configuration file

```
interface eth0
```

A basic multi-interface dpserver configuration file

```
interface eth0
interface eth1
interface eth2
interface eth3
```

A more complex dpserver configuration file

```
interface eth1 address 10.3.21.2
tcp_window_size_kb 64
transfer_buffer_size_kb 256
transfer_buffer_count 16
server_buffer_count 8
daemon_threads 8
```

LIMITATIONS

Only the **Linux** platform is supported with dpserver and sndpscfg commands

To view and adjust the Proxy Server settings on Windows systems, user the LAN Client/Gateway table in the Client Configuration tool instead.

FILES

/usr/cvfs/config/dpserver
*/usr/cvfs/config/dpserver.*FsName

SEE ALSO

mount_cvfs(8), **sndpscfg**(8)

NAME

fsforeignservers − StorNext File System Foreign Server List

SYNOPSIS

/usr/cvfs/config/fsforeignservers

DESCRIPTION

The *StorNext File System* (*SNFS*) **fsforeignservers** file contains a list of systems acting as *File System Foreign Server(s)*. The file system *Foreign Servers* provide StorNext File System Services (*FSS*) to computers that do not belong to the *SNFS* cluster of the nodes acting as *Foreign Servers*. A **fsforeignservers** file is not required, and can be used in place of or in addition to the **fsnameservers** file. A *Foreign Server* will present only those file systems that are hosted locally on the *Foreign Server*.

Note: Instead of using **fsforeignservers**, you may now configure multiple clusters. This is a more efficient and flexible way of configuring groups of file systems. Clients can now configure access to multiple clusters by putting the addresses and cluster names of the NSS coordinators for all clusters into the **fsnameservers** file. See the man page for **fsnameservers**.

A *Foreign Client* is a computer that is accessing a StorNext File System via a Foreign Server. Foreign Clients do not belong to the *SNFS* cluster per se, meaning that they cannot activate a file system or vote in elections.

Typically, a **fsforeignservers** file entry is created on the client to provide the client computer access to a specific StorNext File System on a specific host. This connection is limited to the StorNext File Systems local to that host. If additional *SNFS* access is desired, an entry for the machine(s) hosting the desired file system must be added to the **fsforeignservers** file. For failover configurations, both primary and secondary systems must be specified. This file is needed only on the *Foreign Client*, it is not needed on the *Foreign Server*. To determine what *SNFS* services are available, use the **cvadmin**(8) command with the **-H** *host* option to view active file systems for each host specified in the **fsforeignservers** file.

The *Foreign Client* will use the IP address of the node that returns the address of the active FSM as the address to connect to for meta data traffic for that file system.

SYNTAX

The format for the **fsforeignservers** is simple. A **fsforeignservers** file contains one entry per line. It contains the IP address or hostname to use as a *Foreign Server*. The use of IP addresses is preferred to avoid problems associated with lookup system (eg., DNS or NIS) failures. The format of an **fsforeignservers** line is:

```
IP address
```

or

```
HostName
```

Where *HostName* is a host name or *IP address* of a host that can service queries for File System Services for file systems hosted directly on that host.

Lines that contain white space only or that contain the comment token as the first non-white space character are ignored.

FILES

/usr/cvfs/config/fsforeignservers

SEE ALSO

cvadmin(8), **cvfs**(8), **dpserver**(4), **fsm**(8), **fsmpm**(8), **fsnameservers**(4), **mount_cvfs**(8)

NAME

fsm − StorNext File System Manager

SYNOPSIS

fsm [*file_system_name* [*cluster_admin] [*host_name*]]*

DESCRIPTION

The **fsm** is the server daemon that manages a StorNext File System (SNFS) The File System Manager (FSM) manages the file system's name space, allocations, and metadata coherency. It is also used for I/O bandwidth and stripe group management functions. The default SNFS file system name is *default*, and the default host name is the system's hostname as found by the **gethostname**(2) library call.

The optional *cluster_admin* argument has the syntax:

@*cluster_name*/*admin_domain*

The **@** distinguishes the *cluster_admin* argument from the *host_name* argument.

Multiple FSM processes may co-exist on one system, as long as they have unique file system names. The file system name is used by the **mount**(8) command, along with the hostname separated by a colon (:). For example, if an FSM process was started on host *fsmhost* and the file system name was declared *projecta* and the mount point was */usr/clips*, then the mount command would be:

```
mount -t cvfs fsmhost:projecta /usr/clips
```

This process runs in the background and is started at boot time. It is enabled or disabled via **chkconfig**(8) or **init.d**(7) using the *cvfs* key word.

To start multiple FSM daemons (therefore multiple file systems) on a single system the **fsmlist** file must be created to describe which FSM daemons to start. See **fsmpm**(8) and **fsmlist**(4) for details.

ENVIRONMENT

FSM_KEEP_ALIVE_TIME

This variable can be used to change the rate that the FSM process sends a keep alive message to each connected client. The value is in seconds, with a default of 5 seconds. It can be set between 1 and 7200 seconds (2 hours).

Note: This variable is not intended for general use, and should only be used when recommended by Quantum Support.

FSM_KEEP_ALIVE_TIMEOUT

This variable can be used to change the timeout value that the FSM process uses after sending a keep alive to a client. The value is in seconds, with a default of 3 seconds. It can be set between 1 and 30 seconds. Note that there are other factors that are also considered by the FSM process before timing out a client, so the actual timeout may be somewhat longer.

Note: This variable is not intended for general use, and should only be used when recommended by Quantum Support.

FILES

/usr/cvfs/config/.cfgx*
/usr/cvfs/config/fsmlist
/usr/cvfs/config/license.dat
/usr/cvfs/data/<file_system_name>*/config_history/*.cfgx.*<TIMESTAMP>

SEE ALSO

cvfs(8), **snfs_config**(5), **fsmlist**(4), **fsmpm**(8), **fstab**(5), **mount**(8)

NAME

fsmcluster − StorNext File System cluster configuration

SYNOPSIS

/usr/cvfs/config/fsmcluster

DESCRIPTION

The *StorNext File System* (*SNFS*) *fsmcluster* file describes cluster configuration information for the **fsmpm**(8) daemon and **mount_cvfs**(8) command. It is a list of keywords with values.

OVERVIEW

Traditionally, StorNext file systems have been grouped into clusters which share a common name service providing visibility to clients. The clients would typically be able to mount these file systems using either direct SAN connections to the data LUNs or proxy access using the distributed LAN client connection to a StorNext gateway server. If a set of clients require access to a file system in a different cluster, an **fsforeignservers**(4) may be configured.

The StorNext cluster and administrative domain feature allows more flexible configurations where sharing access between clusters is desired. Conversely, this feature may also be used to separate large clusters into multiple smaller clusters where sets of file systems may be grouped with clients that access only those file systems.

Multiple file systems with the same name may coexist, provided they reside in separate clusters.

SYNTAX

default_cluster *cluster*

Specify the default cluster name for this machine. If not specified, it will default to **_cluster0**.

default_addom *admin_domain*

Specify the default administrative domain name for this machine. An administrative domain is a group of named clusters. If not specified, it will default to **_addom0**.

Lines that contain white space only or that contain the comment token as the first non-white space character are ignored.

The default administrative domain name and default cluster names are used when an fsm name is specified without a cluster name or administrative domain name, such as when mounting a StorNext File System (SNFS)

Changing the Cluster or Administrative Domain Name

Changing the cluster name on an MDC node affects the location of the file systems running on that node. Changing the cluster name on a coordinator node affects the services that are being advertised by that coordinator. All clients must unmount the file systems that are to be affected by the change. Services must be stopped on the MDC and coordinator nodes, including both primary and secondary nodes of an HA pair. The *fsmcluster* file may then be updated on all affected nodes including both nodes of an HA pair. Start the services on the coordinator nodes first if these nodes are different than the MDC nodes. Start the services on the MDC nodes next. The file systems are now available for client mounts at their new cluster location.

When converting a system to HA, if the first node has an *fsmcluster* file, the second node must have one as well, prior to attempting the conversion.

FILES

/usr/cvfs/config/fsmcluster

SEE ALSO

cvadmin(8), **fsforeignservers**(4), **fsnameservers**(4), **fsmpm**(8), **mount_cvfs**(8)

NAME

fsmlist − StorNext File System FSM Auto-Start List

SYNOPSIS

/usr/cvfs/config/fsmlist

DESCRIPTION

The *StorNext File System* (*SNFS*) **fsmlist** file defines for the **fsmpm**(8) daemon the *File System Manager* (*FSM*) daemons to start. When the file does not exist, the **fsmpm** will not start any *FSM* daemons.

SYNTAX

The format for the **fsmlist** is simple. On each line is the name of one local file system to start, and an optional priority number from zero (0) to nine (9). The name must not include cluster-naming information.

The optional priority number is used when there is a redundant metadata controller (MDC). A priority of zero makes the specified *FSM* top priority and any number greater than zero means lower priority. See **cvfs_failover**(8) for details about setting up a failover-capable file system service.

The format of an **fsmlist** line has the following format:

File_System_Name [*priority*]

File_System_Name is the public name of the file system used in the **mount**(8) command, and as the prefix for the configuration file (see **snfs_config**(5)).

The *priority* field is used to designate a priority when there are redundant **fsm** daemons for a file system. Only one may be active at a time and the **fsmpm** daemon executes failover votes to determine the daemon to activate. The priority value helps the **fsmpm** determine, all other things being equal, which service to activate.

The previous version of **fsmlist**(4) was documented as requiring a dot (.) character as an argument before the *priority* argument, for compatibility with even older versions of **fsmlist**(4). In reality, it was optional. The first argument will be ignored if it is just the dot (.) character.

Lines that contain white space only or that contain the comment token as the first non-white space character are ignored.

FILES

/usr/cvfs/config/fsmlist

SEE ALSO

cvfs(8), **cvfs_failover**(8), **fsm**(8), **fsmpm**(8), **fsnameservers**(4), **mount**(8), **snfs_config**(5)

NAME

fsmpm − StorNext File System Port Mapper daemon

SYNOPSIS

fsmpm [**-nc**] [*host-ip* [*debug* [*sync* [*diskscan* [*extHA*]]]]]

DESCRIPTION

The *FSM Portmapper* is a server daemon residing on each StorNext File System client and server. It registers an RPC identifier to the system's *portmap* daemon. The **fsmpm** publishes a well known port where the SNFS File System Manager (FSM) daemons can register their file system name and port access number. All clients then talk to their local FSS port mapper to discover access information for their associated service. This process runs in the background and is started at boot time. It is enabled or disabled (along with the file system) via **chkconfig**(8) or **init.d** using the **cvfs** key word.

OPTIONS

Quantum Internal Use Only - contact support before adding or modifying command line arguments to **fsmpm**. Changes from the defaults may result in intermittent or total failure of StorNext. Options may change abruptly between releases.

-n Don't send a SIGQUIT signal to fsm processes that appear to be hung; normally if an fsm process doesn't exit within 60 seconds after breaking the connection to the fsmpm, the fsmpm sends a SIGQUIT signal to the fsm.

-c Assume the client kernel module has not been loaded and skip any operations related to it.

host-ip The IP address used to access this host. The default is to try to resolve the system's hostname to an address. If set to 127.1, no nameserver will be used and no heartbeats will be sent. (default: no hostname)

debug Bitmask specifying which debug messages to print. (default: 0)

sync The name of a file whose creation is used to detect when the fsmpm has successfully started. (default: no file)

diskscan
How often (in seconds) to rescan for changed paths. 0 disables the disk scan. (default: 0)

extHA External HA mode prevents the fsmpm from automatically starting FSMs in the fsmlist, as well as disabling calls for elections. 1 enables external HA mode; 0 disables it. (default: 0)

NOTE: Defaults specified with '--'.

FSNAMESERVERS

The **fsmpm** reads the file */usr/cvfs/config/fsnameservers* to establish file system name servers for the StorNext file system services. This list is used to coordinate the whereabouts of StorNext file system servers.

A name server list must be established on each client that has StorNext installed so that all clients can discover the location of the StorNext FSM servers. It is important that this list is consistent across the SAN. Inconsistent *fsnameservers* configuration may result in the inability for some clients to find a file system service. See **fsnameservers**(4) for specifics of the file format.

FSMLIST

The **fsmpm** is responsible for launching the FSM daemon(s). If */usr/cvfs/config/fsmlist* exists then the **fsmpm** reads from the list file and starts the FSM daemons that are specified. If no */usr/cvfs/config/fsmlist* file exists then the **fsmpm** tries to launch the FSM daemon for the file system named **default**. See **fsmlist**(4) for specifics of the file format.

Unique Identifier (UUID)

The **fsmpm** creates or accesses the */usr/cvfs/config/snfs_uuid* file. This contains a StorNext unique identifier (UUID) for this host. The file should not initially exist on a host that deploys StorNext. If configuration information is copied from one host to another, this file should either not be copied or be deleted on the new host.

ENVIRONMENT

These variables are for use by Quantum support personnel only.

FSMPM_MAX_LOGFILES
: Controls the number of old **nssdbg.out** log files that will be saved by **fsmpm** when the current log file reaches its maximum size. The default is 4.

FSMPM_MAX_LOGSIZE
: Controls the maximum size that the **nssdbg.out** file can grow to before a new log file will be started. The value is a number followed by an optional suffix (**K** for Kilobytes, **M** for Megabytes, **G** for Gigabytes), or the string **unlimited**, indicating that the file can grow without bound. The default is 1M and the minimum is 64K.

FSMPM_ACTIVE_FSM_CLIENT_TIMEOUT
: Adjusts the logic that determines when an FSM should be considered lost and an election started to determine a new active FSM. The value is specified in seconds.

WATCHER_NODETACH
: If set, **fsmpm** will not detach from its parent process. Used start **fsmpm** under the control of a debugger. Does not apply to Windows platforms.

WATCHER_NORESTART
: If set, **fsmpm** will not be restarted automatically after an unsuccessful exit or crash. Does not apply to Windows platforms.

DEBUG

Debugging traces are written to the file */usr/cvfs/debug/nssdbg.out*. The amount of debug information is controlled by the */usr/cvfs/debug/verbose* file. This file contains the list of debug traces to turn on. If the file does not exist, none of the optional debug traces are enabled. Blank lines and comments that begin with a **#** are ignored. Everything else is treated as a name for what debug traces to turn on. Names are separated by whitespace or commas, and may be listed on multiple lines. Unknown names are silently ignored.

general
: Print general trace information. This include information about acquiring port numbers for coordinators, listing FSMs, mapping IDs hostnames, disk requests, portmap inquiries, device event handlers, and other events.

input Print a trace for every NSS packet received.

output Print a trace for every NSS packet sent.

mbr Print traces about Heartbeat Membership changes.

vote Print traces about FSM elections.

ldap_cred
: Print traces for LDAP credential processing.

proxy Print traces about proxy processing (Distributed LAN client).

cctl Print traces about the cluster-wide central control (nss_cctl).

hamon_reset
: Print traces for HAMON resets.

helper Print traces about starting and stopping helper processes.

ahb Print traces about NSS accelerated heartbeats.

protobuf
: Print details of all NSS2 packets as they are created and parsed.

all Enable all debug tracing

Numeric values are also recognized as specific debug bits to set. This is mainly for backwards compatibility. A value of **-1** or **0xffffffff** is the same as specifying **all**.

If there are no recognized names or numeric values in the *verbose* file, e.g. if it exists but is empty, then the **mbr** and **vote** traces will be enabled.

An example of a *verbose* file:

```
# Comments are ignored
vote, mbr # Turn on the vote and membership traces
   general
foo bar # Unknown items are silently ignored

# These are very noisy
#input
#output

0x1000 # turn on an undefined bit
```

FILES

/etc/init.d/cvfs
/usr/cvfs/config/fsnameservers
/usr/cvfs/config/fsforeignservers
/usr/cvfs/config/fsmlist
/usr/cvfs/config/nss_cctl.xml*
/usr/cvfs/debug/nssdbg.out
/usr/cvfs/debug/snfs_uuid
/usr/cvfs/debug/verbose

NOTES

When a **SIGHUP** signal is received, the configuration files will be re-read. This allows, for example, the FSM list to be modified, or debugging levels to be changed.

SEE ALSO

chkconfig(8), **cvfs**(8), **fsm**(8), **fsnameservers**(4), **fsmlist**(4), **nss_cctl**(4), **portmap**(8), **rpcinfo**(8)

NAME

fsnameservers − StorNext File System Name Server List

SYNOPSIS

/usr/cvfs/config/fsnameservers

DESCRIPTION

The *StorNext File System* (*SNFS*) *fsnameservers* file describes to the **fsmpm**(8) daemon which machines are serving as *File System Name Server* coordinator(s) for the StorNext cluster. The file system name coordinators are a critical component of the StorNext File System Services (*FSS*). One of the principal functions of the coordinator is to manage failover voting in a high availability configuration. Therefore it is crucial to choose highly reliable systems as the coordinators. Redundancy is provided by listing multiple entries in the *fsnameservers* file, one entry per line. When StorNext services are started on a host, the **fsmpm** daemon sends a heartbeat to each coordinator in the list. The first response received determines the primary coordinator for that client in a given cluster. The remaining coordinators are backup coordinators and will be used if the primary is lost. It is recommended to configure at least two systems per cluster to utilize this redundancy benefit. The systems chosen may also be configured for the File System Manager (*FSM*) services but this is not required.

The *fsnameservers* files on the coordinator hosts should include the host names or IP addresses of all of the coordinators including its own. The *fsnameservers* files on the client and MDC hosts normally contain all these addresses as well, but these hosts will function normally with the address of at least one active coordinator.

If you are configuring multiple clusters, it is best practices to include the *cluster* and the *administrative_domain* on the same line as the IP address of each coordinator (see below).

See the **fsmcluster**(4) man page for a discussion on StorNext clusters and administrative domains.

If *fsnameservers* does not exist then the system will operate as a "local" filesystem representing both client and server. It will not communicate with any other *SNFS* product on the network unless an **fsforeignservers**(4) file is configured. A local filesystem may be desired when no SAN sharing is required.

The *fsnameservers* file is also used to specify the preferred metadata network(s) used for SNFS. That is, if an address in the *fsnameservers* file is on IP network *x*, then network *x* will be used to carry SNFS metadata traffic, if possible. Networks available over other interfaces will be used if there is no connectivity from a client to the FSM on a preferred network. A network other than preferred will also be selected if this network is directly connected (on the same subnet) and none of the preferred networks are directly connected. Multiple, redundant metadata networks can be created by using additional network interfaces on every system; each coordinator is then specified in *fsnameservers* multiple times, once for each of its metadata-network addresses.

It is possible to filter the networks or IP addresses that are advertised to clients using the *snfs_metadata_network_filter.json* configuration file. See the **snfs_metadata_network_filter.json**(5) man page for details. This capability is only offerred on Linux systems.

Proxy coordinators can be configured by specifying a masklength along with the IP address. A proxy coordinator acts as a coordinator for other clients on the same subnet, relaying information between the clients and the real coordinators. For a group of clients on the same subnet, this will keep the underlying heartbeat traffic localized to the subnet, and only the proxies will communicate with the real coordinators. When configured, clients on the same subnet as the proxy will prefer to use the proxy coordinator for their primary coordinator. If the proxy coordinators are non-responsive, clients will fall back to using the normal coordinators.

UPDATING fsnameservers

NOTE The following applies to StorNext coordinators, clients and MDCs that have been updated to a level that supports this procedure. Verify that this text is present in this man page on these nodes before proceeding.

If a set of coordinators for a cluster is being replaced, the old coordinators should remain active through the transition. The first step is to synchronize the new and old coordinators. Populate the *fsnameservers* file on

all coordinators with the IP addresses of all the coordinators. Then start or restart services on all coordinators. Once this is done, all clients and MDCs will dynamically learn the location of the new coordinators through the NSS protocol. This can be verified with the **coord** subcommand of the **cvadmin** utility. You should see entries for the old and new coordinators. Next, update all of the clients and MDCs *fsnameservers* files to contain only the addresses of just the new coordinators. There is no urgency to update the clients and MDCs as long as at least one of the old coordinators remains active. When all the client and MDC *fsnameservers* files have been updated, the old coordinators can be taken off line. The clients and MDCs will now switch over to the new coordinators for primary and secondary. The *fsnameservers* file on the new coordinators should now be updated to contain only the new coordinator names or addresses. There is no need to restart services on any of the clients or MDCs.

The process is similar if only a subset of the coordinators is being replaced. If, for example, a coordinator node failed, a new coordinator node can be brought on line with the new coordinator configuration in its *fsnameservers* file. Next, the existing coordinators should have their *fsnameservers* files updated. Start or restart services on all coordinators. Then, as above, update all the client and MDC *fsnameservers* files.

SYNTAX

The format for the *fsnameservers* is simple. It contains the IP address or the hostname to use as either a primary or a secondary coordinator. The use of IP addresses is preferred to avoid problems associated with lookup system (eg., DNS or NIS) failures. The format of an *fsnameservers* line is:

host[/*masklength*][ ][**@**[*cluster*[/*admin_domain*]]]

The *host* is a host name or IP address of a host that can coordinate queries and failover votes for the File System Services. The optional *cluster* and *admin_domain* fields specify the name of the cluster and administrative domain to which the coordinator belongs. To specify that *host* is a coordinator for more than one cluster, there must be a line for each cluster.

If only *host* is specified, the system will attempt to identify the cluster based on responses from the *host*. If it responds as an NSS1.X coordinator, it will be added to the default cluster, **@_cluster0/_addom0**. If it responds as an NSS2 coordinator, the coordinator will give the client its list of clusters and all the coordinators for each of those clusters. If you are running a multi-cluster configuration, it is highly recommended that you specify the cluster for each coordinator in the *fsnameservers* file.

If *admin_domain* is not specified, it will use the default name from **fsmcluster**(4).

If *cluster* is not specified, but the **@** is present, i. e.,

*host***@**
host **@**

it will use the default name from **fsmcluster**(4).

The optional /*masklength* configures *host* as a proxy coordinator for the specified subnet.

Lines that contain white space only or that contain the comment token as the first non-white space character are ignored.

FILES

/usr/cvfs/config/fsnameservers

SEE ALSO

cvfs(8), **cvadmin**(8), **snfs_config**(5), **snfs_metadata_network_filter.json**(5), **fsforeignservers**(4), **fsm**(8), **fsmpm**(8), **fsmcluster**(4), **mount_cvfs**(8)

NAME

fsports − StorNext File System Port Restrictions

SYNOPSIS

/usr/cvfs/config/fsports

DESCRIPTION

The StorNext File System (SNFS) **fsports** file provides a way to constrain the TCP and UDP ports used by core SNFS server processes consisting of the FSMs, the FSMPM, and any Distributed LAN servers. This file can also be used to redefine the port used by the SNFS Alternate Portmapper service. The **fsports** file is usually only necessary when the SNFS control-network configuration must pass through a firewall. Use of the **fsports** file permits firewall pinholing for improved security. If no **fsports** file is used then port assignment is operating-system dependent.

SYNTAX

When an **fsports** file exists in the SNFS 'config' directory, it restricts the TCP and UDP port bindings to the user-specified range. The format of the **fsports** file has two required lines, one optional line, and comments starting with pound-sign (#) in column one as follows:

```
MinPort value
MaxPort value
AltPmap value
```

The *value* fields are port numbers that define an inclusive range of ports that the SNFS server processes can use. Be careful to choose port range values that are appropriate for your operating system. Note that the Internet Assigned Numbers Authority (IANA) suggests a dynamic client port range for outgoing connections of 49152 through 65535. With that in mind, most Linux kernels use a port range of 32768 through 61000, Microsoft Windows operating systems through XP use the range 1025 to 5000 by default, Windows Vista, Windows 7, and Server 2008 use the IANA range by default, and Windows Server 2003 uses the range 1025 to 5000 by default, until Microsoft security update MS08-037 from 2008 is installed, after which it uses the IANA range by default.

The optional **AltPmap** value changes the TCP port used for the SNFS Alternate Portmapper service, which defaults to port 5164. When SNFS is used with a firewall, you can either configure the firewall to allow port 5164 or change the SNFS port number via the **AltPmap** keyword to a port that is allowed to pass through the firewall.

The **AltPmap** value must be defined the same way on all SNFS server and client nodes in order for SNFS to function properly. When using **AltPmap** to change the SNFS Alternate Portmapper service port number, you must include an fsports file on every SNFS client and server with the same **AltPmap** value. When the **AltPmap** keyword is not used, the **fsports** file is not necessary on SNFS clients.

The minimum number of ports needed on a given node can be computed as follows: one port for the FsmPm process, plus one port for each FSM process, plus one port for each file system served by this node as a Distributed LAN server.

EXAMPLE

To restrict SNFS Server processes to using ports 52,000 through 52,100 the **fsports** file would contain the following lines:

```
MinPort 52000
MaxPort 52100
```

This **fsports** file is only needed on SNFS servers.

To restrict SNFS Alternate Portmapper service to a specific port, select a port number outside the range of the **MinPort** and **MaxPort** values. For example if **MinPort** is 52000 and **MaxPort** is 52100 then select a value outside that range.

```
MinPort 52000
MaxPort 52100
AltPmap 52101
```

This **fsports** file is required on all SNFS clients and servers.

COMMON FSPORTS COOKBOOK

The primary use case for fsports is allowing StorNext SAN and LAN clients outside of a firewall to access the services provided by StorNext servers inside of a firewall. While other use cases exist, and more complex and restrictive configurations are possible, the following "cookbook" steps describe how to produce a common fsports file for all servers inside of a firewall. The resultant file provides security without sacrificing ease of deployment.

Step 1.
Determine the set of servers inside of the firewall that need to be accessed. This may include primary and standby MDCs and Distributed LAN servers.

Step 2.
For the list of servers from Step 1, determine the maximum number of StorNext ports used by any one server. The following equation can be used for this calculation:

ports required for a given server = (
Three ports for each file system in the "fsmlist" file +
One port for each file system for which this server is acting as a Distributed LAN server +
Two ports for the Tiering, if enabled +
Three ports for the Alternate Portmapper, the NSS protocol, and fsmpm web services) * *Multiplier*

where *Multiplier* is 2 for Windows servers and 1 for non-Windows servers.

For example, consider the following configuration of servers inside of a firewall that need to be accessed by clients outside of the firewall:

A Linux MDC hosting 4 file systems and Tiering enabled
ports: (4 * 3 + 0 + 2 + 3) * 1 = 17

A second Linux MDC hosting 6 file systems
ports: (6 * 3 + 0 + 0 + 3) * 1 = 21

A Windows MDC hosting 1 file system and acting as a Distributed LAN server for just that file system
ports: (1 * 3 + 1 + 0 + 3) * 2 = 14

A dedicated Linux Distributed LAN server that serves 9 file systems from the various MDC pairs
ports: (0 + 9 + 0 + 3) * 1 = 12

The maximum number of StorNext ports needed by any one server is:
MAX(17, 21, 14, 12) = 21 ports

Step 3.
Based on the maximum port count determined in step 2, pick a range of unused ports on the firewall and open them up. For example, for a maximum port count of 21, ports 52000 through 52020 could be chosen if they are available. The firewall should be configured to allow outside UDP and TCP connections using any source port to connect to StorNext servers inside the firewall having the restricted range of ports. Port 5164 should also be opened up for TCP for the AltPortMap service. (See NOTES below if this is not possible.)

Step 4.
Configure the common fsports file. Using the example from step 3 of where 10 ports are needed starting at 52000. The configuration is as follows:

```
MinPort 52000
MaxPort 52020
```

Step 5.
Install the resulting file on all servers from Step 1. Also install the file on all clients if the **AltPmap** directive was used. Then restart StorNext.

NOTES

Servers having common fsports files will use the same range of network ports for core StorNext file system services. This does not result in conflicts since each network address is comprised of an IP address and a port number and is therefore unique even when using the same port number as another network address.

As mentioned under EXAMPLES, when port 5164 cannot be opened on the firewall for the AltPortMap service, it is possible to use the **AltPmap** directive in the fsports file so that a different port is used. This also requires that clients outside of the firewall use an fsports file.

The fsports file does not constrain the ports used by the client end of connections. Ephemeral ports are used instead. Therefore, the fsports file is only useful on clients when the **AltPmap** directive is used.

When using fsports files, if services fail to start or clients fail to connect, to debug the problem, try slightly increasing the range of open StorNext ports on the firewall and, correspondingly, in the fsports files. Running netstat on the servers may reveal that unexpected processes are binding to ports within the range specified in the fsports file. Also, if services are restarted on Windows servers, in some cases ports may not be reusable for several minutes. Using an expanded port range will work around this.

The information above covers the ports used by the core StorNext file system services. If firewall pinholing is needed for other StorNext services (for example, replication), additional hard-wired ports may need to be opened on the firewall outside of the domain of the fsports file. Refer to the "Port Used by StorNext" section in the StorNext File System Tuning Guide.

UPDATING FSPORTS CONFIGURATION ON AN HA PAIR

When changing the fsports configuration on an HA pair and the **AltPmap** directive is used, special care must be taken to avoid having the HA mechanism reboot one or both of the HA nodes. Here is the sequence to follow:

Step 1.
Use the StorNext GUI to enter HA config mode. This locks the secondary node and allows configuration changes to be made on the primary node.

Step 2.
Stop the cvfs service on the primary node.

Step 3.
Stop the cvfs service on the coordinator (name service) nodes.

Step 4.
Update, remove or create the fsports file on the coordinator nodes.

Step 5.
Start the cvfs service on the coordinator nodes.

Step 6.
Update, remove or create the fsports file on both the HA primary and HA secondary nodes.

Step 7.
Start the cvfs service on the primary node.

Step 8.
Use the StorNext GUI to exit HA config mode.

If not using external coordinators, that is, the HA MDC pair are the coordinators for the cluster, eliminate steps 3, 4 and 5.

FILES

/usr/cvfs/config/fsports
/usr/cvfs/examples/fsports.example

SEE ALSO

cvfs(8), **snfs_config**(5), **fsm**(8), **fsmpm**(8)

NAME

fs_scsi − Send SCSI commands to a device.

SYNOPSIS

fs_scsi -h

fs_scsi [**-ad**] [**-sxirtecSR**] *device*

fs_scsi [**-ad**] [**-s**] **-l**|**-m** *page device*

fs_scsi [**-ad**] **-p**

fs_scsi [**-ad**] **-f** *serial_number*

fs_scsi [**-ad**] [**-O** *file*] [**-P** *file*]

fs_scsi [**-ad**] **-R**

DESCRIPTION

This utility can be used to get device information using SCSI commands. Some commands have corresponding command line options. Others must be invoked from the menu mode. This mode is entered if *device* and the -p option are not specified.

fs_scsi(1) can also be used to verify drive functionality.

OPTIONS

-h Help. Shows usage.

-a Enable support for all known device types instead of just supported types.

-d Debug enables logging in /tmp/logs. A trace log of all SCSI commands is generated in /tmp/logs/trace/trace_02.

-s Treats *device* as standard tape device path and attempts to convert it to the corresponding SCSI device path.

-x Translates specified *device* from a standard tape device path to the corresponding SCSI device path.

-i Displays the results of the SCSI inquiry command sent to the specified device.

-r Displays the results of the SCSI request sense command sent to the specified device.

-c Issues the appropriate SCSI commands to determine the capacity of the media currently mounted in the specified device. If no media is mounted, the capacity will be reported as 0.

-t This generates a summary of some device information. The output contains the following fields. Scsi Path, Drive Type, Product Id, Device type, Scsi Id, Serial Number.

-S This will report status of the drive by indicating if it is ready or not ready. Multiple SCSI test unit ready commands will be executed until the drive becomes ready or an error is encountered.

-e The will eject (unload) a media from the specified device if a media is loaded.

-l Displays the page results of the SCSI log sense command sent to the specified device. The page value should be entered in hex.

-m Displays the page results of the SCSI mode sense command sent to the specified device. The page value should be entered in hex.

-p This will probe scsi devices sending an inquiry command to each device and then display its Serial Number, Product Id, Device Type, and Device Path.

-R Query the state of persistent reservations and registrations. Lists all the reservation keys registered and information about the current holder of the reservation on the device. If there is no current reservation this will be noted.

-f *serial number*
This will attempt to find the device path associated with the the specified serial number.

-O *file* This will log performance statistics to the specified file for read and write operations performed from menu mode.

-P *file* The pattern file to use when writing. NOTE: The size of the pattern file will be used as the block size for I/O so care should be taken when creating and using a pattern file. Ideally the size of the file would be a multiple of 1024 bytes.

device The device path to operate on.

WARNINGS

This command is installed and available on the MDC as well as client machines where DDM is to be run. For this reason the command includes libraries that are required for access to the database on the MDC. The database however is not accessible from the client machines so some run-time warnings occur when the command is issued on those clients. For example:

... Cannot find installation location for MySQL ...
... Parameter FS_HOME not defined ...

These warnings are not fatal and the output for reporting options (like: **-p**) should be considered correct.

EXIT STATUS

This will exit with 0 upon successful execution. Anything else indicates failure.

NAME

has_snfs_label − Check for StorNext Disk Label

SYNOPSIS

has_snfs_label *device_path*

DESCRIPTION

The StorNext File System (SNFS) **has_snfs_label** utility used to test a device for a SNFS formatted label.

EXIT VALUE

```
0 - Device does not have a SNFS label
1 - Device has a SNFS label
```

SEE ALSO

cvlabel(8)

NAME

ha_peer − StorNext HA Peer Server IP Address

SYNOPSIS

/usr/cvfs/config/ha_peer

DESCRIPTION

The StorNext File System (SNFS) */usr/cvfs/config/ha_peer* file provides the IP address of the peer server in a StorNext High Availability (HA) cluster configuration to the *snhamgr_daemon* and *fsmpm* processes. It must be configured on both servers. The **ha_peer** file may also be used to change the default IP port (5189) used by the *snhamgr_daemon* and *snhamgr* processes.

The **ha_peer** IP address allows the fsmpm processes to negotiate the restarting of HA Timers to avoid unnecessary HA Reset incidents when metadata writes are delayed by heavy disk activity, and to detect misconfigured */usr/cvfs/config/ha_smith_interval* HA Timer override values. For HA clusters with an *HaShared* file system, the address also allows communication between the *snhamgr_daemon* processes running on the metadata controllers to collect operational status for the *snhamgr* command.

SYNTAX

Any entry that does not begin with # is assumed to be the peer IP address and optional port number. The address should be in IPv4 or IPv6 numerical format. The address and port entries are specified as follows:

```
<address>
```

or

```
<address>:<port>
```

For IPv6, enclose the address portion in square brackets if also specifying a port.

EXAMPLE

To set the peer address, the **ha_peer** file would contain the following line:

```
192.168.10.1
```

To set the peer address and port, the **ha_peer** file would contain the following line:

```
192.168.10.1:8888
```

To set an IPv6 address or address and port, the following lines could be specified:

```
fe80::250:56ff:fe9b:74b8
```

or

```
[fe80::250:56ff:fe9b:74b8]:4600
```

FILES

/usr/cvfs/config/ha_peer

SEE ALSO

snhamgr(8), **fsmpm**(8), *Quantum StorNext User's Guide*

NAME

mdt − Metadata performance test

SYNOPSIS

mdt [*options*] *path* [*path*...]

DESCRIPTION

mdt is a utility that can be used to measure the relative metadata performance of a given file system(s). By default, **mdt** measures creates/second and removes/second. Optionally **mdt** will also measure the per second rates for stat, chmod, and move (rename) operations.

Each path specified on the command line corresponds to a mount point for a file system to be tested. By default, **mdt** will start 8 pthreads. Each pthread will make a directory and create 16384 files in the directory. The number of pthreads (directories) can be specified via the -n option, and the number of file per directory can be specified via the -f option.

mdt is a metadata only test and no writes or reads are done. By default files are zero length. For a StorNext file system, space may be allocated to the file via the -b option which performs a StorNext API CvApi_VerifyAlloc call. Be aware that allocating space to files in this manner is not reflected in the file size.

On completion, **mdt** will print the number of files and rates for the specified metadata requests in the form of operations/second. The -v option will expand output listing per stream detail.

Unless the -R option is specified, **mdt** will remove all created files and directories in the remove test section.

For StorNext file systems, **mdt** will unmount and then mount the target file systems between test sections in order to clear the client metadata cache. Targeted file systems should not be busy as to prevent remounting. The -U option will cause **mdt** to bypass remounting between test sections.

OPTIONS

-? Display usage.

-a Run all test sections. In addition to create and remove, rates for stat, chmod, move (rename) are also measured.

-b *prealloc_size_in_bytes*

Specify the size in bytes to preallocate after creating a file. The default is 0 bytes and no preallocate.

Optionally, a single letter suffix can represent bytes in units. **mdt** single letter unit suffixes are as follows.

```
k     KB    10^3                1,000
m     MB    10^6            1,000,000
b     GB    10^9        1,000,000,000
K     KiB   2^10                1,024
M     MiB   2^20            1,048,576
B     GiB   2^30        1,073,741,824
```

-C Do not create files. This option requires that a create run was done at some previous time and the files were not removed. See the -R option.

-c Run the chmod test section.

-d[**dd**] Run in debug mode. The more "d's" specified, the more debug information is printed.

-E *csv_file*
Create and write results to the specified .csv file. A .csv file, (comma separated value) is a file that can be consumed by a spreadsheet application like Microsoft Excel.

-e *csv_file*
Write results to the specified .csv file; one that can be read by Microsoft Excel or other spreadsheet application. Data is appended to the specified csv file which must exist. See the -E option.

-f *number_of_files*
Specify the number of files per directory. Optionally a single letter suffix can be provided. See the -b option for suffix detail.

-F *fsname*
Specify the name of the file system to use on the remount. The file system name returned by the mount command varies by operating system. If you have trouble with remounts, you may have to specify the file system name.

-m Run the move/rename test section. Files are renamed in within their parent directories.

-n *number_of_directories*
The specified number of directories are created. Each will contain the number of files specified by the -f option. A pthread is created for each directory to execute the designated operations.

-R Skip the remove section and leave the files in place.

-s Run the stat test section. This section will execute readdir operations and stat each file in each directory; the equivalent of an ls -l command.

-T *tag_string*
Write the specified tag string to the standard output and to the csv file if the option to write a csv file is selected. This can be used to stamp results with a name identifying a given test scenario.

-U Do not unmount and remount the file system between test sections.

-v Print verbose output. The result for each stream is reported.

-V Print the **mdt** version information and exit.

EXAMPLES

Run the create and remove test sections on a single file system:

```
per1-# mdt /jhb10
mdt: Timing metadata ops/second on 8 streams of 16384 files on 1 file system(s)
    mount point        #files    creates  removes
    /jhb10             131072      18441    23792
```

Run the create and remove test sections on four file systems:

```
per1-# mdt /jhb10 /jhb11 /jhb12 /jhb13
mdt: Timing metadata ops/second on 8 streams of 16384 files on 4 file system(s)
    mount point        #files    creates  removes
    /jhb10             131072      13818    12134
    /jhb11             131072      13772    11821
    /jhb12             131072      13839    12117
    /jhb13             131072      13726    12098
mdt: Aggregate:        524288      54901    47283
```

Run all test sections on four file systems:

```
per1-# mdt -a /jhb10 /jhb11 /jhb12 /jhb13
mdt: Timing metadata ops/second on 8 streams of 16384 files on 4 file system(s)
    mount point        #files    creates    stats   chmods    moves  removes
    /jhb10             131072      14105    32904    16721     8366    12072
```

```
    /jhb11             131072      14135    32912    16616     8375    11969
    /jhb12             131072      14175    32883    16659     8436    11895
    /jhb13             131072      14003    33917    17092     8411    12131
mdt: Aggregate:        524288      56005   131510    66462    33465    47578
```

Run the create and remove test sections on four file systems. Create 32 directories using 32 threads with 8192 files in each.

```
per1-# mdt -f8K -n32 /jhb10 /jhb11 /jhb12 /jhb13
mdt: Timing metadata ops/second on 32 streams of 8192 files on 4 file system(s)
    mount point        #files    creates  removes
    /jhb10             262144      12149    10683
    /jhb11             262144      12292    10730
    /jhb12             262144      11859    10780
    /jhb13             262144      11782    10806
mdt: Aggregate:       1048576      47123    42732
```

NOTES

The number of files (-f) and directories (-n) are per file system.

The number of pthreads (streams) that are run in each file systems corresponds to the number of directories specified.

Other than #files, the output values represent metadata operations/second.

FILES

/usr/cvfs/bin/mdt

SEE ALSO

cvfs(8)

NAME

Mio − Multi-stream streaming I/O test

SYNOPSIS

Mio [*options*] *filename* [*filename*...]

DESCRIPTION

Mio is a utility that can be used to measure the I/O performance, in terms of bandwidth, of a system, I/O infrastructure, disk subsystem, disk device and/or file system. **Mio** uses pthreads to asynchronously queue requests to a stream or streams.

Each file name specified on the command line corresponds to a stream. The -q option defaults to 2 as described below, and specifies the number of asynchronous read or write requests that are queued to each stream. A file name can represent a regular file, or a block or character device (see NOTES). By default, **Mio** will issue read requests, with a -w option required for writes.

Upon completion, **Mio** will print a summary of I/O performance for each stream as well as the aggregate performance off all streams. Because the aggregate performance is the performance of all streams over the test run time, it may not reflect the sum of the individual streams.

OPTIONS

-? Display usage.

-b *buffer_size*

Specify the I/O buffer size to use. This value represents the size of each I/O request in bytes. The default buffer size value is 1048576 bytes. Optionally, a single letter suffix can represent bytes in units. **Mio** single letter unit suffixes are as follows.

k	KB	10^3	1,000
m	MB	10^6	1,000,000
b	GB	10^9	1,000,000,000
K	KiB	2^10	1,024
M	MiB	2^20	1,048,576
B	GiB	2^30	1,073,741,824

-B Do buffered I/O. By default, **Mio** will open files for direct I/O. See NOTES below for more information on direct I/O.

-c Create the file(s). This is useful and necessary for regular files, as they must be created before they can be read or written. The -c option is valid only when specified with the -w option.

-d[**dd**] Run in debug mode. The more "d's" specified, the more debug information is printed.

-e *csv_file*

Write results to the specified .csv file; one that can be read by Microsoft Excel or other spreadsheet application. Data is appended to the specified csv file which must exist. See the -E option.

-E *csv_file*

Create and write results to the specified .csv file. A .csv file, (comma separated value) is a file that can be consumed by a spreadsheet application like Microsoft Excel. The maximum number of streams that can be recorded in a .csv file is limited to 16 unless the -u option is specified.

-f Fsync on close. The default is no fsync.

-I *file_offset_increment*

Set the file offset increment for each I/O. For example, when set to zero, we repetitively do I/O to the same file location. The default is to use the buffer size. An **Mio** single letter unit suffix may be specified.

-n *nios* Specifies the number of I/O requests. The default is 1024.

-O *file_offset*
Start the I/O at the given file offset. An **Mio** single letter unit suffix may be specified. The default starting file offset for a regular file is zero. The default starting file offset for a block device is two times the I/O buffer size. This allows writing to a block device without destroying StorNext labels (writing to the block device will, however, destroy the file system).

-p Preallocate the file using the StorNext CvApi_VerifyAlloc API call. The file size will correspond to the number of I/Os times the buffer size. This option is supported only for regular files on a StorNext file system.

-q *queue_depth*
Queue the specified number of I/O requests on each stream. The default is 2. The maximum is currently 256.

-r *rtios* Set realtime mode by setting the number of I/O's per second to the specified value. Realtime mode is available only for regular files on the StorNext file system. An Mio single letter unit suffix may be specified.

-R Record I/O response times. **Mio** will record the I/O response time for each I/O request and increment a counter in the corresponding response time bucket. I/O response time buckets are then printed for each stream along with the transfer rate summary.

-s Preallocate the file using the specified stripe group only. If allocation can't be satisfied on the specified stripe group, the preallocation call fails. This option is only effective if used in conjunction with the -p option. This option is supported only for regular files on a StorNext file system.

-S *seconds*
Run the test for the maximum number of seconds specified. This only has an effect as a maximum, in that, if the test has not completed the specified number of requests after the specified time.

-t *total_data*
Terminate the test after the specified amount of data has been moved rather than a given number of requests. This option is not compatible with the -n option. An **Mio** single letter unit suffix may be specified.

-T *tag string*
Specify a tag string that will be written to the output. An additional line containing the specified tag string will be written in the test summary. If the -E option is specified, the tag string will also be written to the .csv file.

-u Specify that the unlimited streams format be used for csv files. Each stream's data is written to a new line in the file allowing for greater than 16 streams.

-w Specify that writes are done. A regular file must exist, or the -c option must be specified to create the file. The default is to read.

-W *write_mask*
The specified write mask will result in writes on streams where bits are set and reads on streams where bits are clear. The write mask may be specified in hexadecimal, octal, or decimal but is limited to 64 bits, hence 64 streams. The -W option is incompatible with the -c option and all files must exist.

EXAMPLES

Create and issue the default number of writes to four files/streams:

```
per1-# Mio -cw /jhb/f0 /jhb/f1 /jhb/f2 /jhb/f3
Mio: Timing 4 stream(s) of 1024 x 1M direct writes queued 2 deep
    stream[0]:          f0: write  1073.74 MBytes @   256.44 MBytes/Second
    stream[1]:          f1: write  1073.74 MBytes @   228.43 MBytes/Second
    stream[2]:          f2: write  1073.74 MBytes @   265.04 MBytes/Second
    stream[3]:          f3: write  1073.74 MBytes @   242.18 MBytes/Second
```

```
Mio: Aggregate:  4294.97 Mbytes @ 913.72 MBytes/Second
```

Issue the default number of reads to the four previously created files:

```
per1-# Mio /jhb/f0 /jhb/f1 /jhb/f2 /jhb/f3
Mio: Timing 4 stream(s) of 1024 x 1M direct reads queued 2 deep
    stream[0]:           f0: read   1073.74 MBytes @  1329.16 MBytes/Second
    stream[1]:           f1: read   1073.74 MBytes @  1334.83 MBytes/Second
    stream[2]:           f2: read   1073.74 MBytes @  1316.28 MBytes/Second
    stream[3]:           f3: read   1073.74 MBytes @  1329.92 MBytes/Second
Mio: Aggregate:  4294.97 Mbytes @ 5265.12 MBytes/Second
```

Issue the default number of reads to two block devices:

```
per1-# Mio /dev/sdd /dev/sdf
Mio: Timing 2 stream(s) of 1024 x 1M direct reads queued 2 deep
    stream[0]:          sdd: read   1073.74 MBytes @   408.38 MBytes/Second
    stream[1]:          sdf: read   1073.74 MBytes @   407.78 MBytes/Second
Mio: Aggregate:  2147.48 Mbytes @ 815.57 MBytes/Second
```

Create and do 10000 writes to four files via four streams:

```
per1-# Mio -cw -n10000 /jhb/f0 /jhb/f1 /jhb/f2 /jhb/f3
Mio: Timing 4 stream(s) of 10000 x 1M direct writes queued 2 deep
    stream[0]:           f0: write 10485.76 MBytes @   488.97 MBytes/Second
    stream[1]:           f1: write 10485.76 MBytes @   448.84 MBytes/Second
    stream[2]:           f2: write 10485.76 MBytes @   458.36 MBytes/Second
    stream[3]:           f3: write 10485.76 MBytes @   497.57 MBytes/Second
Mio: Aggregate:  41943.04 Mbytes @ 1795.35 MBytes/Second
```

Issue 10000 reads, queued 8 deep to four streams:

```
per1-# Mio -q8 -n10000 /jhb/f0 /jhb/f1 /jhb/f2 /jhb/f3
Mio: Timing 4 stream(s) of 10000 x 1M direct reads queued 8 deep
    stream[0]:           f0: read  10485.76 MBytes @   898.16 MBytes/Second
    stream[1]:           f1: read  10485.76 MBytes @  1075.04 MBytes/Second
    stream[2]:           f2: read  10485.76 MBytes @  1050.13 MBytes/Second
    stream[3]:           f3: read  10485.76 MBytes @   869.96 MBytes/Second
Mio: Aggregate:  41943.04 Mbytes @ 3479.83 MBytes/Second
```

Issue 240 x 12746752 reads to four streams and "tag" the summary:

```
per1-# Mio -b12746752 -n240 -T full_aperture_2K /jhb/f0 /jhb/f1 /jhb/f2 /jhb/f3
Mio: full_aperture_2K
Mio: Timing 4 stream(s) of 240 x 12448K direct reads queued 2 deep
    stream[0]:           f0: read   3059.22 MBytes @   884.50 MBytes/Second
    stream[1]:           f1: read   3059.22 MBytes @   726.70 MBytes/Second
    stream[2]:           f2: read   3059.22 MBytes @   808.67 MBytes/Second
    stream[3]:           f3: read   3059.22 MBytes @   656.74 MBytes/Second
Mio: Aggregate:  12236.88 Mbytes @ 2626.95 MBytes/Second
```

Issue 10000 reads to four streams and report the response times:

```
per1-# Mio -R -n10000 /jhb/f0 /jhb/f1 /jhb/f2 /jhb/f3
Mio: Timing 4 stream(s) of 10000 x 1M direct reads queued 2 deep
    stream[0]:           f0: read  10485.76 MBytes @   536.02 MBytes/Second
    stream[1]:           f1: read  10485.76 MBytes @   688.96 MBytes/Second
    stream[2]:           f2: read  10485.76 MBytes @   856.74 MBytes/Second
```

```
    stream[3]:           f3: read  10485.76 MBytes @   841.03 MBytes/Second
Mio: Aggregate:  41943.04 Mbytes @ 2144.08 MBytes/Second

I/O response time buckets:      0ms    5ms    10ms    20ms    50ms   100ms   500m
    stream[0]:           f0:     7632    1459     768     137       4       0
    stream[1]:           f1:     8570     928     395     103       4       0
    stream[2]:           f2:     9240     491     197      68       4       0
    stream[3]:           f3:     9196     494     234      76       0       0
```

NOTES

Writing to a device that contains data can result in data loss. Do not write to drives that contain any useful information.

Character devices - Some platforms have character device representations of disk devices and/or disk partitions that support read/write and some do not.

Direct I/O - Direct I/O is not supported on all platforms, although it is pretty well tested on Linux and Windows. Direct I/O support for specific files on other platforms is schizophrenic at best, and may not be supported on a particular file system, block device, character special device, or without appropriate mount options.

Windows - **Mio** on Windows can be used from either a cygwin environment or a DOS shell environment. File names are specified using the drive letter format. Use of a leading slash is recommended to avoid interpretation of the path based on the last relative location in that drive. A Windows physical drive is specified using the \\.\ notation. For example, PhysicalDrive9 is specified as \\.\PhysicalDrive9.

The Windows cygwin environment requires escaping the backslash resulting in 4 backslashes, period, 2 backslashes. Failure to escape the backslash cygwin can cause writing to the root directory as the name is changed to \.PhysicalDrive9.

FILES

/usr/cvfs/bin/Mio

SEE ALSO

cvfs(8)

NAME

mount_cvfs − mount a StorNext client file system

SYNOPSIS

Solaris:

mount [**-F cvfs**] [**-o** *options*] *filesystem dir*

Linux:

mount [**-fnv**] [**-t cvfs**] [**-o** <*options*>] <*filesystem*> <*dir*>

Direct Execution:

mount_cvfs *filesystem dir* **cvfs** *options*
mount_cvfs *server***:** *filesystem dir* **cvfs** *options*

Direct Execution (Linux)

mount_cvfs *control_device filesystem dir* **cvfs** *options*

DESCRIPTION

mount_cvfs is a *mount helper* utility that mounts a StorNext file system on client machines. On Linux and Solaris these utilities are called by the **mount**(8) utility to mount file systems of type **cvfs**. These helper utilities are designed to be invoked only by the **mount**(8) utility; if they are invoked directly on the command line, the option and argument location is strictly positional.

Each client file system must communicate with a File System Manager (FSM) running either locally or on a remote host. The FSM manages all the activity for the client in terms of storage allocation and metadata. Data transfers go directly between disks and the client. In the second form of the **mount_cvfs** command, the hostname of the FSM server is explicitly given in a syntax similar to NFS.

The FSM can manage a number of different StorNext file systems. Each different file system is specified in a configuration file on the FSM host. For example, a sample file system configuration is provided in the FSM configuration file */usr/cvfs/examples/example.cfgx*.

The **mount_cvfs** command supports mounting file systems that are running in a cluster other than your default cluster. Your default cluster is defined with the **fsmcluster**(4) file or, if this doesn't exist, the default is **_cluster0/_addom0**. When mounting a file system in a non-default cluster, the filesystem must be qualified with the correct cluster information. The syntax is **filesystem@<cluster>[/addom]**.

OPTIONS

Options supported by the mount command:

-f **LINUX ONLY**

Fakes the mount process but updates the */etc/mtab* file. The mount call will fail if the mtab entry already exists.

-n **LINUX ONLY**

Mounts the filesystem without updating the */etc/mtab* file.

-v Verbose mode.

Additional options may be specified in the */etc/fstab* file or on the **mount**(8) command line via the **-o** parameter. The **-o** parameter should be specified only once. If multiple options are needed, they should follow the **-o** in a comma-separated list.

ro Default: rw

Mount the file system read-only.

rw Default: rw

Mount the file system read/write.

noatime
Default: off

Do not update inode access times on this file system. Silently disabled on a managed filesystem.

nodiratime
Default: off

Do not update directory access times on this file system.

compat32
Default: off

When set, force directory offsets to fit into 31 bits and inode numbers to 32 bits. This should only be used when a problem has been identified with using the full size of the struct dirent d_off field from **readdir(2)** or older clients that are unable to handle large inode numbers.

nodev Default: off

Do not allow device special files to be accessed on the file system.

noexec Default: off

Do not allow the execution of programs resident on this file system.

nosuid Default: off

When executing programs resident on this file system, do not honor the set-user-ID and set-group-ID bits.

threads=*n*
Default: **12**

Determines the number of kernel threads that are created. On some platforms these threads will show up as **cvfsiod** processes in the output of **ps**.

This setting does not affect other kernel threads, for example, **cvfsd**, **cvfsbufiod**, **cvfsflusher**, **cvfs_dputter**.

The minimum value allowed is 12.

stripeclusters=*n*
Default: **8**

In certain cases, such as with using JBOD devices it may be possible to over-load their command queues using SNFS. If this occurs, the I/O concurrency can be reduced by reducing the number of concurrent *stripeclusters* the file system uses. The reduction is at the cost of performance.

buffers={**yes**|**no**}
Default: **yes**

When set to **yes**, the file system will use buffer caching for unaligned I/O.

protect_alloc={**yes**|**no**}
Default: **no**

When set to **yes**, non-root users are unable to use the preallocation ioctl. Note: protect_alloc=yes implies sparse=yes.

diskless={yes|no}
Default: **no**

If the diskless option is set to **yes** then the mount will succeed, even if the file system's disks are unavailable. Any subsequent I/O will fail until the file system's disks are visible through the SNFS portmapper.

diskproxy={client|server|both}
If the diskproxy option is set to **client**, then the mount may use a Proxy Server to do its data I/O. If the client host has SAN connectivity to some or all of the disks in the file system, then those disks will be accessed via the SAN connection, not the network. This client is then referred to as a disk proxy hybrid client. When SAN connectivity is used, the server license on the MDC will be charged for this mount. If it is desired that this client use the network for the mount, then the disks should be made unavailable to this host or the **cvpaths** file should be configured to prevent StorNext from using the directly attached disks. The **who** subcommand of **cvadmin** shows the type of proxy mount.

If the diskproxy option is set to **server**, then this system will become a Proxy Server for this file system. A **dpserver** configuration file must exist to define the operating parameters for the Server. See **dpserver**(4) and **sndpscfg**(8) for details.

A set of proxy servers may be configured in a sparse manner where each server sees only a subset of the disks in the file system. The servers make use of the "diskless" mount option. The proxy client will issue disk i/o requests to the appropriate server. No special configuration is needed on the client. A proxy server configured this way will not be able to access disks it does not see locally.

If the diskproxy option is set to **both**, then the host acts as both a client and a server. This overcomes the issues of the sparse proxy server configuration mentioned above. This option is only supported on Linux.

Note: The diskproxy option is available only on Linux and OS X systems, and the server option is available on Linux systems. The diskproxy selection on Windows clients is made through the Client Configuration utility.

proxypath={balance|rotate|sticky|filestickybalance|filestickyrotate}
Only used if the diskproxy option is set to **client**, controls the algorithm used to balance I/O across Proxy Server connections.

The proxy client keeps track of bytes of I/O pending, bytes of I/O completed and the elapsed time for each I/O request. It uses these values and certain rules to determine the server that is used for subsequent I/O requests. These collected counters are decayed over time so that only the most recent (minute or so) I/O requests are relevant.

There are two main components of the selection - the algorithm itself and the use of file sticky behavior. The algorithms are balance, rotate and sticky.

The **balance** algorithm attempts to keep the same amount of time's worth of I/O outstanding on each connection. i.e. Faster links will tend to get more of the I/O. A link could be faster because a given server is more efficient or less busy. It also may be the case that network traffic over a given link uses higher speed interconnects such as 10G ethernet.

The **rotate** algorithm attempts to keep the same number of bytes of I/O pending on each Proxy Server connection. This is similar to balance in that servers which respond more quickly to I/O requests will have the outstanding I/O bytes reduced at a more rapid pace than slower servers and will thus be used more often than slower links.

The difference between balance and rotate is that with balance, higher speed links will have more bytes of I/O outstanding than slower links.

In both balance and rotate, if more than one path has the best score, a pseudo-random selection among the winning paths is made to break the tie.

The **sticky** algorithm assigns I/O to specific luns to specific Proxy Server connections.

Filesticky behavior attempts to assign I/O for a given file to a specific proxy server. It does this by using the file's inode number modulo the number of servers to select a server index. Since all clients see the same inode number for a given file, all clients will select the same server. If there is more than one path to that server, then the algorithm (balance or rotate) will be used to select among the paths.

Filesticky behavior is controlled through a mount option.

When no proxypath mount option is specified, the balance option is selected.

For mount options balance and rotate, filesticky is not selected. For filestickybalance and filestickyrotate filesticky is selected.

Note: The proxypath mount option is available only on Linux, OS X, and Solaris systems. The proxypath options are selected on Windows clients through the Client Configuration utility.

proxyclient_max_conns=*n*

Only used if the diskproxy option is set to **client**. This parameter is used to limit the number of LAN client connections made per gateway. It overrides the **server_conn_count** parameter in the dpserver(4) file. However, it can only be used to decrease the number of connections. If the value of **proxyclient_max_conns** is equal to or larger than **server_conn_count**, it has no effect.

The minimum value is 1 and the maximum is 16 which is also the default.

proxyclient_rto=*n*

Only used if the diskproxy option is set to **client**. Defines the starting value in seconds to wait for a Proxy Client I/O read request to complete before disconnecting from the Proxy Server and resubmitting the request to a different Proxy Server. If reads are completing but coming close to the configured timeout, the timeout will be increased.

The minimum value is 1 second, maximum is 3600 and the default value is 15.

Note: This option is available only on Linux, OS X, and Solaris systems.

proxyclient_wto=*n*

Only used if the diskproxy option is set to **client**. Defines the starting value in seconds to wait for a Proxy Client I/O write request to complete before disconnecting from the Proxy Server and resubmitting the request to a different Proxy Server. If writes are completing but coming close to the configured timeout, the timeout will be increased.

The minimum value is 1 second, maximum is 3600 and the default value is 30.

Note: This option is available only on Linux and Solaris systems.

proxyclient_wrq=*n*

Only used if the diskproxy option is set to **client**. Defines the number of seconds to wait for lost write requests. A lost write request is an active write through a gateway and the connection to that gateway is unexpectedly lost. These writes may or may not have been flushed to disk or even started at the time the client notices the connection is lost. The default behavior (0) is that lost

writes are immediately re-queued to an available gateway. If the connection to the gateway over which the lost writes were sent is reactivated, the gateway will be queried if any writes from this connection are still active. If there are none, such as would be the case if the server unexpectedly re-booted, the client will immediately requeue all lost writes from the previous connection to this gateway. A value of -1 indicates that the client will never time out lost writes.

The minimum value is -1, maximum is 2147483647 and the default is 0.

Note: This option is available only on Linux systems.

atimedelay={**yes**|**no**}
Default: **no**

Perform lazy atime updates. This option improves performance by waiting until closing a file before updating the atime value of the file. This reduces extra network traffic and latency linked to atime updates.

nrtiotokenhold=*n*
Default: **60**

The QOS Token Hold Time (nrtiotokenhold) parameter is only applicable when using the SNFS Quality of Service (QOS) feature for real-time I/O. The parameter determines the number of seconds that a client stripe group will hold on to a non-realtime I/O token during periods of inactivity. If no I/O is performed on a stripe group within the specified number of seconds, the token will be released back to the FSM.

The parameter should be specified in five second increments; if the parameter is not a multiple of five, it will be rounded up automatically.

auto_concwrite={**yes**|**no**}
Default: **no**

When set to **yes**, allows multiple threads to write to files concurrently.

Note: setting auto_concwrite=yes requires that sparse=no also be specified. Also, protect_alloc=yes is disallowed with auto_concwrite=yes.

cachebufsize=*size*
Default (in bytes): **256K** if **buffercachecap** is <= 1gb else **1024K**

This option sets the size of each cache buffer. This determines the I/O transfer size used for the buffer cache. For optimal performance, cachebufsize should match the RAID stripe size. If cachebufsize is less than the RAID stripe size write performance may be severely degraded.

The maximum value allowed is 16384k. The minimum value allowed is 1. The value is rounded up to be a multiple of file system blocks. For example, if the file system block size is 4k and a value of 1 is used, the cachebufsize will be 4k and a value of 2047k would be rounded up to 2048k.

Can be specified in bytes (e.g. 131072) or kilobytes (e.g. 128k).

buffercachecap=*n*
Depicted in megabytes (MB).

Default: varies by the amount of detected physical memory:

with < 2GB of memory => **256MB**
from 2GB to 4GB - 1 of memory => **512MB**
from 4GB to 8GB - 1 of memory => **1024MB**
with >= 8GB of memory => **2048MB**

Tell the system how much memory in MB units to use for the **cachebufsize** associated with this mount. All mounted file systems with the same **cachebufsize** share this amount of memory unless the mount option **privatebufcache** is specified. It is generally advantageous to have file system mounts share caches to reduce memory consumption. However, there are cases where having a private cache may improve stability and performance since cache contention is reduced. For example, using a private cache for the HA Shared file system is recommended.

If a subsequent mount with the same **cachebufsize** increases **buffercachecap**, an attempt is made to allocate more buffers. If **buffercachecap** is less than or equal to a previously mounted file system already mounted with the same **cachebufsize**, the value is ignored. If the number of buffers already allocated for this **cachebufsize** is less than **buffercachecap**, an attempt is made to allocate more buffers. If any allocation fails, mount stops trying to allocate and the mount succeeds unless not even 10 buffers could be allocated. In this last case, mount fails with ENOMEM.

If the total amount of memory on the system is 4GB or less, the value of **buffercachecap** must be between 1 and 1/2 the memory size (in MB). For example, if the machine has 2GB of memory, **buffercachecap** can be a value from 1 up to and including 1024.

If the total amount of memory on the system is greater than 4GB, the maximum value for **buffercachecap** is given by:

MINIMUM(memory_size_in_MB - 2048, .9 * memory_size_in_MB)

Note that some operating systems reserve a percentage of memory for special purposes making the available memory somewhat smaller than the physical capacity of the installed RAM.

Also note that while a maximum value exists for **buffercachecap** that attempts to prevent having a single mount consume too much memory, no checks are made across other caches or other memory consumers including user processes. For example, it is possible to oversubscribe memory by configuring different values of **cachebufsize** across mounts and specifying large values of **buffercachecap**. Oversubscription is also possible when specifying very large values of **dircachesize** and **buffercachecap** for a particular mount.

The **cvdb**(8) command can be used with the **-b** option to see how many buffers have been allocated for each **cachebufsize**.

bufferlowdirty=*n*
bufferhighdirty=*n*

These options set the low and high watermarks for background buffer flushing and values are depicted in megabytes (MB). Background buffer flushing is initiated at bufferhighdirty and continues until bufferlowdirty is reached. Note: these options are not intended for general use. Only use when recommended by Quantum Support.

buffercachemin=*n*

See **buffercachecap**. This option has been deprecated and is ignored. Depicted in megabytes (MB).

buffercachemax=*n*

See **buffercachecap**. This option has been deprecated and is ignored. Depicted in megabytes (MB).

verbose={**yes**|**no**}

Default: **no**

When set to **yes**, **mount_cvfs** will display configuration information about the file system being mounted.

debug={**yes**|**no**}

Default: **no**

When set to **yes**, **mount_cvfs** will display debugging information. This can be useful in diagnosing configuration or disk problems.

mnt_retrans=*n*
Default: **1**

Indicates the number of retransmission attempts the file system will make during the execution of the **mount**(2) system call. Until the file system is mounted, the kernel will only retransmit messages to the FSM **mnt_retrans** times. This parameter works in conjunction with the **mnt_recon** parameter. This can help reduce the amount of time a mount command will hang during boot; see the **mnt_type** option.

mnt_recon={**hard**|**soft**}
Default: **soft**

Controls whether after *mnt_retrans* attempts at contacting the FSS during the mounting and unmounting of a file system, the kernel will either give up or continue retrying forever. It is advisable to leave this option at **soft** so that an unresponsive FSS does not hang the client during boot.

mnt_type={**bg**|**fg**}
Default: **fg** (foreground)

Setting **mnt_type** to **bg** will cause the mount to run in the background if the mount of the indicated file system fails. **mount_cvfs** will retry the mount **mnt_retry** number of times before giving up. Without this option, an unresponsive FSM could cause a machine to hang during boot while attempting to mount StorNext file systems.

During background mounts, all output is re-directed to **/var/adm/SYSLOG**.

mnt_retry=*n*
Default: Depends on **mnt_type**. If **mnt_type**==fg, **mnt_retry**=5 otherwise **mnt_retry**=500.

If a mount attempt fails, retry the connection up to *n* times.

retrans=*n*
Default: **5**

Indicates the number of attempts that the kernel will make to transmit a message to the FSM. If no response to a transmitted message arrives in the amount of time indicated by the **timeout** parameter, the request will be retransmitted. If the file system was mounted with the **recon=soft** parameter, the file system will give up after *retrans* attempts at sending the message to the FSM and will return an error to the user.

recon={**hard**|**soft**}
Default: **hard**

This option controls whether after *retrans* attempts at sending a message to the FSM, the file system will give up or continue retrying forever. For **hard** mounted file systems, the kernel will retry the connection attempt forever, regardless of the value of the *retrans* field. For **soft** mounted file systems, the kernel will only try *retrans* number of times before giving up and returning an error of **ETIME** (62). This is analogous to the *hard* and *soft* options to NFS (see **fstab**(5)).

timeout=*n*
Default: **100** (ten seconds)

The timeout value, in tenths of a second (0.1 seconds) to use when sending message to the FSM. This timeout parameter is similar to the one used by NFS (see **fstab**(5) for more information on NFS timeouts). If no response is received from the FSM in the indicated period the request is tried again. On heavily loaded systems, you may want to adjust the timeout value higher.

syslog={**none**|**notice**|**info**|**debug**}
Default: **notice**

During normal operations, certain messages will be logged to the system console using the *syslog* facility. **debug** is the most verbose, with **notice** being reserved for critical information. It is important to note that the syslog level is **global** per system, not unique to each file system. Changing the level for one file system will affect all other SNFS file systems.

blkbufsize=*n*
Default: Linux **512K** Windows **64K**

This option sets the maximum buffer size, in bytes, for the unaligned I/O transition buffer. Use caution when setting this option since values that are too small may degrade performance or produce errors when performing large unaligned I/O.

dircachesize=*n*
Default: **10 MB**

This option sets the size of the directory cache. Directory entries are cached on the client to reduce client-FSM communications during directory reads. Note: the directory cache on a client is shared across all mounted SNFS file systems. If different values of dircachesize are specified for multiple file systems, the maximum is used. When applying this setting, ensure that the system has sufficient kernel memory.

Can be specified in bytes (e.g. 2097152), kilobytes (e.g. 2048k), or megabytes (e.g. 2m).

auto_dma_read_length=*n*
Default: **1048577 Bytes (1MB + 1)**

The minimum transfer size used for performing direct DMA I/O instead of using the buffer cache for well-formed reads. All well-formed reads equal to, or larger than this value will be transferred with DMA. All read transfers of a smaller size will use the buffer cache. Reads larger than this value, that are not well-formed will use a temporary memory buffer, separate from the buffer cache.

The minimum value is the cachebufsize. By default, well-formed reads of greater than 1 Megabyte will be transferred with DMA; smaller reads will use the buffer cache.

Auto_dma_read_length can be specified in bytes (e.g. 2097152), kilobytes (e.g. 2048k), or megabytes (e.g. 2m).

auto_dma_write_length=*n*
Default: **1048577 Bytes (1MB + 1)**

The minimum transfer size used for performing direct DMA I/O instead of using the buffer cache for well-formed writes. All well-formed writes equal to, or larger than this value will be transferred with DMA. All write transfers of a smaller size use the buffer cache.

The minimum value is the cachebufsize. By default, well-formed writes of greater than 1 Megabyte will be transferred with DMA; smaller writes will use the buffer cache. Writes larger than this value, that are not well-formed will use a temporary memory buffer, separate from the buffer cache.

Auto_dma_write_length can be specified in bytes (e.g. 2097152), kilobytes (e.g. 2048k), or megabytes (e.g. 2m).

buffercache_readahead=*n*
Default: **16**

This option modifies the size of the read-ahead window, represented in cache buffers. Setting this value to 0 disables read-ahead.

buffercache_iods=*n*
Default: **8**, Minimum: **1**, Maximum: **100**

The number of background daemons used for performing buffer cache I/O.

cvnode_max=*n*
Default: varies by platform.

This option sets the maximum number of cvnode entries cached on the client. Caching cvnode entries improves performance by reducing Client-FSM communication. However, each cached cvnode entry must be maintained by the FSM as well. In environments with many SNFS clients the FSM may be overloaded with cvnode references. In this case reducing the size of the client cvnode cache will alleviate this issue.

max_dma=*n*
LINUX ONLY

Default: varies by platform.

This option tells the kernel the maximum DMA size a user process can issue. This can impact the number of concurrent I/Os the file system issues to the driver for a user I/O. There are other factors that can also limit the number of concurrent I/Os. The default is 512m
on Linux. WARNING: Incorrectly setting this value may degrade performance or cause a crash/hang.

max_dev=*n*
LINUX ONLY

Default: **Linux: 512M with Linux DM/Multipath. 512K with StorNext multipath.**

This option tells the kernel the maximum I/O size to use when issuing I/Os to the underlying disk driver handling a LUN. The file system attempts to get the maximum I/O size using the IOCINFO ioctl. Since the ioctl is not always reliable, this mount option exists to override the ioctl return value. Example usage max_dev=1m or max_dev=256k. WARNING: Incorrectly setting this value may result in I/O failures or cause a crash/hang. For Linux clients, only use when recommended by Quantum Support.

sparse={**yes**|**no**}
Default: varies by platform.

Some utilities detect "holes" in a file and assume the file system will fill the hole with zeroes. To ensure that SNFS writes zeroes to allocated, but uninitialized areas on the disk, set *sparse=yes*.

max_dma_ios=*n*
LINUX ONLY Default: **0** (disabled)

This option controls the amount of dma based I/O which is can be queued while there is buffer cache based I/O pending. The numerical value is multiplied by the cachebufsize (256K or 1024K default), once this amount of buffered I/O is pending, all DMA I/O will be suspended until some of the buffered I/O completes. This gives buffered I/O some priority over heavy DMA loads. If you do not understand how to use this for your workload, do not use it.

io_penalty_time=*n*
Default: **300** *(seconds)*

This option controls the Multi-Path I/O penalty value, where *n* is expressed in seconds with a minimum value of *1* and a maximum value of 4294967295 (*0xFFFFFFFF*). This parameter establishes the amount of time that a *Path_In_Error* will be bypassed in favor of an *Operational Path* during a Multi-Path selection sequence. If all paths are currently in the *Path_In_Error* state, then the first available path will be selected, independent of the *Path_In_Error* state.

io_retry_time=*n*
Default: **0** *(Forever)*

This option controls the I/O retry behavior. *n* is expressed in seconds (**0** is no time limit), and establishes the amount of time that may elapse during an I/O retry sequence. An I/O retry sequence consists of:

1. Retry an I/O request across all available paths that are currently present.
2. Compare the current time to the *Instantiation* time of the I/O request, if at least *n* seconds have elapsed, return the I/O request in error, otherwise reset the paths used, and retry again.

iso8859={**1**|**15**}
Default: disabled

This option provides conversion of file and directory names from legacy ISO-8859-1 and ISO-8859-15 to SNFS native UTF8 format. This may be needed in environments where a critical application does not yet support UNICODE and the required locale is ISO-8859-1 or ISO-8859-15.

This option is supported on Linux and Solaris.

Limitations:

1. cannot be enabled on SNSM metadata server (clients only).
2. filesystem names must be plain ascii.
3. pre-existing filesystems with erroneous ISO-8859 object names must be manually converted to UTF8 before using this feature. Note, this can be easily accomplished with a perl script using Encode::from_to().
4. does not solve issues surrounding composed versus decomposed presentation.

caseinsensitive={**yes**|**no**}
LINUX ONLY Default: **0** (no)

Causes a Linux client to evaluate path names in a case insensitive, case preserving mode. This is intended for use in SMB environments to reduce the overhead of emulating the behavior of a Windows file system on Linux in user space.

allowdupmount={**yes**|**no**}
Default: **no**

UNSUPPORTED - reserved for Quantum internal usage. Not intended for production use. When set to **yes**, **mount_cvfs** will allow multiple mounts for the same file system.

dmnfsthreads=*n*
Linux ONLY Default: 0.

Determines the number of threads used for servicing retrieves of off-line files initiated as a side ef-

fect of NFS activity. This option should only be used on Linux systems acting as NFS servers and is only useful for managed file systems. *dmnfsthreads* should be set to a value as least as large as the maximum number of concurrent retrieves possible for the file system. For example, with tape-based configurations, this will be equal to the number of tape drives configured for retrieval.

loopdev={**on**|**off**}
Linux ONLY

Default: off.

Enable support for loopback devices on top of Stornext. This causes heavy use of the linux page cache when loop devices are used, it will also change how an NFS server interacts with Stornext. Use of the option is not recommended unless loopback device or sendfile support is required.

hadoop={**yes**|**no**}
Linux ONLY

Default: no.

Change some client behaviors to be more in line with Hadoop file system support. In particular, deferred close does not happen, data is flushed on last close, and allocation sessions are no longer per directory.

class=text-string
Linux ONLY

Default: no value.

Pass the specified string to the FSM during connect. This allows the identification of a set of clients of a particular class from the FSM. The information is available via an xattr on files in the file system, via cvadmin and via a web service call. This is initially for Hadoop support.

ingest_max_seg_size=*n*
Default: 0.

Set the maximum size of ingest segments to *n* (bytes).

ingest_max_seg_age=*n*
Default: 0.

Set the maximum time to wait before sending ingest segment to *n* (seconds).

memalign=*n*
bufalign=*n*
Default: varies by platform.

Set the required memory alignment for dma transfers to *n*. Use of the option is not intended for general use and may be deprecated in the future. Only use when recommended by Quantum Support.

bcacheprefersreaders=<*type*>
Linux ONLY Default: none

Enable special buffer cache ingest throttling and LRU handling. This option is not intended for general use and incorrect use may lead to poor performance and system instability. Only use when recommended by Quantum Support.

parallel_submit

Linux and Windows

When large direct I/O is performed, submit request components in parallel using a pool of daemons. This can improve performance and reduce latency if the backend storage provides more than 10 GB/s of bandwidth. On slower storage, the performance improvement will not be noticeable. On Linux, transparent huge pages must be disabled or performance will actually be worse when using the option. Also, in order for the performance impact to be seen, the number of requests configured for the Linux block devices (nr_requests) must be large enough so that I/O submission does not block due to request structure exhaustion (see deviceparams(4)). Note that "large I/Os", here, are any having sizes at least 16 times the stripebreadth of the stripe group where the file resides.

This option is also available on StorNext Windows clients through the Enable Parallel I/O Submisison checkbox on the Advanced Mount Options tab in the Client Configuration Tool.

This option is not intended for general use and incorrect use may lead to poor performance and system instability. Only use when recommended by Quantum Support.

privatebufcache

Default: off

Use a private data buffer cache for the file system even when there may be other mounted file systems with the same cachebufsize. See **buffercachecap** above.

AUTOMATIC MOUNTING

On Linux, the initialization of SNFS can be controlled by the **chkconfig**(8) mechanism. If the **cvfs** chkconfig flag is set to **on**, then all SNFS file systems specified in the */etc/fstab* file will be mounted when the system is booted to multi-user state. When the system is being shut down, the file systems will be unmounted. See the **cvfs** man page for more information.

On Solaris, the installation of CVFS adds a startup script to */etc/init.d/* that will automatically mount CVFS file systems present in the */etc/vfstab* file with a "yes" keyword in the "mount at boot" column.

DISK DEVICES

mount_cvfs will query the local portmapper for the list of all accessible SNFS disk devices. SNFS disks are recognized by their label. This list is matched with the list of devices for each stripe group in the file system. If any disk is missing, I/O will be prohibited, and you will receive I/O errors.

RECONNECT

A socket is maintained for each unique SNFS client file system for sending and receiving commands to and from the FSM. If the socket connection is lost for any reason, it must be reconnected.

There are two daemons involved in re-establishing the connection between an SNFS client and the FSM. The first is the socket input daemon, which is a dedicated daemon that handles all input from the FSM. The second is the reconnect daemon, which handles the work of re-establishing the logical connection with the FSM. Both of these daemons appear as **cvfsd** in the output from **ps**.

Messages will be printed on the system console and to *syslog* during reconnect processing; the verbosity of the messages displayed can be controlled via the **syslog=** parameter and **cvdb**(8).

When the socket input daemon detects that the connection has been lost, it will attempt to first connect to the fsm portmapper process, **fsmpm**(8). Once it has succeeded and has the port number of the **fsm**(8) to use, it attempts to create a new socket to the FSM using the port number returned by fsmportmapper.

If no response is received from either the SNFS portmapper or the FSM, the daemon will pend for the amount of time specified by the **timeout=** parameter. The socket input daemon will attempt to reconnect to the FSM forever.

If any of the configuration parameters in the FSM configuration file changed, then the connection will be terminated, and no further I/O will be allowed. The only recourse will be to unmount and remount the file systems. See **snfs_config**(5) (part of the *cvfs_server* product) for more information on configuring the FSM.

INTERRUPTIBLE SLEEPS

Whenever a process must go to sleep in the SNFS file system, the sleep is interruptible, meaning that the process can be sent a signal and the operation will fail with an error (usually **EINTR**). The only exceptions are when a process is executing the **exit**(2) system call and is closing out all open files; due to Unix limitations, processes are immune to signals at that point.

EXAMPLES

File systems can be specified either on the command line or in */etc/fstab*. To mount the default file system that is served by a host in the SAN, the entry in */etc/fstab* would be:

```
default /usr/tmp/cvfs cvfs verbose=yes
```

If this is the only SNFS file system in */etc/fstab*, it could be mounted with the command:

```
# mount -at cvfs
```

To mount the same file system, but specifying a soft connection with a retransmit count of two, and a soft background mount with a retry count of two, the entry in */etc/fstab* would be (line is shown broken up for readability; in practice, it would wrap):

```
default /usr/tmp/foo cvfs verbose=yes,recon=soft,retrans=2,
      mnt_recon=soft,mnt_retry=2,mnt_type=bg
```

To mount a file system in a cluster other than your default, use an */etc/fstab* entry similar to the following:

```
snfs1@cluster1 /stornext/snfs1 cvfs rw 0 0
```

And the corresponding mount command would be:

```
mount snfs1@cluster1
```

Filesystems can also be specified on the command line, without an entry in */etc/fstab*. To mount the default file system on mount point */usr/tmp/foo*:

```
mount -t cvfs default /usr/tmp/foo
```

The command to mount a filesystem not in your default cluster would be:

```
mount -t cvfs snfs1@cluster1 /stornext/snfs1
```

To mount a file system verbosely that is described by the FSS configuration file *mycvdr.cfgx* on that host:

```
mount -o verbose=yes -t cvfs mycvdr /usr/tmp/foo
```

LIMITATIONS

Only the **Linux** and **Unix** platforms are supported with the mount helper mount_cvfs

For Windows instructions mounting filesystemes follow the "StorNext - Getting Started" section of the help page.

SEE ALSO

cvfsd(8), **cvdb**(8), **mount**(8), **chkconfig**(8), **fsmcluster**(4) , **sndpscfg**(8), **dpserver**(4), **deviceparams**(4)

NAME

mount_snmetadb − Readonly fuse mount of StorNext metadata archive image

SYNOPSIS

fuse mount StorNext metadata archive image as a file system.

mount_snmetadb [**-o** *options*] *fsname /mountpath*

WARNING: mount_snmetadb is experimental and its behavior may change in future releases.

DESCRIPTION

mount_snmetadb presents a metadata archive image as a readonly fuse file system. The metadata archive image can either be local on disk files, or be remotely hosted in an S3 object store. File content can be read for files stored by storage manager to an object store, or by snmetadb itself in the case of non-managed file systems. The fuse mount may then be further exported via Samba and/or NFS. **mount_snmetadb** defaults to looking for the json specification of a remote metadata archive copy, the json is generated by **snmetadb** on the source system and needs to be copied to the target host and placed in /usr/cvfs/backups/inventory.

The system running the mount needs to have the access information for the object store placed in

/usr/cvfs/config/snmetadb_access.json

See the snmetadb man page for more information here.

Alternatively, a local metadata archive copy can be mounted, or a json file in a different location specified, both of these use the -o dump option.

-o dump=/path/to/json-file

-o dump=/path/to/dump-directory

The archive-directory itself can be the live metadata archive of the running FSM, or a backup copy of a metadata archive. StorNext does not need to be running to mount a dump image, including reading file content. After the command is started up, it will background itself until the unmount command is issued.

The fuse filesystem content will not automatically update from the dump copy, however a refresh can be triggered by reading the metadb.refresh extended attribute from any file or directory.

getfattr -n metadb.refresh .

LOG FILE

mount_snmetadb logs activity to the file /usr/cvfs/debug/mount-snmetadb.log

SEE ALSO

snmetadb(8)

NAME

nss_cctl − StorNext Cluster-Wide Central Control File

SYNOPSIS

/usr/cvfs/config/nss_cctl.xml

DESCRIPTION

The **StorNext File System** supports cluster-wide central control to restrict the behavior of SNFS cluster nodes (fsm server, file system client and administrative commands such as cvadmin) from a central place. A central control file *nss_cctl.xml* is used to specify the desired controls on the cluster nodes. This file resides under */usr/cvfs/config* on an **nss coordinator server.** The filename of the nss_cctl file follows the format:

nss_cctl.xml

NOTE: The recommended way to get started with the central control facility is to first use the **nss_cctl_template**(8) commmand to generate a file that can then be edited to form your *nss_cctl.xml* file. The addresses of hosts and names of filesystems will be those that are currently in use and mounted in the cluster in which you are running the template command.

This control file is in **xml** format and has hierarchical structure. The top level element is **snfsControl**. It contains control element **securityControl** for certain file systems. If you have different controls for different file systems, then each file system should have its own control definition. A special virtual file system **#SNFS_ALL#** is used as the default control for file systems not defined in this control file. It is also the required file system name when configuring the **snfsAdmin** and **snfsAdminControl** options. *NOTE*: You cannot have a real file system named as **#SNFS_ALL#**.

Each file system related control element (i.e. **securityControl**) has a list of **controlEntry**. Each **controlEntry** defines the client and the intended controls. The simplest and preferred way of defining a client within the **controlEntry** is in the following manner:

<client>
    <address value="*value***"/>**
</client>

where *value* can either be an IP address (or hostname) by itself, or followed by a netmask length separated by a slash (e.g "192.0.2.0/24") if one would like to specify a subnet. An IP address by itself is equivalent to the same IP address followed by the maximum netmask length of the IP version in use (e.g. "192.0.2.10" is equivalent to "192.0.2.10/32"). Both IPv4 and IPv6 are supported. There may be multiple **address** entries within the **client** element if the specified addresses will be sharing the same controls within the **controlEntry**.

A value which matches any client can be specified by using a zero length netmask, for example "0.0.0.0/0".

Another way of defining a client is supported for backwards compatibility, wherein its type is explicitly defined: **host** or **netgrp**. For type **host**, the client is defined in the following format below, where *value* may be either an IP address or hostname:

<client type="host">
    <hostname value="*value***"/>**
</client>

For type **netgrp**, two sub-elements **network**, which defines the IP address of the subnet, and **maskbits**, which defines the netmask length of the subnet, need to be included as follows:

<client type="netgrp">
    <network value="*value***"/>**
    <maskbits value="*value***"/>**
</client>

NOTE: When there is overlap between client IP addresses, the controls which correspond to the IP address with the longest netmask length will take precedence.

Controls

Currently eight controls are supported. Each control has this format:

<*control* **value="***value***"/>**

The *value* can be either **true** or **false**. The *control* is one of the following controls:

mountReadOnly

Controls whether the client should mount the given file system as readonly. Value **true**(**false**) means the file system is mounted as readonly (read/write). If this control is not specified, the default is read/write.

mountDlanClient

Controls whether the client can mount the given file system via proxy client. Value **true**(**false**) means the file system is (not) allowed to mount via proxy client. The default is "mount via proxy client not allowed".

takeOwnership

Controls whether users on a windows client are allowed to take ownership of file or directory of a stornext file system. Value **true**(**false**) means that taking ownership is (not) allowed. The default is that "take ownership is not allowed". *Note*: this control only applies to the clients running on Windows platforms.

snfsAdmin

Controls whether **cvadmin**(8), **sgmanage**(8), and **snquota**(1) running on a host are allowed to have **super-admin** privilege to run **privileged** commands such as start/stop a file system, manipulate stripe groups or change quota settings. Value **true**(**false**) means users with **superadmin** privilege is (not) allowed to run privileged commands. The default value is "false". This option can only be defined under the **#SNFS_ALL#** file system name.

snfsAdminConnect

Controls whether **cvadmin**(8) running on a client is allowed to connect to other fsm host via **-H** option. Value **true**(**false**) means **cvadmin**(8) is (not) allowed to connect to other fsm host via **-H** option. The default value is **false**. This option can only be defined under the **#SNFS_ALL#** file system name.

exec Controls whether binary executable files on the file system are allowed to be executed. Value **true**(**false**) means binary executable files are (not) allowed to be executed. The default value is **false**, i.e. the execution is not allowed. Note: the StorNext upgrade process may rely on the ability to run binary executables residing on the HA file system. Therefore, setting exec to "true" for this file system is required, at least during upgrades.

suid Controls whether setuid bit is allowed to take effect. Value **true**(**false**) means setuid bit is (not) allowed to take effect. The default value is **false**.

denyRetrieves

Controls whether a client is permitted to retrieve offline files by reading them. This only applies to managed file systems. When a file is offline, storage manager is responsible for retrieving its contents, this can usually be triggered by a file read. If read based retrieve is disabled then an fsretrieve command must be used, either directly, or via a web services call. Value **true**(**false**) means that the client cannot (can) initiate retrieves. The default value is **false**.

globalSuperUser

Controls whether a client can override the file system configuration for global super user. This nss_cctl variable only has meaning when the file system configuration variable **globalSuperUser** has been set to **false**, disabling global super user for clients. When this nss_cctl variable is set to **true** for a set of clients, these clients behave as if the configuration variable had been set to **true**. See the **snfs_config**(5) man page for a complete description of this privilege. The default value is **false**. Apple Xsan clients do not honor the setting of **globalSuperUser**.

NOTE: If no match is found for a given client's IP address, then the client has no privilege to access a SNFS cluster. If a file system has been defined but the client is not defined in that file system's control (security-

Control), then the client has no access privilege to the specified file system.

Non-voting and Voting client configuration

The element **nonVotingCluster** can be included (on the same level as the **securityControl** element) to set the default client behavior (voting or non-voting) within the cluster during the election that will choose the host on which a specific file system manager will run. The cluster to which this control is applied will be the one specified by the nss_cctl filename. *NOTE*: If no cluster is specified in the filename, please refer to the beginning of this man page's **DESCRIPTION** section for more information on which cluster this control will take effect.

The element **nonVotingCluster** has the following format:

<nonVotingCluster value="*value***"/>**

where *value* can either be **true** or **false**. If *value* is **true**, the default behavior of all clients within the cluster will be to abstain from voting in the election. In effect, unnecessary NSS message traffic may be greatly reduced. If the **nonVotingCluster** is not included in the xml file, client behavior will be the same as if its value were set to **false** (i.e. all clients within the cluster will be voting in the election).

Note that clients running versions of StorNext prior to StorNext 6.0 use an older version of the NSS protocol (NSS1), which does not honor the non-voting cluster configuration options specified in this file.

NOTE: There always needs to be voting clients within the cluster so that a decision can be derived from the election. Therefore, when the **nonVotingCluster** element is set to **true**, it should be used in conjunction with the **votingClients** element (described in the following paragraphs) which allows one to specify an explicit list of voting clients.

It is also possible to specify a group of non-voting clients within a cluster by creating a list of client addresses with the **nonVotingClients** element (also used on the same level as that of the **securityControl** element). The **nonVotingClients** element has the following format:

<nonVotingClients>
<address value="*value***"/>**
<address value="*value***"/>**
.
.
.
<address value="*value***"/>**
</nonVotingClients>

where each **address** element is the same element used when specifying a client within a **controlEntry**, and there must be at least one **address** element. To specify a group of voting clients, the same format is used but replacing **nonVotingClients** with **votingClients**.

NOTE: For more information on the **address** element and its allowed values, please refer to the beginning of this man page's **DESCRIPTION** section.

NOTE: All three elements (i.e. **nonVotingCluster**, **nonVotingClients** and **votingClients**) may be in the *nss_cctl.xml* file at the same time. The **votingClients** and **nonVotingClients** elements will take precedence over the **nonVotingCluster** element. When a client IP address matches elements in both **nonVotingClients** and **votingClients**, the element with the longest netmask will take precedence; if there is a tie, the **votingClients** element will be used.

LIMITATIONS

Only the **Linux** platform is supported to be a nss coordinator server capable of parsing this xml file.

EXAMPLE

The following is an example of *nss_cctl.xml* file. It defines the control of file system **snfs1**, and also the special virtual file system **#SNFS_ALL#**.

NOTE: As stated earlier, instead of using this example, you can generate a starting point *nss_cctl.xml* file using the **nss_cctl_template**(8) command.

```
<snfsControl xmlns="http://www.quantum.com/snfs/cctl/v1.0">
    <nonVotingCluster value="true"/>
    <votingClients>
        <address value="192.0.2.108/24"/>
        <address value="198.51.100.215"/>
    </votingClients>
    <securityControl fileSystem="snfs1">
       <controlEntry>
            <client>
                <address value="192.0.2.108"/>
                <address value="198.51.100.215"/>
            </client>
            <controls>
                <mountReadOnly value="false"/>
                <mountDlanClient value="false"/>
                <takeOwnership value="false"/>
                <exec value="true"/>
                <suid value="false"/>
            </controls>
        </controlEntry>
      <controlEntry>
          <client type="host">
            <hostName value="192.0.2.132"/>
          </client>
          <controls>
            <mountReadOnly value="true"/>
            <mountDlanClient value="true"/>
            <takeOwnership value="false"/>
            <exec value="true"/>
            <suid value="false"/>
          </controls>
      </controlEntry>
      <controlEntry>
          <client type="netgrp">
            <network value="192.0.2.0"/>
            <maskbits value="24"/>
          </client>
          <controls>
            <takeOwnership value="true"/>
            <mountReadOnly value="true"/>
                <exec value="true"/>
                <suid value="false"/>
          </controls>
      </controlEntry>
    </securityControl>
    <securityControl fileSystem="#SNFS_ALL#">
      <controlEntry>
          <client type="host">
            <hostName value="linux_ludev"/>
          </client>
          <controls>
            <snfsAdmin value="true"/>
            <snfsAdminConnect value="true"/>
                <exec value="true"/>
```

```
                        <suid value="false"/>
                    </controls>
                </controlEntry>
            </securityControl>
        </snfsControl>
```

FILES

/usr/cvfs/config/nss_cctl.xml
/usr/cvfs/examples/nss_cctl.example

SEE ALSO

cvadmin(8), **fsnameservers**(4), **nss_cctl_template**(8), **sgmanage**(8), **snfs_config**(5), **snquota**(1)

NAME

Central control facility template generator

SYNOPSIS

na

nss_cctl_template [-h] [--host HOST] [FILE]

DESCRIPTION

nss_cctl_template is a utility to create an initial configuration file for the central control facility. It uses the **snprobe** command to scan the existing cluster as seen from the host it is run on and generates an initial **nss_cctl.xml** file. The output is a permissive version of the control file with all features enabled. This can be used as a basis for a version which restricts the type of access allowed for different clients. The output needs to be placed in /usr/cvfs/config/nss_cctl.xml on the cluster name server hosts. Output defaults to std-out unless a file name is specified.

By default the cluster is scanned from the local host, this can be overridden using the --host option.

SEE ALSO

nss_cctl(8)

NAME

qbm-analyze − Qos Bandwidth Management Measurement Analyzer

SYNOPSIS

qbm-analyze --action *eval* **--fsname** *FsName* **--testid** *test_number* [**--verbose**]

qbm-analyze --action *eval* **--db** *database_name* **--testid** *test_number* [**--verbose**]

qbm-analyze --action *list* **--fsname** *FsName* [**--testid** *test_number*] [**--verbose**]

qbm-analyze --action *list* **--db** *database_name* [**--testid** *test_number*] [**--verbose**]

qbm-analyze --action *testid*|*streams*|*client_name* **--fsname** *FsName*

qbm-analyze --action *testid*|*streams*|*client_name* **--db** *database_name*

qbm-analyze [**--help**]

DESCRIPTION

qbm-analyze is used to analyze qbm-ladder test results to determine the maximum bandwidth for a file system stripe group using multiple clients, multiple streams, variable queue depth and variable buffer sizes. The database is queried and results are compared to determine the maximum bandwidth. The number of times this bandwidth was reached is included, as this can show there was more bandwidth available than could be used by the test parameter combinations. Use of the verbose keyword prints individual test parameters and the resulting bandwidth.

The command requires either the file system name to locate the database or the name of the database.

The database can be queried using the action list keyword to list the test ids retained in the database and the parameters used for those test ids. The test ids, number of streams and client names can also be listed.

OPTIONS

Options that start with -- and take an argument can have the argument separated by either a space or an equal sign, e.g.

--fsname *FsName*
--fsname=*FsName*

are equivalent.

--action *eval*|*list*|*testid*|*streams*|*client_name*
Determines the action taken. The "eval" action causes the test results for the specified test number to be evaluated and a maximum bandwidth determined. The "list" action prints the test numbers found in the database with their associated parameters and time when qbm-ladder was started. The "testid" action prints the unique test numbers available for analysis and the date and time they started execution. The "streams" action prints the stream variables found for all the test numbers. The "client_name" action prints the clients where the test executed for all the test numbers.

--fsname *FsName*
Selects a given file system.

--db *database_name*
Uses the specified database.

--verbose
When specified with the eval action, causes individual test parameters and bandwidth results to be printed.

--help Displays usage information.

EXAMPLES

Analyze results for test number 1234.

```
rock # qbm-analyze --fsname snfs1 --action eval --testid 1479
```

```
Test id 1479
Largest bandwidth is 173MB/s which occurred in 1 out of 12 tests.
```

Analyze results for test number 2238 and include individual test results.

```
[root@snt018163-mdc1 bin]# qbm-analyze --fsname snfs1 --action eval --testid 2238

Test id 2238
Largest bandwidth is 367MB/s which occurred in 1 out of 36 tests.

MB/sec    clients  streams  queue_depth   buffersz
 93        1       2         1           500k
 30        1       2         1           500k
 101       2       2         1           500k
 142       2       2         1           500k
 101       2       2         1           500k
 284       2       2         1           500k
 101       1       2         1           1024k
 97        1       2         1           1024k
 102       2       2         1           1024k
 145       2       2         1           1024k
```

Show list of all test ids and parameters in tabular form.

```
rock # qbm-analyze --action=list --fsname --snfs1
Test id   Streams   Clients   Buffer size   Time test started
1318       1       1       500k     Tue Feb  6 12:47:14 2018
2238       2       1       1024k     Tue Feb  6 12:59:47 2018
2238       2       1       500k     Tue Feb  6 12:59:47 2018
2238       2       2       1024k     Tue Feb  6 12:59:47 2018
2238       2       2       500k     Tue Feb  6 12:59:47 2018
25042       1       1       500k     Mon Feb  5 21:06:06 2018
25189       1       1       500k     Mon Feb  5 21:10:08 2018
25309       1       1       500k     Mon Feb  5 21:10:58 2018
25439       1       1       500k     Mon Feb  5 21:13:43 2018
25567       1       1       500k     Mon Feb  5 21:15:13 2018
25691       1       1       500k     Mon Feb  5 21:17:44 2018
368       1       1       500k     Tue Feb  6 12:19:07 2018
531       1       1       500k     Tue Feb  6 12:28:00 2018
742       1       1       500k     Tue Feb  6 12:30:31 2018
870       1       1       500k     Tue Feb  6 12:31:02 2018
```

SEE ALSO

qbm-ladder(8)

NAME

qbm-ladder − Qos Bandwidth Management Measurement Ladder Test

SYNOPSIS

qbm-ladder --fsname *FsName* **--clients** *client_list* [**--bufsize** *size_list*] [**--qd** *queue_depth_list*] [**--streams** *number_list*] [**--sgnum** *sg_number*] [**--long**] [**--short**] [**--csvbase** *file_name_base*] [**--mntpt** *path*] [**--winmpt** *path*] [**--outputdir** *path*] [**--totalxmit** *bytes*] [**--seconds** *runtime*] [**--schema** *schema_file*] [**--db** *database_name*] [**--group_id** *group_id*]

qbm-ladder --help

DESCRIPTION

qbm-ladder is used to run a set of performance tests using combinations of user specified parameter lists. The user can specify lists for clients, queue depth, number of streams and buffer size. The lists are comma separated. The client list can include an alternate user name using the form <user>@<client>. The order of the client list is significant in that it specifies the order clients are added into the combination. The first client will be run as a single client, then the second client is added for running 2 clients simultaneously, with the third client added for running 3 clients simultaneously and so on.

qbm-ladder uses ssh for running processes on client systems. Setting up ssh remote passwordless login must be done to ensure tests are running simultaneously on client systems. See ssh-keygen and ssh-copy-id utilities to assist setup. Failure to set this up will result in multiple password prompts.

OPTIONS

Options that start with -- and take an argument can have the argument separated by either a space or an equal sign, e.g.

--fsname *FsName*
--fsname=*FsName*

are equivalent.

--fsname *FsName*
Selects a given file system.

--clients *client_list*
Comma separated list of names which selects where the tests will run. The client names can include the username using the syntax <username>@client.

--bufsize *size_list*
List of buffer sizes used for I/O. If the argument is not specified, it defaults to 64K,128K,256K,512K,1M,2M,4M,8M,12M,16M,32M. Optionally, a single letter suffix can represent bytes in units. The single letter unit suffixes are as follows.

k	KB	10^3	1,000
m	MB	10^6	1,000,000
b	GB	10^9	1,000,000,000
K	KiB	2^10	1,024
M	MiB	2^20	1,048,576
B	GiB	2^30	1,073,741,824

--qd *queue_depth_list*
List that specifies the number of asynchronous read or write requests that are queued to each stream.
If the argument is not specified, it defaults to 1,2,4,8.
The maximum is 256.

--streams *number_list*
Number of files used simultaneously for read/write. Each file is written by a different thread.
If the argument is not specified, it defaults to
1,2,3,4,6,8,10,12,14,15,16.

--sgnum *number*
Number of the stripe group being tested. The default is 1.

--short If set, modifies the default list values.
If the argument is specified, defaults are
bufsize="64K,512K,2M,8M"
qd="2"
streams="1,2,4,8"
seconds=300
This will run a minimum of 16 combinations times the number of clients. Each combination will run for a maximum of 5 minutes. Any value can be overridden by specifying it explicitly.

--long If set, modifies the default list values.
If the argument is specified, defaults are
bufsize="64K,256K,512K,1M,2M,8M"
qd="1,2,4,6,8"
streams="1,2,4,6,8,9,10,12,16"
This will run a minimum of 270 combinations times the number of clients. There is no time limit for each combination.

--csvbase *name*
Base of file name used to create file names for csv output from **Mio**.
If the argument is not specified, it defaults to "base".

--mntpt *path*
Specifies mount point of file system on linux systems.
If the argument is not specified, it defaults to /stornext/<fsname>/.

--winmpt *path*
Specifies mount point of file system on windows systems.
If the argument is not specified, there is no default.

--outputdir *path*
Directory where output csv files and the database are placed.
If the argument is not specified, it defaults to /usr/cvfs/data/<fsname>/qbm.

--totalxmit *number_bytes*
Total amount of data to read and write to each file.
If the argument is not specified, it defaults to 1GB.

--db *database file*
Name of sqlite database for storing test parameters and results.
If the argument is not specified, it defaults to qbm-db.

--schema *name*
Database schema table location.
If the argument is not specified, it defaults to /usr/cvfs/bin/qbm-ladder-schema.

--groupid *group id*
This value is placed in the database, but not currently used for identification of tests.
This argument defaults to 1.

--help Displays usage information.

EXAMPLES

Run Mio tests on clients cl1 and cl2 using the combinations:
clients=cl1 bufsize=1M qd=2 streams=2
clients=cl1 bufsize=1M qd=2 streams=4

clients=cl1 bufsize=1M qd=2 streams=8
clients=cl1,cl2 bufsize=1M qd=2 streams=2
clients=cl1,cl2 bufsize=1M qd=2 streams=4
clients=cl1,cl2 bufsize=1M qd=2 streams=8

```
rock # qbm-ladder --fsname snfs1 --clients cl1,cl2 --bufsize 1M --qd 2 --streams
```

Run Mio tests on clients cl1, cl2 and cl3 using the following combinations, repeated for "cl1,cl2" and "cl1,cl2,cl3":
clients=cl1 bufsize=512K qd=2 streams=2
clients=cl1 bufsize=512K qd=2 streams=4
clients=cl1 bufsize=512K qd=4 streams=2
clients=cl1 bufsize=512K qd=4 streams=4
clients=cl1 bufsize=1M qd=2 streams=2
clients=cl1 bufsize=1M qd=2 streams=4
clients=cl1 bufsize=1M qd=4 streams=2
clients=cl1 bufsize=1M qd=4 streams=4

```
rock # qbm-ladder --fsname --snfs1 --clients cl1,cl2,cl3 --streams 2,4 --bufsize
```

SEE ALSO

qbm-mio(8)

NAME

qbm-mio − Qos Bandwidth Management Measurement

SYNOPSIS

qbm-mio --fsname *FsName* **--clients** *client_list* [**--bufsize** *iosize*] [**--qd** *queue_depth*]
[**--streams** *number_streams*] [**--filebase** *file_name_base*] [**--csvbase** *file_name_base*]
[**--mntpt** *path*] [**--winmpt** *path*] [**--outputdir** *directory_for_output*] [**--totalxmit** *bytes*]
[**--seconds** *seconds*] [**--keepcsv***=[1|0]]* [**--Rflag***=0|1***]**

qbm-mio --help

DESCRIPTION

qbm-mio is used to run Mio on StorNext client systems. An Mio process is started on each of the specified clients using the parameters specified and the ouput of the tests are returned and placed in outputdir for analysis. **qbm-mio** is used by **qbm-ladder** to run a sequence of tests.

qbm-mio uses ssh for running processes and commands on client systems. Setting up remote passwordless login must be done to ensure tests are running simultaneously on client systems. See ssh-keygen and ssh-copy-id utilities to assist setup.

OPTIONS

Options that start with -- and take an argument can have the argument separated by either a space or an equal sign, e.g.

--fsname *FsName*
--fsname=*FsName*

are equivalent.

--fsname *FsName*
Selects a given file system.

--clients *client_list*
Comma separated list of names which selects where the tests will run. Can include an alternate user name with the syntax <user>@<host>.

--bufsize *iosize*
Buffer size used for I/O. The default is 1M.

--qd *queue_depth*
Number of I/O's in each read or write queue. The default is 2.

--streams *number_streams*
Number of files used simultaneously for read/write on each client. The default is 1.

--filebase *filename_base*
All filenames created for I/O will begin with this name. If the argument is not specified, it defaults to "testfile".

--csvbase *filename_base*
All filenames for csv output will start with this name. The default is NULL.

--mntpt *path*
Specifies mount point of file system. If the argument is not specified, it defaults to /stornext/<fs-name>/.

--winmpt *path*
Specifies mount point of file system on windows systems. If the argument is not specified, it defaults to /cygdrive/z.

--outputdir *path*
Directory where output csv files are placed. If the argument is not specified, it defaults to /tmp/qbm.

--totalxmit *bytes*
Total number of bytes to read/write per stream. The default is 1GB.

--seconds *seconds*
Maximum number of seconds to run test. The test may run for less time, but not more. If the argument is not specified, there is no time limit.

--keepcsv
If set to 1, csv files created by Mio will not be removed from the client systems. The default is 0.

--Rflag If set to 1, reports I/O response times. The default is 0.

--help Displays usage information.

EXAMPLES

Run Mio tests on clients cl1 and cl2.

```
rock # qbm-mio --fsname snfs1 --clients cl1,cl2
```

Run Mio tests on clients cl1 and cl2 using 4 streams and I/O buffer size of 1M.

```
rock # qbm-mio --fsname --snfs1 --clients cl1,cl2 --streams 4 --bufsize 1M
```

Run Mio tests on clients using username "myuser".

```
rock # qbm-mio --fsname snfs1 --clients myuser@cl1,myuser@cl2
```

SEE ALSO

qbm-ladder(8)

NAME

qbm.conf − StorNext QOS Bandwidth Management configuration file

SYNOPSIS

/usr/cvfs/config/*_qbm.conf

DESCRIPTION

The *StorNext File System* (*SNFS*) **fsname_qbm.conf** file provides a way to configure Qos bandwidth management which handles I/O bandwidth for a file system on a client and stripe group basis. This file should not be edited by hand, but created and modified using the **qbmanage** command - see **qbmanage**(8) for details.

SYNTAX

The *fsname_qbm.conf* file follows the JSON schema. Each of the attribute-value pairs occupies one line and is terminated by a comma, save for the final attribute-value pair. There are 3 sections in the file for general parameters, stripe group configuration and client configuration. Stripe groups and clients are json arrays, and thus are enclosed by a pair of square brackets. No comments of any kind are allowed. Refer to this man page's **EXAMPLE** section for a visual representation of the described format. Refer to the **qbmanage**(8) manpage for creation of the file.

"**general**"
: Contains the options generally applicable and options that are used as default values.

"**stripe groups**"
: Contains the options for a given stripe group and default options used for clients not otherwise configured.

"**client**"
: Contains the options for a given client for a particular stripe group or if none is specified, for all stripe groups configured to use Qos bandwidth management.

NOTE: This file only needs to exist where the **fsm**(8) daemon will run, typically the metadata controllers.

EXAMPLE

The following is an example of the **fsname_qbm.conf** file.

```
{
   "general": {
      "on": true,
      "mover": false,
      "all_sg": true,
      "disallow_rate_change": false,
      "min_change": 0,
      "min_alloc": 32768,
      "capacity": 104857600,
      "reserved_first_come": 0,
      "reserved_fair_share": 0,
      "reserved_low": 0,
      "reserved_mover": 0,
      "default_minbw": 0,
      "default_maxbw": 0,
      "default_class": "fair_share",
      "activate": ""
   },
   "stripe_groups": [
      {
         "sname": "Data1",
         "capacity": 209715200,
         "default_class": "fair_share",
         "default_minbw": 0,
```

```
            "default_maxbw": 0,
            "reserved_first_come": 0,
            "reserved_fair_share": 0,
            "reserved_low": 0,
            "reserved_mover": 0
        },
        {
            "sname": "Data2",
            "capacity": 314572800,
            "default_class": "fair_share",
            "default_minbw": 0,
            "default_maxbw": 0,
            "reserved_first_come": 0,
            "reserved_fair_share": 0,
            "reserved_low": 0,
            "reserved_mover": 0
        }
    ],
    "clients": [
        {
            "cname": "10.65.191.253",
            "sname": "Data2",
            "default_minbw": 10485760,
            "default_maxbw": 209715200
        },
        {
            "cname": "10.65.170.237",
            "sname": "Data1",
            "default_minbw": 10485760,
            "default_maxbw": 209715200
        }
    ] }
```


FILES

/usr/cvfs/config/fsname_qbm.conf
/usr/cvfs/examples/fsname_qbm.conf.example

SEE ALSO

qbmanage(8), **cvadmin**(8)

NAME

qbmanage − Qos Bandwidth Management Utility

SYNOPSIS

qbmanage --help [**--command**] [**--verbose**]

qbmanage --new --fsname|**-f** *FsName* [**--on**=*[true|false|1|0]*] [**--mover**=*[true|false|1|0]*] [**--all_sg**=*[true|1]* **--sgcapacity** *amount*] [**--all_sg**=*[false|0]***]** [**--sgcapacity** *amount*] [**--activate** *when*] [**--min_change** *amount*] [**--min_alloc** *amount*] [**--reserved_fc** *amount*] [**--reserved_fs** *amount*] [**--reserved_low** *amount*] [**--reserved_mover** *amount*] [**--class** *fair_share|first_come|low_share|""*] [**--min_bw** *amount*] [**--max_bw** *amount*] [**--nofsm**] *common_options*

qbmanage --modify --fsname|**-f** *FsName* [**--on**=*[true|false|1|0]*] [**--mover**=*[true|false|1|0]*] [**--all_sg**=*[true|1]* **--sgcapacity** *amount***]** [**--all_sg**=*[false|0]***]** [**--activate** *when*] [**--min_change** *amount*] [**--min_alloc** *amount*] [**--sgcapacity** *amount*] [**--reserved_fc** *amount*] [**--reserved_fs** *amount*] [**--reserved_low** *amount*] [**--reserved_mover** *amount*] [**--class** *fair_share|first_come|low_share|""*] [**--min_bw** *amount*] [**--max_bw** *amount*] [**--timeout** *seconds*] [**--nofsm**] *common_options*

qbmanage --addsg --fsname|**-f** *FsName* **--sgname** *SgName* | '#SgIndex' [**--sgcapacity** *amount*] [**--reserved_fc** *amount*] [**--reserved_fs** *amount*] [**--reserved_low** *amount*] [**--reserved_mover** *amount*] [**--class** *fair_share|first_come|low_share|""*] [**--min_bw** *amount*] [**--max_bw** *amount*] [**--nofsm**] *common_options*

qbmanage --addclient --fsname|**-f** *FsName* **--clientname** *IpAddress* [**--sgname**|**-g** *SgName* | '#SgIndex'] [**--class** *fair_share|first_come|low_share|""*] [**--min_bw** *amount*] [**--max_bw** *amount*] [**--nofsm**] *common_options*

qbmanage --rmsg --fsname|**-f** *FsName* **--sgname** *SgName* | '#SgIndex' [**--nofsm**] *common_options*

qbmanage --rmclient --fsname|**-f** *FsName* **--clientname** *IpAddress* [**--sgname**|**-g** *SgName* | '#SgIndex'] [**--nofsm**] *common_options*

qbmanage --delete --fsname|**-f** *FsName* [**--nofsm**] *common_options*

qbmanage --start --fsname|**-f** *FsName* [**--timeout** *seconds*] *common_options*

qbmanage --reread --fsname|**-f** *FsName* [**--timeout** *seconds*] *common_options*

qbmanage --newconfig --fsname|**-f** *FsName* [**--timeout** *seconds*] *common_options*

qbmanage --putconfig --fsname|**-f** *FsName* [**--configfile** *filename***]** [**--timeout** *seconds*] *common_options*

qbmanage --getconfig --fsname|**-f** *FsName* [**--configfile** *filename***]** [**--timeout** *seconds*] *common_options*

qbmanage --status --fsname|**-f** *FsName* [**--timeout** *seconds*] *common_options*

qbmanage --validate --fsname|**-f** *FsName* [**--print**] [**--configfile** *filename***]** [**--nofsm**] *common_options*

DESCRIPTION

qbmanage is used to manage the QOS bandwidth management(QBM) configuration for file systems. QOS bandwidth management supports StorNext 6 SAN and LAN clients. The mover capability of QBM requires StorNext 6.2 (or newer) on all StorNext clients mounting the file system. **qbmanage** can be used to create or remove a configuration file, modify the general parameters, add and remove stripe groups, add and remove clients, and validate and print the configuration file. The configuration file has sections for general parameters, stripe group parameters and client parameters. See the **qbm.conf** manpage for more information. Creation or modification of the QBM configuration file requires restarting the file system or use of the **--newconfig** option.

Three classes are defined, with different behaviors. Each client mounting the file system is assigned to a class which determines the bandwidth allocation behavior. The classes are called first_come, fair_share and low_share. Each has a priority with first_come being the highest priority followed by fair_share and then low_share.

Alternatively, a mover configuration can be defined for throttling storage manager Distributed Data Movers (DDM). Other clients are not throttled and must not be configured.

The utility can also be used to signal the fsm to start QBM, to read the current QBM configuration file, to retrieve the configuration file or to replace the configuration file. The start command is available when the initial configuration file has the --on option set to false. To use these commands, a QBM configuration file must exist in the config directory before the related file system is started. The start command is only available when the initial configuration file has the --on option set to false.

When running this command on an HA pair, the putconfig command can only be run if the fsm is running on the primary node. The start, newconfig, getconfig and status can be run from both nodes of the HA pair. The validate command can be run on either node of an HA pair. All other commands must be run on the primary node of an HA pair, regardless of where the fsm is running. The configuration file must always be updated on the primary node of an HA pair.

If any of the following keywords, reserved_fc, reserved_fs, reserved_low, reserved_mover, are specified for a stripe group, the general reserved values will not be used for any of the reserved settings. If either the minimum bandwidth or the maximum bandwidth is specified for a stripe group or client, both of the values are considered specified. If the minimum bandwidth is specified without the maximum bandwidth, the maximum bandwidth is set equal to the minimum bandwidth. If the maximum bandwidth is specified without the minimum bandwidth, the minimum bandwidth is set to the default minimum bandwidth of 1048576.

OPTIONS

Options that start with -- can be abbreviated to just their unique prefix; however when called from scripts the full option name should be used to avoid future abbreviation conflicts if new options are added.

Numerical values can be specified or displayed using suffixes. Prior to 6.3, if **K, M, or G** was appended to the value, then the value was in multiples of 1024, 1048576 or 1073741824. Now suffixes specified as **K, M, G** or **KB, MB, GB** are in thousands, millions and billions. Suffixes specified as **KiB, MiB, GiB** are multiples of 1024, 1048576 or 1073741824. When displaying numerical values, if the value is evenly divisible by 1024, then suffixes of KiB, MiB and GiB are used. Setting the environment variable QBM_OLD_KMG forces the old behavior.

For options that begin with -- and have optional arguments, the argument must be preceded by an equal sign. Options that have required arguments can have the argument separated by either a space or an equal sign, e.g.

--fsname *FsName*
--fsname=*FsName*

are equivalent.

--fsname|**-f** *FsName*
Selects a given file system. If the file system is not running in the default cluster or administrative domain, those may added to the file system name using the syntax:
FsName[**@***cluster*[*/addom*]]

--sgname|**-g** *SgName* | '#SgIndex'
Selects a given stripe group by name or index. The hash sign, which is part of the index syntax, must be escaped to avoid special processing by the shell.

--new Creates a QBM configuration file with the specified general parameters. The file system must be restarted to activate the new configuration file. If it replaces an existing configuration, the **--newconfig** option may be used for activation.

--modify
Modifies the general parameters of an existing QBM configuration file.

--addsg
Adds a stripe group to an existing QBM configuration file.

--rmsg Removes a stripe group from an existing QBM configuration file.

--addclient
Adds a client to an existing QBM configuration file.

--rmclient
Removes a client from an existing QBM configuration file.

--delete
Removes an existing QBM configuration file.

--start Activates an existing QBM configuration file that specified the **--on** option as false. This is used to signal a file system to activate the configuration file that existed when the file system was started.

--newconfig
Activates the current QBM configuration file. This is used to replace the configuration being used by the file system.

--reread
Deprecated option which is the same as --newconfig.

--getconfig
Retrieve the active QBM configuration file from the file system. This requires that the file system be mounted on the system running the command. If no active configuration exists, nothing is returned. The default is to write to stdout.

--putconfig
Sends the specified QBM configuration file to the running file system to replace the configuration being used by the file system.

--status
Retrieve the QBM bandwidth information from the running file system and display the information.

--validate
Validate the current QBM configuration file.

--on=[*true*|*false*|*1*|*0*]
Change the initial state when the filesystem is started. If the argument is true, bandwidth management is immediately started. If false, the **qbmanage** must be used to initiate bandwidth management. If the argument is not specified, it defaults to **true**.

--mover=[*true*|*false*|*1*|*0*]
Specify the mover implementation of bandwidth management. Only configured clients have their bandwidth managed. Non-regulated clients report bandwidth usage which is used to manage the configured clients. If the argument is not specified, it defaults to **false**.

--class *first_come*|*fair_share*|*low_share*|""
Specify the name of the class to be used for the file system (general section), stripe group or client when used with the --new, --addsg, --addclient commands respectively. If mover is configured, no classes can be used. The class is assigned to a client by checking the client configuration, stripe group configuration and file system configuration in that order. If no class is found then the default value of low_share is assgined.

The first_come class has priority over all other classes. A client in this class that is granted bandwidth is guaranteed to retain its minimum bandwidth.

The fair_share class is second in priority for obtaining bandwidth. QBM shares allocation across clients in proportion to their configuration.

The low_share class is third in priority for obtaining bandwidth. QBM shares allocation across clients in proportion to their configuration. Unconfigured clients are assigned to this class by default.

--all_sg=[*true*|*false*|*1*|*0*]
Specify whether all data stripe groups will have their bandwidth managed. If the argument is true, bandwidth management is implemented on all data stripe groups. A bandwidth capacity is required for bandwidth managed stripe groups, thus **--sgcapacity** must also be specified with this option. If no configuration exists for a given stripe group, the **--sgcapacity** value specified as a general parameter will be used. If the argument is not specified, it defaults to **false**.

--min_change *min_change*
Set the value used as the minimum amount of change reported to a client. For example, a minimum change of 128KiB would not change the allocated bandwidth for a client if the change would be less than 128KiB difference from the previous value. If the argument is not specified, it defaults to **128KiB**.

NOTE: Not intended for general use. Only use when recommended by Quantum Support.

--min_alloc *min_alloc*
Sets the lowest bandwidth allocation for a client, which is the initial allocation for unconfigured clients. Specifying a large value combined with a large number of clients can lead to oversubscription. If the argument is not specified, it defaults to **32768**. The minimum value is 32768. The maximum value is 1048576.

NOTE: Not intended for general use. Only use when recommended by Quantum Support.

--sgcapacity *sgcapacity*
Set the default stripe group bandwidth capacity. If no specific configuration of bandwidth capacity is specified for a stripe group, this value will be used. This applies if --all_sg is set, or if a stripe group is configured without a capacity specified. This option is required with the use of **--all_sg**. If not specified for --add_sg, the value specified in the general options is the default. If there is no value specified in the general options, there is no default. The minimum value is 104857600.

--reserved_fc *reserved_fc*
Set the default stripe group reserved value for bandwidth for the first come class. If no specific configuration of reserved first come bandwidth is specified for a stripe group, this value will be used. This value applies if --all_sg is set, or if a stripe group is configured without a reserved_fc value specified. If not specified for --add_sg, the value specified in the general options is the default. If there is no value specified in the general options, the default is **0**.

--reserved_fs *reserved_fs*
Set the default stripe group reserved value for bandwidth for the fair share class. If no specific configuration of reserved fair share bandwidth is specified for a stripe group, this value will be used. This value applies if --all_sg is set, or if a stripe group is configured without a reserved_fs value specified. If not specified for --add_sg, the value specified in the general options is the default. If there is no value specified in the general options, the default is **0**.

--reserved_low *reserved_low*
Set the default stripe group reserved value for bandwidth for the low class. If no specific configuration of reserved low bandwidth is specified for a stripe group, this value will be used. This value applies if --all_sg is set, or if a stripe group is configured without a reserved_low value specified. If not specified for --add_sg, the value specified in the general options is the default. If there is no value specified in the general options, the default is **0**.

--reserved_mover *reserved_mover*
Set the default stripe group reserved value for bandwidth for the mover class. If no specific configuration of reserved mover bandwidth is specified for a stripe group, this value will be used. This value applies if --all_sg is set, or if a stripe group is configured without a reserved_mover value specified. If not specified for --add_sg, the value specified in the general options is the default. If there is no value specified in the general options, the default is **0**.

--min_bw *min_bw*
Set the default minimum bandwidth given to a client. If specified for the --addclient command, this value is used. If specified for the --addsg command, the value is used for a client which has no specific value. If specified for the --new or --modify commands, the value is used only when there is no specific client value and no specific stripe group value. It is recommended that both --min_bw and --max_bw are explicitly specified together. Failing to do so results in the default value being used. The minimum value and default value is **1048576**.

--max_bw *max_bw*
Set the default maximum bandwidth given to a client. If specified for the --addclient command, this value is used. If specified for the --addsg command, the value is used for a client which has no specific value. If specified for the --new or --modify commands, the value is used only when there is no specific client value and no specific stripe group value. The minimum value and default is **1048576**, but will not be less than the minimum bandwidth. It is recommended that both --min_bw and --max_bw are explicitly specified together. Failing to do so results in the default value being used. The minimum value and default value is **1048576**.

--activate *activate*
Specify the action a client must take to initiate bandwidth allocation. If "mount" is specified, the client will be allocated bandwidth at mount time. If "io" is specified, a minimum amount of bandwidth is allocated at mount time, and the first I/O request above the minimum will initiate bandwidth allocation for this client. The default value is **io**.

--print Prints the contents of the configuration file with validation.

--configfile
Specifies the name of the configuration file. When retrieving the configuration file, it specifies the file name for the output. When replacing the configuration file, it specifies the input file name.

--nofsm
Not recommended. Allows command not requiring an active fsm to run without the fsm, which prevents some validation. Some commands require the file system be running to execute the command, and don't support this option.

--timeout *seconds*
Set the amount of time to wait for the FSM to respond to a query. This value can also be specified in *snfs_rest_config.json* for all instances of this command. The default is 10 seconds.

--verbose
Verbose mode. It is used with the **-h** help option.

--yes|**-y** Executes command without operator intervention.

EXAMPLES

Create a Qos bandwidth management configuration file with the general parameters set to default values except the all_sg parameter, which is set to true.

```
rock # qbmanage --new --fsname snfs1 --all_sg --sgcapacity 1G
```

Add stripe group sg1 to the configuration of file system snfs1 with a bandwidth capacity of 500 MB/sec.

```
rock # qbmanage --addsg --fsname snfs1 --sgname sg1 --sgcapacity 500M
```

Add client 190.160.25.100 to the configuration of file system snfs1 for stripe group sg1.

```
rock # qbmanage --addclient --clientname 190.160.25.100 -f snfs1 --sgname sg1 --min_
```

Get Qos bandwidth management configuration for file system snfs1.

```
rock # qbmanage --getconfig -f snfs1 --configfile current_file
```

Update Qos bandwidth management configuration for file system snfs1.

```
rock # qbmanage --putconfig -f snfs1  --configfile new_file
```

Validate and print the QBM configuration file.

```
rock # qbmanage --validate -f snfs1  --print
```

SEE ALSO

cvadmin(8), **qbm.conf**(5), **snfs_rest_config.json**(4)

NAME

qos − StorNext File System FSM QoS Central Config File

SYNOPSIS

/usr/cvfs/config/*_rvio.opt

DESCRIPTION

The *StorNext File System* (*SNFS*) **qos** config file defines the bandwidth reservation for non-rtio (rvio) on certain clients from a central place, i.e. the fsm server. The allocated rvio bandwidth is used for non-rtio IOs on the client. The config file has the name of the designated file system's name appended by "_rvio.opt". So if the file system is "default", the QoS config file will be "default_rvio.opt". If this file does not exist, there will be no rvio bandwidth reservation for any client.

non-rtio reservation (rvio) provides a second priority bandwidth reservation mechanism for certain clients, while rtio request has the highest priority and will be satisfied first. All rvio requests share the remaining bandwidth after rtio requests are satisfied. If all rvio requests cannot be satisfied, then each rvio client is allocated the bandwidth directly proportional to its request amount. With dynamic changes of rtio requests, the allocated rvio bandwidths for clients are dynamically adjusted.

SYNTAX

A QoS central config file has multiple lines, each line defines the rvio reservation for a client. If a client has multiple IP addresses, the rvio reservation should be defined for each ip address.

The format of an line in QoS central config file is:

```
<host> <bw-type> <sg=yy>[,sg=yy]
```

<**host**> is the client host name. This can be the IP address (either V4 or V6), host name or FQDN of the client. Note: the host name should be able to be resolved (converted to a valid IP address) by the fsm server.

The <**bw-type**> is the type of bandwidth to be specified. Two types exists:

```
qosios            the bandwidth unit is IOs per second
qosmb       the bandwidth unit is mega bytes per second
```

If "qosios" is used, you may also append multiplier suffix to the amount of bandwidth:

```
Suffix    Name    Multiplier
------    ----    ----------
  K       kilo    1,024
  M       mega    1,048,576
```

The <**sg=yy**> defines the bandwidth on the designated stripe group. **sg** is the designated stripe group, **yy** is the reserved bandwidth. You can only reserve bandwidth on stripe groups whose QoS parameters have been configured in the file system config file. There are three ways to specify a stripe group:

```
*       the wildcard for stripe group, applies to all stripe group
sgname  the name of a stripe group
sgnum   the number of a stripe group, stripe group starts from 0.
```

Lines that contain white space only or that contain the comment token as the first non-white space character are ignored. An example on how to configure QoS central config file can be found:

/usr/cvfs/examples/rvio.example

FILES

*/usr/cvfs/config/*_rvio.opt*

SEE ALSO

cvfs(8), **snfs_config**(5), **fsm**(8), **mount**(8)

NAME

qustat − StorNext Statistics Utility

SYNOPSIS

qustat *command* [*Object_Identifiers*] [*Formatting*]

Manage and View StorNext Statistics

Commands using a group or module name will require the group, module or both names.

Print:	**qustat [-g** *group_name*] [**-m** *module_name*] [**-h** *host*] [**-t** *tbl*] [**-F** *opt*]
Print Hourly:	**qustat -T** [**-g** *group_name*] [**-m** *module_name*] [**-h** *host*] [**-t** *tbl*] [**-F** *opt*]
Print CSV File:	**qustat -c** *csv_file*
Print + Reset:	**qustat -P -R** [**-g** *group_name*] [**-m** *module_name*] [*Print_Options*]
Reset:	**qustat -R** [**-g** *group_name*] [**-m** *module_name*]
Reset Hourly Min/Max:	**qustat -M** [**-g** *group_name*] [**-m** *module_name*]
Interval:	**qustat -I** *interval*
Version:	**qustat -V**
Description:	**qustat** [**-g** *group_name*] [**-m** *module_name*] [**-h** *host*] [**-t** *tbl*] **-D** *Search_String*
Help:	**qustat -H**

Commands

-P, --print	Print tables (default command)
-T, --time_hourly	Print hourly time tables
-I, --interval	Set collection interval
-A, --archive	Forward stats to central archive
-R, --reset	Reset table or group if no table
-M, --reset min/max	Reset hourly minimum and maximum for table or group if no table
-S, --selftest	Run Self Test
-V, --version	Print qustat version information
-H, --help	Print qustat help information
-D, --description *str*	Print table or stat description

Object Identifiers

-c, --csv *file*	Specify a .csv file to load
-h, --host *opt*	Host name or IP address
-m, --module *opt*	Module name
-g, --group *opt*	Group name
-t, --table *opt*	Table Number
-f, --fs *opt*	File system name (same as -g)

Statistics are identified by a group and module name. Example group,module identifiers are *FS_name*,FSM; kernel,client; FSMPM,FSMPM. The default module name is FSM. If the module name is unique, the group name may be optional.

Formatting

-F, --format all	Show all records (including zeros)
-F, --format csv	Output in .csv format
-F, --format xml	Output in XML format (with -D only)
-F, --format graphite	Output graphite format data
-F, --format protobuf	Output protobuf format data (with -A only)

COMMANDS

-P, --print

This is the default command if no other commands are specified. To use the print command, you must also supply a group (file system) name with the **-g** option.

The **print** command fetches the statistics table(s) from the specified **group** (e.g. the FSM) and prints them to standard output. The output is described below.

-T, --time_hourly
Hourly statistics are kept and written to CSV files. This prints the hourly statistics. To use the print command, you must also supply a group (file system) name with the **-g** option.

-I, --interval
The **interval** command controls the rate at which statistics are forwarded from running processes to the snstatd process. The interval parameter is in seconds. The value must be -1 (turn off) or between 5 and 2147483 inclusive.

-A, --archive
The **archive** command specifies the location that the snstatd process should periodically send statistics to. The target is in terms of a hostname and tcp port number. A TCP/IP connection is opened ever **interval** seconds and any new statistics are send in graphite format down the connection. The connection is then closed. Use -F with this command to switch between graphite and protobuf output.

-R, --reset

WARNING! Resetting statistics affects hourly reports and anybody
else using qustats, including connect. Note: Using the -R and -P option together is not atomic. First the current values are collected and then the reset command is processed. Any statistics updated between the collection and reset are lost.

This command resets the internal statistics tables for the specified **Object Identifier**.

-M, --minmax

WARNING! Resetting minimum and maximum affects the hourly report for that time period.

This command resets only the hourly minimum and maximum values for the specified **Object Identifier**.

-S, --selftest
Runs self-test to verify that internal functions are working properly. If your system is busy, it may be normal for the timing tests to fail.

-V, --version
Display version information for the qustat command.

-D, --description
Use the -D command to find the description for tables and stats. The -D search string may contain the wildcard '*'. Note that strings with an asterisk should be quoted to avoid globbing. You must provide object identifiers for at least one table.

Search Examples:

```
qustat -g myFileSys -D "*"
qustat -g myFileSys -D "VOP Lookup"
qustat -g myFileSys -t 1 -D "VOP Lookup"
qustat -g myFileSys -t 1 -D "VOP *"
qustat -g myFileSys -t 1 -D "*"
```

-H, --help
Displays help information.

OBJECT IDENTIFIERS

-c *csv_file*, **--csv** *csv_file*
Specify a .csv file to load.

-h *host*, **--host** *host*
The host or IP address where the **Module** (e.g. FSM) is located. This option is normally not needed when displaying FSM statistics if your computer is joined to the cluster.

-m *module*, **--module** *module*
Specifies the module (process or service) from which to extract statistics. The module specifier **all** can be used to select all types of modules.

The default module is **FSM** if not specified.

-g *group*, **--group** *group*
Identifies the group of tables. For **FSM** modules, the **group** specifies the file system name.

This is the same as the **-f** option.

-t *table_number*, **--table** *table_number*
Use **-t** to print a single table. If you print all statistics, the table numbers are displayed in the table header. See **TABLES** below.

If you do not specify a table, all tables for the group are displayed.

-f *file_system*, **--fs** *file_system*
This is the same as the **-g** option.

FORMATTING

-F *option*, **--format** *option*
You may specify the **-F** option multiple times. The **all** option will print all statistics including those with all zero values. The **csv** option will display output in comma-separated-values format. The **xml** option will display output in XML format.

OUTPUT

The Group header includes revision, host, module, group and time recorded. The time_t value is the output of the **time**(2) function call.

Columns include:

NAME	The name of the statistic being gathered.
TYP	The type of statistic.
CNT	The number of times something occurred.
LVL	The current *level* of something (e.g. number of free buffers).
SUM	The accumulated sum such as the amount of data written.
TIM	The amount of time consumed (in microseconds).
COUNT	Number of times the operation was performed.
MIN/MAX	Minimum and maximum values.
TOT/LVL	Total or current level (depending on TYP)
AVG	The average (TOT divided by COUNT)

TABLES

A **Group** is normally split into multiple tables. Each table is identified by a unique *table number*.

Table numbers are guaranteed to identify a single unique object throughout the lifespan of the given group, but not across reboots/restarts for the group.

For the FSM, the main table numbers will remain consistent across restarts, but the per-client statistics will vary depending on connection order.

ACCURACY

Statistics are not guaranteed to be 100% accurate.

For performance reasons, the implementation does not explicitly lock the code when gathering statistics. However most operations are already protected by other multi-thread locking and therefore inaccuracies should be minimal.

In addition, operations are not halted or locked when resetting or gathering statistics. It is possible to have a statistic dropped during a reset or snap of statistics.

CVLOG HOURLY STATISTICS DUMPS

In prior releases, the FSM dumped statistics on an hourly basis to the cvlogs which were located under the data directory. These statistics dumps have been moved to a separate directory called **qustats**. The format has also been changed to .csv files which can be opened in most spreadsheets and databases. They can also be easily parsed by most programming languages and utilities.

EXAMPLES

Print the FSM statistics for file system snfs1.

```
rock # qustat -g snfs1
```

Print the kernel client statistics.

```
rock # qustat -g kernel -m client
```

Print the statistic for the client table 2 statistics.

```
rock # qustat -m client -t 2
```

Print the statistic descriptions for client definitions.

```
rock # qustat -g kernel -m client -D "*"
```

Print the statistic descriptions for client definitions.

```
rock # qustat -g FSMPM -m FSMPM -D "*"
```

Print the statistic descriptions for the fsm VOP statistics

```
rock # qustat -g snfs1 -D "VOP"
```

NAME

qustat.conf − StorNext snstatd configuration file

SYNOPSIS

/usr/cvfs/config/qustat.conf

DESCRIPTION

The *StorNext File System* (*SNFS*) **qustat.conf** file provides a way to configure snstatd as well as providing debugging levels for qustat commands. Refer to this man page's **EXAMPLE** section for a visual representation of the described format. If the file is not present at snstatd start time, a file will be written with default values.

The default log level is Inf(informational). The default archive age is 31536000 seconds (1 year). The default archive rate is 3600 seconds (1 hour).

WARNING: It is highly recommended that Quantum Technical Support be contacted before changing the default configuration. In normal usage, the default values are adequate and this file should not be changed.

EXAMPLE

The following is an example of the **qustat.conf** file.

```
# Maximum age of .csv files in seconds or 'infinite'
archive_age_max_secs=31536000
# Rate to dump .csv files in seconds or 'off'
archive_dump_rate_secs=3600
# Debug logging levels (0=Crit, 1=Err, 2=Warn, 3=Inf, 4-6=Dbg
dbg_default=3
# Debug logging levels (0=Crit, 1=Err, 2=Warn, 3=Inf, 4-6=Dbg
dbg_api=3
# Debug logging levels (0=Crit, 1=Err, 2=Warn, 3=Inf, 4-6=Dbg
dbg_protobufs=3
# Debug logging levels (0=Crit, 1=Err, 2=Warn, 3=Inf, 4-6=Dbg
dbg_net=3
# Debug logging levels (0=Crit, 1=Err, 2=Warn, 3=Inf, 4-6=Dbg
dbg_help_hash=3
```

FILES

/usr/cvfs/config/qustat.conf
/usr/cvfs/debug/snstatd.log
/usr/cvfs/debug/qustat_cmd.log
/usr/cvfs/debug/qustat_lib.log

SEE ALSO

snstatd(8)

NAME

sgadd − Utility to Add a Stripe Group to a File System

SYNOPSIS

sgadd --fsname|**-f** *FsName* [**--sb** *stripebreadth*] **--disks**|**-u** *diskunit,....,diskunit*
[**--template**|**-g** *TemplateSgName* | *'#TemplateSgIndex'*]
[**--affinity**|**-a** *affinity_name,...,affinity_name*] [**--sgname**|**-g** *SgName*] [**--rtios** *rt_ios_per_sec*]
[**--rtiosreserve** *non_rt_ios_per_sec*] [**--rtmb** *rt_data_rate*] [**--rtmbreserve** *non_rt_data_rate*]
[**--excl[**=true|false|1|0**]]** [**--read[**=true|false|1|0**]]** [**--write[**=true|false|1|0**]]** [**--alloc[**=true|false|1|0**]]**
[**--data[**=true|false|1|0**]]** [**--metadata[**=true|false|1|0**]]** [**--timeout** *seconds*] [**--yes**|**-y**] [**--verbose**]
[**--debug**|**-D**] [**--help**] [**--dryrun**] [**--diskcheck**] [**--bypass_trim**]

sgmanage --add ...

DESCRIPTION

sgadd can be used to create and configure a stripe group for a StorNext File System. The **--fsnamef** *FsName*, **--disks** and *diskunit,...,diskunit* options are required. The **--sb** *stripebreadth* option is required if the **--template** option is not specified. If no stripe group name is given, the utility creates the name.

If a vacant stripe group is present, it will be used, otherwise one will be appended to the existing stripe groups. If a vacant stripe group is used and there are file system clients running a version of StorNext prior to 5.4.0, they should un-mount the file system prior to the add and re-mount after.

The use of **sgadd** requires that the file system be active on the primary node of an HA pair, if HA is configured. If it is active on the secondary, **cvadmin** subcommand **fail** can be used to switch it to the primary.

The **sgadd**(8) command may also be invoked as:

sgmanage --add ...

OPTIONS

Options that start with -- and take an argument can have the argument separated by either a space or an equal sign, e.g.

--fsname *FsName*
--fsname=*FsName*

are equivalent.

--fsname|**-f** *FsName*

Selects a given file system. If the file system is not running in the default cluster or administrative domain, those may added to file system name using the syntax:

FsName[**@***cluster*[*/addom*]]

--sgname|**-g** *SgName*

Use this name for the new stripe group.

--sb *stripebreadth*

Set stripebreadth value in bytes. If **K ,M ,G** is appended to the value, then the value is in KByte, Mbyte or Gbyte units.

--disks|**-u** *diskunit,...,diskunit*

List of disks to add to stripe group. Delimited by ",", disk unit names must be seen by the **cvlabel**(8) command.

--template|**-g** *TemplateSgName* | *'#TemplateSgIndex'*

The template stripe group is used for setting up the new stripe group. Any of the values from the template stripe group can be explicitly overridden by specifying them on the command line. If the template stripe group contains the journal, the new stripe group will not inherit this setting. The template stripe group may be specified by numerical index using the hash sign and index number. The hash sign must be escaped to avoid special processing by the shell.

--affinity|**-a** *affinity_name,...,affinity_name*

List of affinities to add to stripe group, delimited by ",". If this option is specified several times, they will be combined into a single list.

--rtios *rt_ios_per_sec*
Set QOS realtime IOs/sec.

--rtmb *rt_data_rate_per_sec*
Set QOS realtime data rate per sec in MBs.

--rtiosreserve *non_rt_ios_per_sec*
Set QOS non realtime IOs/sec.

--rtmbreserve *non_rt_data_rate_per_sec*
Set QOS non realtime date rate per sec in MBs.

--rttokentimeout *rt_timeout*
Set QOS realtime timeout value.

--excl[=**true**|**false**|**1**|**0**]
Specify whether or not the new stripe group affinities should be exclusive. If the argument is not specified, it defaults to **false**.

--read[=**true**|**false**|**1**|**0**]
Specify whether or not reads should be enabled. If the argument is not specified, it defaults to **true**.

--write[=**true**|**false**|**1**|**0**]
Specify whether or not writes should be enabled. If the argument is not specified, it defaults to **true**.

--alloc[=**true**|**false**|**1**|**0**]
Specify whether or not allocations should be initially allowed. If the argument is not specified, it defaults to **true**.

--data[=**true**|**false**|**1**|**0**]
Specify whether or not the stripe group will allow user data allocations. If neither **--data** nor **--metadata** is specified, **--data** defaults to **true**. If **--metadata** is true, **--data** defaults to **false**.

--metadata[=**true**|**false**|**1**|**0**]
Specify whether or not the stripe group will allow metadata allocations. The default is **false**.

--timeout *seconds*
Set the amount of time to wait for the FSM to respond to a query. This value can also be specified in *snfs_rest_config.json* for all instances of this command using the process name entry **sgmanage** in *snfs_rest_config.json*. This file is installed by default and the **sgmanage** timeout has a default value of 60 seconds.

--yes|**-y** Run the command without prompting for confirmation before making the change.

--verbose
Show additional output.

--debug|**-D**
Turn on debugging code

--help Print a usage statement.

--dryrun
Check for warnings and return an error if a warning is present. Return success if no warnings are present. No other action is taken.

--diskcheck
Do a check for disks that are included in the stripe group that is being added to see if they are currently in use in another file system that is visible to the cluster. In some configurations, this may take a long time. If there are disks in use, the operation is aborted.

--bypass_trim

By default, **sgadd** will check to see if any of the LUNs in the stripe group being added are thin provisioned. If they are, an unmap operation will be attempted to clear all mappings. This operation can take several minutes and can not be undone. If this behavior is not desired, the unmap operation can be bypassed by specifying this option. Note: for data only stripe groups, a trim operation can be done on-line at any time with **sgmanage --trim**.

STRIPE GROUP CONFIGURATION

Quantum recommends that all file systems be configured to run with separate stripe groups for metadata and user data. In this configuration, **sgadd** can be used to add either metadata or user data stripe groups to any StorNext file system, including the HA shared file system.

With file systems that are configured to run with "mixed" stripe groups that contain both metadata and user data it is recommended that **sgadd** be used to add only mixed stripes groups to the file system. Adding exclusive user data-only stripes groups to file systems configured with mixed stripe groups is not recommended. However, with the exception of the HA shared file system, a user data-only stripe group can be added to the file system if the original mixed stripe group is converted into a metadata-only stripe group using **sgoffload**. Note that sgoffload is not available to be used on the HA shared file system. See the **sgoffload** man page for additional information.

NOTE: The sgadd command is not available on Debian and Ubuntu clients.

EXAMPLES

Add a user data stripe group with 3 disks, 256K stripebreadth

```
rock # sgadd -f snfs1 -u disk_0001,disk_0002,disk_0003 --sb 256K
```

Add a metadata stripe group with 3 disks, 256K stripebreadth

```
rock # sgadd -f snfs1 -u disk_0001,disk_0002,disk_0003 --sb 256K --data=false --r
```

SEE ALSO

cvadmin(8), **cvlabel**(8), **sgdefrag**(8), **sgmanage**(8), **sgoffload**(8), **sgresize**(8)

NAME

sgdefrag − Defrag the free space in a Stripe Group

SYNOPSIS

sgdefrag --fsname|-f *FsName* [**--sgname|-g** *SgName* | *'#SgIndex'*] [**--alloc_at_end[=**true|false|1|0**]**
[**--blocks** *nblocks*] [**--keep**] [**--ignore_affinity**] [**--minspace|-M**] [**--yes|-y**] [**--verbose**]
[**--debug|-D**] [**--progress** *ProgressFilePath*] [**--no_admin_lock**] [**--timeout** *seconds*]
[**--otime <[yyyy-mm-dd:]hh:mm:ss>**] [**--list_frag_count**] [**--help**]

sgmanage --defrag ...

DESCRIPTION

sgdefrag reduces the amount of free space fragmentation in a stripe group through a process called defragmentation. For each file that contains extents on the stripe group, the list of all extents for that file on this stripe group is retrieved from the metadata archive database. These extents are then re-allocated in the file system across all stripe groups or on the same stripe group (see the **--keep** option).

When re-allocating space, the new blocks are by default allocated at the end of a stripe group. This behavior can be changed to use the allocater's default algorithms for extent placement. This may be at the start of the stripe group but can be influenced by other factors such as best fit or allocation session reservation. If the only available space is at the end of the stripe group, the allocator will place the new extents here. The placement of file extents affects the performance of writes and reads on traditional spinning disks. Since the speed of the disk is greater on the outside tracks, writes and reads to and from blocks allocated here will outperform writes and reads to and from blocks on the innermost tracks. In some cases, it may be useful to use **sgdefrag** to free up higher performance blocks taken by older files and leave the new free space available for newly created files. There is no performance difference when a Solid State Device (SSD) is used to hold file data on a stripe group.

Defragging creates fewer larger blocks of free space and often results in fewer larger data extents on the files it operates on. This generally results in better performance for the file system. See also the **snfsdefrag**(1) command which defrags files instead of stripe groups. Also see the **cvfsck**(8) command for information regarding the current state of free space fragmentation in a stripe group.

The use of **sgdefrag** requires that the file system be active on the primary node of an HA pair, if HA is configured. If it is active on the secondary, **cvadmin** subcommand **fail** can be used to switch it to the primary.

The use of **sgdefrag** also requires that the global configuration variable metadataArchive be set to true. If this change needs to be made, the file system must be stopped and restarted for the change to take effect. Wait for the metadata archive database rebuild to complete by monitoring the status with the **cvadmin** subcommand **mdarchive status**.

Sgdefrag will also attempt to defrag files as well as free space by combining adjacent source extents from a file on the source stripe group into a single allocation request on the target. This behavior is the default, but will not be used if the **minspace** option is used. In that case, each source extent will result in a separate call to the allocator. Note also that when **snfsdefrag**(1) is used to defragment files, it will consolidate all contiguous source extents from all stripe groups.

Sgdefrag makes use of the adminstrative lock feature of StorNext which allows it to move files that are currently open from a client node. It does this by revoking the I/O token from that client. During the time the token is revoked, I/O operations are suspended on that file. This behavior can be overridden with the **--no_admin_lock** option. Some clients may not support the I/O token operation if the software version is older. In this case, the file is seen to be busy by **sgdefrag** and the file is skipped. To completely defrag a stripe group, **sgdefrag** may have to be invoked repeatedly to move the once busy files.

Multiple **sgdefrag** commands may be invoked simultaneously as long as the source stripe group is not the same.

The **sgdefrag**(8) command may also be invoked as:

sgmanage --defrag ...

OPTIONS

Options that start with -- and take an argument can have the argument separated by either a space or an equal sign, e.g.

--fsname *FsName*
--fsname=*FsName*

are equivalent.

--fsname|**-f** *FsName*
Selects a given file system. If the file system is not running in the default cluster or administrative domain, those may added to file system name using the syntax:
FsName[**@***cluster*[*/addom*]]

--sgname|**-g** *SgName | '#SgIndex'*
Name or index of the stripegroup to defragment. The hash sign, which is part of the index syntax, must be escaped to avoid special processing by the shell.

--alloc_at_end[= true | false | 1 | 0]
Specify if new allocations are to be made at the end of the stripe group. The default is that new allocations are made at the end of the stripe group. If specified alone, the value of **true** is used.

--blocks *nblocks*
Specify the number of blocks to move. If *nblocks* is **0**, 1/2 of the used blocks will be moved, which is the default behavior. If *nblocks* is **-1**, all used blocks will be moved. Otherwise, *nblocks* **will be moved.**

--keep Keep the allocations on the same stripe group.

--ignore_affinity
Ignore affinities and allow data to be allocated anywhere. The **--keep** and **--ignore_affinity** options are mutually exclusive.

--minspace|**-M**
Allocate the smallest possible pieces.

--yes|**-y** Run the command without prompting for confirmation before making the change.

--verbose
Show additional output.

--debug|**-D**
Turn on debugging code

--progress *ProgressFilePath*
Create and periodically update this file with information about how far along we are and how much time remains. This file is in JSON format. Progress information is also written to standard output in normal (not JSON) format.

--no_admin_lock
Do not attempt to temporarily stop I/O related operations on an open file by setting the administrative lock on a file. The default behavior is to attempt to set the administrative lock on each file that is being defragged. If **sgdefrag** successfully sets the administrative lock, I/O related operations will be suspended for any application that has the file open during the time **sgdefrag** is operating on that file. If a file is open and the administrative lock cannot be obtained, that file will be skipped.

--timeout *seconds*
Set the amount of time to wait for the FSM to respond to a query. This value can also be specified in *snfs_rest_config.json* for all instances of this command using the process name entry **sgmanage** in *snfs_rest_config.json*. This file is installed by default and the **sgmanage** timeout has a default value of 60 seconds.

--otime *[yyyy-mm-dd:]hh:mm:ss*
Only operate on files that are older than the specified time, as observed in the atime or mtime field for the file. Note here that it is possible for a client to modify the updating of the atime for a file using mount options **noatime** or **atimedelay**. This could cause a file to be defragged even if it has been recently accessed.

--list_frag_count
List the free space fragment count only, do not attempt the defrag operation.

--help Print a usage statement.

NOTE: The sgdefrag command is not available on Debian and Ubuntu clients.

EXAMPLES

SEE ALSO

sgadd(8), **sgmanage**(8), **sgoffload**(8), **sgresize**(8), **cvadmin**(8), **cvfsck**(8), **snfsdefrag**(1), **snfs_config**(5), **mount_cvfs**(8), **snfs_rest_config.json**(4)

NAME

sgmanage − Stripe Group Management Utility

SYNOPSIS

sgmanage *command* [**--sgname**|**-g** *SgName* | *'#SgIndex'*] [**--fsname**|**-f** *FsName*] [**--yes**|**-y**] [**--verbose**] [**--debug**|**-D**] [**--help**] [**--timeout** *seconds*]

sgmanage --list|**-l** *common_options*

sgmanage --list_files [--list_output <list_file_path>] *common_options*

sgmanage --suspend|**-s** *common_options*

sgmanage --resume|**-r** *common_options*

sgmanage --delete|**-d** *common_options*

sgmanage --trim *common_options*

sgmanage --multipath|**-m** *method common_options*

sgmanage --offload ...
sgoffload ...

sgmanage --defrag ...
sgdefrag ...

sgmanage --add ...
sgadd ...

sgmanage --resize
sgresize ...

DESCRIPTION

sgmanage and related stripe group management utilities allow an administrator to perform various tasks related to stripe groups while the file system is active and in use by clients and their applications. See the **snfs_config**(5) man page for the definition of a stripe group.

In its basic form, **sgmanage** can be used to list information about existing stripe groups in a StorNext File System. If no file system name is given, the utility lists the information for all the file systems it can see. The utility can also be used to control the allocation state of a stripe group by enabling or disabling space allocation. In addition, the multipath method of the stripe group may be modified. When using thin-provisioned storage, the size of the LUNS in a stripe group may be increased. Finally, a stripe group can be deleted, which makes it vacant and available for re-use. This can be especially useful for file systems which have stripe groups that have been downed. Prior to StorNext 6.0, this was the only way to retire obsolete storage. The downed stripe group must be empty.

The use of **sgmanage** requires that the file system be active on the primary node of an HA pair, if HA is configured. If it is active on the secondary, **cvadmin** subcommand **fail** can be used to switch it to the primary.

The offload, defrag and delete functions require that the global configuration variable metadataArchive be set to true. If this change needs to be made, the file system must be stopped and restarted for the change to take effect. Wait for the metadata archive database rebuild to complete by monitoring the status with the **cvadmin** subcommand **mdarchive status**.

See **sgoffload**(8) for usage of **sgmanage --offload**.
See **sgdefrag**(8) for usage of **sgmanage --defrag**.
See **sgadd**(8) for usage of **sgmanage --add**.
See **sgresize**(8) for usage of **sgmanage --resize**.

OPTIONS

Options that start with **--** can be abbreviated to just their unique prefix; however when called from scripts the full option name should be used to avoid future abbreviation conflicts if new options are added.

Options that start with -- and take an argument can have the argument separated by either a space or an

equal sign, e.g.

--fsname *FsName*
--fsname=*FsName*

are equivalent.

--fsname|**-f** *FsName*

Selects a given file system. If the file system is not running in the default cluster or administrative domain, those may added to file system name using the syntax:

FsName[**@***cluster*[/*addom*]]

--sgname|**-g** *SgName* | *'#SgIndex'*

Selects a stripe group by name or index. The hash sign, which is part of the index syntax, must be escaped to avoid special processing by the shell.

--multipath|**-m** *method*

Change the multipath method of the stripe group. *Method* **can be rotate**, **static**, **sticky**, **balance** or **cycle**.

StorNext has the capability of utilizing multiple paths from a system to the SAN disks.

This capability is referred to as "multi-pathing", or sometimes "multi-HBA support". (HBA := Host Based Adaptor).

At "disk discovery" time, for each physical path (HBA), a scan of all of the SAN disks visible to that path is initiated, accumulating information such as the SNFS label, and where possible, the disk (or LUN) serial number.

At mount time, the visible set of StorNext labeled disks is matched against the requested disks for the file system to be mounted.

If the requested disk label appears more than once, then a "multi-path" table entry is built for each available path.

If the disk (or LUN) device is capable of returning a serial number, then that serial number is used to further verify that all of the paths to that StorNext labeled device share the same serial number.

If the disk (or LUN) device is not capable of returning a serial number then the device will be used, but StorNext will not be able to discern the difference between a multi-path accessible device, and two or more unique devices that have been assigned duplicate StorNext labels.

The presence of serial numbers can be validated by using the "cvlabel -ls" command. The "-s" option requests the displaying of the serial number along with the normal label information.

There are five modes of multi-path usage which can also be specified in the filesystem config file. In cases where there are multiple paths and an error has been detected, the algorithm falls back to the **rotate** method. The **balance** and **cycle** methods will provide the best aggregate throughput for a cluster of hosts sharing storage.

balance

The **balance** mode provides load balancing across all the available, active, paths to a device. At I/O submission time, the least used HBA/controller port combination is used as the preferred path. All StorNext File System I/O in progress at the time is taken into account.

cycle The **cycle** mode rotates I/O to a LUN across all the available, active, paths to it. As each new I/O is submitted, the next path is selected.

rotate The **rotate** mode is the default for configurations where the operating system presents multiple paths to a device.

In this mode, as an I/O is initiated, an HBA controller pair to use for this I/O is selected based on a load balance method calculation.

If an I/O terminates in error, a "time penalty" is assessed against that path, and another "Active" path is used. If there are not any "Active" paths that are not already in the "error penalty" state, then a search for an available "Passive" path will occur, possibly triggering an Automatic Volume Transfer to occur in the Raid Controller.

static The "default" mode for all disks other than Dual Raid controller configurations that are operating in Active/Active mode with AVT enabled.

As disks (or LUNs) are recognized at mount time, they are statically associated with an HBA in rotation.

i.e. given 2 HBA's, and for disks/LUNs:

```
disk 0 -> HBA 0
disk 1 -> HBA 1
disk 2 -> HBA 0
disk 3 -> HBA 1

and so on...
```

sticky In this mode, the path to use for an I/O is based on the identity of the target file. This mode will better utilize the controller cache, but will not take advantage of multiple paths for a single file.

The current mode employed by a stripe group can be viewed via the "cvadmin" command "show long".

Permanent modifications may be made by incorporating a "MultiPathMethod" configuration statement in the configuration file for a stripe group.

In the case of an I/O error, that HBA is assessed an "error penalty", and will not be used for a period of time, after which another attempt to use it will occur.

The first "hard" failure of an HBA often results in a fairly long time-out period (anywhere from 30 seconds to a couple of minutes).

With most HBA's, once a "hard" failure (e.g. unplugged cable) has been recognized, the HBA immediately returns failure status without a time-out, minimizing the impact of attempting to re-use the HBA periodically after a failure. If the link is restored, most HBA's will return to operational state on the next I/O request.

--list|-l List the stripe group information including status and disk nodes in the stripe group. If the **--fsname** option in not present, it will list the information for all stripe groups in all file systems. Information for a specific stripe group is displayed if **--sgname** is given. If the **--verbose** is also selected, a longer version of stats is displayed.

--list_files
List the file paths of files which have at least one extent on the selected stripe group. Information is written to standard output, unless the **--list_output** option is specified.

--list_output <list_file_path>
Used with **--list_files**, specifies the file which will contain the list of file paths.

--suspend|**--stop**|**-s**
Suspend any new space allocations on a given data stripe group. The **--fsname** *FsName* and **--sgname** *SgName | '#SgIndex'* options must be included. While writes will be allocated on other data stripe groups, reads can continue on the select stripe group. If there are no other data stripe groups then ENOSPC (no space) is returned. In-use space will reflect the loss of allocatable blocks from the selected stripe group.

--resume|**-r**
Resume new space allocations on a given data stripe group. The **--fsname** *FsName* and **--sgname** *SgName | '#SgIndex'* options must be included. Writes will now be allocated on the selected data stripe group. In-use percentage of space now is decreased as blocks are made available for allocation.

--delete|**-d**
Delete a data stripe group. This can only be done if there is no user data on the stripe group. For a user data only stripe group, the disk luns are actually removed from the configuration and the stripe group is marked **VACANT**. For a shared metadata/user data stripe group, the stripe group is marked **EXCLUSIVE** and can used only for metadata allocation. File system clients prior to StorNext 5.4.0 should un-mount the file system prior to the delete and re-mount after.

--trim This option is for use with thin provisioned devices in the given file system. It causes UNMAPS or TRIM operations for all free space in the given stripe group. This performs the same operations as the **cvfsck -U** operation, except that with **sgmanage**, the file system stays on line. During the trim operation, allocations to the given stripe group are suspended, so another data stripe group must be available for file system operations to continue. This option currently only works with Quantum QXS series storage.

--verbose
Verbose mode. More stats are displayed when used with the **-l** list option. Additional debugging information is also displayed on most commands.

--yes|**-y** Executes command without operator intervention.

--timeout *seconds*
Set the amount of time to wait for the FSM to respond to a query. This value can also be specified in *snfs_rest_config.json* for all instances of this command using the process name entry **sgmanage** in *snfs_rest_config.json*. This file is installed by default and the **sgmanage** timeout has a default value of 60 seconds.

EXAMPLES

List state of all file systems stripe groups

```
rock # sgmanage --list
```

List state and information for all stripe groups in file system snfs1.

```
rock # sgmanage --list -f snfs1
```

Suspend allocation on stripe group sg1, file system snfs1.

```
rock # sgmanage --suspend -f snfs1 -g sg1
```

Resume allocation on stripe group 1, file system snfs1.

```
rock # sgmanage --resume -f snfs1 -g sg1
```

NOTE: The sgmanage command is not available on Debian and Ubuntu clients.

SEE ALSO

cvadmin(8), **snfs_config**(5), **sgadd**(8), **sgdefrag**(8), **sgoffload**(8), **sgresize**(8), **snfs_rest_config.json**(4)

NAME

sgoffload − Stripe Group Offload Utility

SYNOPSIS

sgoffload --fsname|**-f** *FsName* --sgname|-g *SgName | '#SgIndex'*
[**--target** *TargetSgName | '#TargetSgIndex'*] [**--percent**|**-p** *percentage*] [**--ignore_affinity**]
[**--vacate**] [--**yes**|**-y**] [**--verbose**] [**--debug**|**-D**] [--**progress** *ProgressFilePath*] [**--no_admin_lock**]
[**--timeout** *seconds*] [**--help**]

sgmanage --offload ...

DESCRIPTION

sgoffload can be used to move existing data off of any data stripe group to another stripe group. The selected stripe group can be then marked vacated and reads/writes continue on from a different stripe group in a StorNext File System. Vacating is removing the configured disks from the stripe group. If a target stripe group, **--target** *TargetSgName | '#SgIndex'*, is selected, the file system will try allocation from only that stripe group independent of affinity. If the **--ignore_affinity** option is set, the original affinity is ignored in determining the new allocation. If the file system only has one data stripe group the command fails with **ENOSPC**.

The use of **sgoffload** requires that the file system be active on the primary node of an HA pair, if HA is configured. If it is active on the secondary, **cvadmin** subcommand **fail** can be used to switch it to the primary.

The use of **sgoffload** also requires that the global configuration variable metadataArchive be set to true. If this change needs to be made, the file system must be stopped and restarted for the change to take effect. Wait for the metadata archive database rebuild to complete by monitoring the status with the **cvadmin** subcommand **mdarchive status**.

Sgoffload will attempt to defragment files as it offloads the stripe group by combining contiguous file extents from the source stripe group into a single allocation request on the destination stripe group. Note that this is not the same behavior as **snfsdefrag**(1) which will consolidate all contiguous extents, regardless of the original stripe group.

Sgoffload makes use of the adminstrative lock feature of StorNext which allows it to move files that are currently open from a client node. While a file is being operated on, I/O operations on that file are suspended. This behavior can be overridden with the **--no_admin_lock** option. Some clients may not support the I/O token operation if the software version is older. In this case, the file is seen to be busy by **sgoffload** and the file is skipped. To completely offload a stripe group, **sgoffload** may have to be invoked repeatedly to move the once busy files.

Multiple **sgoffload** commands may be invoked simultaneously as long as the source stripe group is not the same.

The **sgoffload** command may also be invoked as:
sgmanage --offload ...

OPTIONS

Options that start with -- and take an argument can have the argument separated by either a space or an equal sign, e.g.
--fsname *FsName*
--fsname=*FsName*
are equivalent.

--fsname|**-f** *FsName*
Selects a given file system. If the file system is not running in the default cluster or administrative domain, those may added to file system name using the syntax:
FsName[**@***cluster*[*/addom*]]

--sgname|**-g** *SgName | '#SgIndex'*
Selects a stripe group by name or index. The hash sign, which is part of the index syntax, must be escaped to avoid special processing by the shell.

--target *TargetSgName | '#TargetSgIndex'*
Selects a target stripe group to move data to. The hash sign, which is part of the index syntax, must be escaped to avoid special processing by the shell.

--percent|-p *percentage*
Set percentage of blocks to move from 1 to 99 percent. The **--vacate** and **--percent** options are mutually exclusive. If the **--percent** option is not selected, all blocks are moved. The percentage of blocks might vary as blocks can be removed while the offload is in progress.

--ignore_affinity
Ignore affinities and allow data to be allocated anywhere. The **--target** and **--ignore_affinity** options are mutually exclusive.

--vacate
Vacate the stripe group once data is moved. The default is not to vacate. If the stripe group is a shared metadata/user data stripe group, the stripe group is made **EXCLUSIVE** for metadata allocations only. If the stripe group is a shared journal/user data stripe group, the stripe group is made **EXCLUSIVE** for journal only. If the stripe group is user data, the disk luns are removed from the configuration and the stripe group is marked **VACANT**. A stripe group that is marked as **DOWN** may also be vacated using this option. A vacant stripe group can be re-used using the **sgadd** functionality. File system clients running a version of StorNext prior to 5.4.0 should un-mount prior to the vacate and re-mount after. At the beginning of the vacate operation, **sgoffload** disables allocations to the stripe group, if they were not already disabled. At the end of the offload, if the operation could not complete due to busy files, allocations to the stripe group remain suspended. The **--vacate** option is the same as running the following commands:

```
sgmanage -f <fs> -g <sg> --suspend
sgoffload -f <fs> -g <sg>
sgmanage -f <fs> -g <sg> --delete
```

--yes|-y Run the command without prompting for confirmation before making the change.

--verbose
Show additional output.

--debug|-D
Prints out debug information of each allocated extent being moved.

--progress *ProgressFilePath*
Create and periodically update this file with information about how far along we are and how much time remains. This file is in JSON format. Progress information is also written to standard output in normal (not JSON) format.

--no_admin_lock
Do not attempt to temporarily stop I/O related operations on an open file by setting the administrative lock on a file. The default behavior is to attempt to set the administrative lock on each file that is being offloaded. If **sgoffload** successfully sets the administrative lock, I/O related operations will be suspended for any application that has the file open during the time **sgoffload** is operating on that file. If a file is open and the administrative lock cannot be obtained, that file will be skipped and the offload may not completely empty the stripe group.

--timeout *seconds*
Set the amount of time to wait for the FSM to respond to a query. This value can also be specified in *snfs_rest_config.json* for all instances of this command using the process name entry **sgmanage** in *snfs_rest_config.json*. This file is installed by default and the **sgmanage** timeout has a default value of 60 seconds.

--help Print a usage statement.

EXAMPLES

Offload data from stripe group sg1

```
rock# sgoffload -f snfs1 -g sg1
```

Offload data from stripe group 1 to target stripe group sg3

```
rock# sgoffload -f snfs1 -g sg1 --target sg3
```

NOTE: The sgoffload command is not available on Debian and Ubuntu clients.

SEE ALSO

cvadmin(8), **snfs_config**(5), **sgadd**(8), **sgdefrag**(8), **sgmanage**(8), **sgresize**(8), **snfsdefrag**(1), **snfs_rest_config.json**(4)

NAME

sgresize − Utility to resize a Stripe Group after extending the disks

SYNOPSIS

sgresize --fsname|**-f** *FsName* **--sgname**|**-g** *SgName* | *'#SgIndex'* [**--sectors all**|*nsectors*] [**--timeout** *seconds*] [**--force**] [**--yes**|**-y**] [**--verbose**] [**--debug**|**-D**] [**--diskcheck**] [**--help**]

sgmanage --resize ...

DESCRIPTION

The **sgresize**(8) is used to resize a stripegroup after the disks have been extended. It is assumed that all the previous data is intact, and new space as been added to the end of the disks, so all the resize operation does is to add to the free space for the stripegroup.

Before running this command, the disks have to be extended, the additional space made visible to the operating system, and the disks then have to be relabeled with the new size with the **cvlabel**(8) command. File system clients running a version of StorNext prior to 5.4.0 should un-mount prior to the resize and re-mount after.

The **--fsname** and **--sgname** options are required. When run without the **--sectors** option or with **--sectors all**, **sgresize**(8) will determine how much to extend the stripegroup based on the sizes of the disks, as labeled by **cvlabel**(8). All of the disks in the stripegroup need to be extended by the same amount at the same time.

The use of **sgresize** requires that the file system be active on the primary node of an HA pair, if HA is configured. If it is active on the secondary, **cvadmin** subcommand **fail** can be used to switch it to the primary.

The **sgresize**(8) command may also be invoked as:

sgmanage --resize ...

OPTIONS

Options that start with -- and take an argument can have the argument separated by either a space or an equal sign, e.g.

--fsname *FsName*
--fsname=*FsName*

are equivalent.

−-fsname|**-f** *FsName*
: Select a given file system. If the file system is not running in the default cluster or administrative domain, those may added to file system name using the syntax:
FsName[**@***cluster*[*/addom*]]

−-sgname|**-g** *SgName* | *'#SgIndex'*
: Name or index of the stripegroup to be resized. The hash sign, which is part of the index syntax, must be escaped to avoid special processing by the shell.

--sectors all|*sector_count*
: Expand the sector count of each disk to *sector_count*. If *sector_count* exceeds the size of any of the LUNs as labeled by *cvlabel*(8), or if *sector_count* is less than the previous configured sector count, the resize operation will fail. If *sector_count* is given as **all**, then **sgresize**(8) will determine how much to extend the stripegroup based on the sizes of the disks, as labeled by **cvlabel**(8).

--timeout *seconds*
: Set the amount of time to wait for the FSM to respond to a query. This value can also be specified in *snfs_rest_config.json* for all instances of this command using the process name entry **sgmanage** in *snfs_rest_config.json*. This file is installed by default and the **sgmanage** timeout has a default value of 60 seconds.

--force Force the resize operation. When used with the **--sectors** option, no bounds checking is done on *sector_count* before attempting the operation. If *sector_count* is smaller than the current sector count the resize operation will still fail, however if *sector_count* is larger than the size of the disk, that will not be caught and the file system may have problems when attempting to use space be-

yond the end of the disk.

If the **--sectors** option is not specified or its argument is **all**, then if **sgresize**(8) is unable to determine the size of any LUN, that LUN will just be ignored for determining what size to use for the resize operation. Without the **--force** option, if **sgresize**(8) is unable to determine the size of any LUN in the stripegroup, the resize operation to fail.

--yes|**-y** Run the command without prompting for confirmation before making the change.

--verbose
Show additional output.

--debug|**-D**
Turn on debugging code

--diskcheck
Do a check for disks that are included in the stripe group that is being resized to see if they are currently in use in another file system that is visible to the cluster. In some configurations, this may take a long time. If there are disks in use, the operation is aborted.

--help Print a usage statement.

EXAMPLES

Resize the **foo** stripegroup in File System **snfs1**

```
rock # sgresize --fsname snfs1 --sgname foo
```

NOTE: The sgresize command is not available on Debian and Ubuntu clients.

SEE ALSO

cvadmin(8), **cvlabel**(8), **sgadd**(8), **sgdefrag**(8), **sgmanage**(8), **sgoffload**(8)

NAME

snaccess − display and edit the access.json file

SYNOPSIS

snaccess [**-e** '*command*; *command*; ...'] **snaccess** [**-h**]

snaccess < *file*

DESCRIPTION

snaccess is a command used to display and edit the **access.json** file. It is strongly encouraged that **snaccess** be used instead of editing the **access.json** file directly.

USAGE

When run without arguments from a terminal window, **snaccess** will prompt the user for input until the "quit" command is given. There are also two non-interactive modes. First, if input is redirected into the command from a file or a pipe, will process commands from stdin until EOF is reached. There is also **-e**:

−e '*command*; *command*; ...'
: This instructs **snaccess** to process one or more commands separated by semicolons. The **-e** option may be used multiple times on a single command line.

−h This causes **snaccess** to emit a brief usage message and exit.

COMMANDS

By design, commands alter an in-memory copy of the rule list and the on-disk version of the **access.json** file is not modified until the **save** command is used. This allows the user to validate the rule set using the **test** command prior to commiting the updates, if desired.

add type={allow | deny} *rule_options*
: Add a new rule to the end of list. The following *rule_options* are available:

  host=*hostspec*
  : Restrict the rule to hosts matching the *hostspec* is an IP address optionally followed by a "/" and the subnet netmask in bits. Examples include "host=10.101.224.1" and "host=10.101.0.0/16" with the former only matching a system whose address is "10.101.224.1" and the latter matching any system whose IP address begins with "10.101" Note that IP addresses supplied should correspond to the network used for metadadata traffic. If **host** is ommitted, the rule will match any host.

  file_systems=*fs1,fs2,...*
  : Restrict the rule to the specified list of file systems. If **file_systems** is not supplied, the rule applies to all file systems.

  path=*some/path*
  : Restrict the rule to the specified relative path. This path must not include the mount point. For example, if the intended full path is /stornext/snfs1/projects/current and the mount point is /stornext/snfs1, the syntax to use is **path=projects/current**. If **path** is not supplied, the rule is assumed to apply to the entire file system namespace.

  days={ mon | tue | wed | thu | fri | sat | sun, ... }
  : Restrict the rule to the specified list of days of the week, separated by commas. If **days** is not supplied, the rule is assumed to apply to every day of the week.

  start_time=HH:MM
  : Restrict the rule to a time-of-day period beginning with the specified **start_time**. If **start_time** is not supplied, the rule is assumed to apply starting at 00:00 (after midnight).

  end_time=HH:MM
  : Restrict the rule to a time-of-day period finishing with the specified **end_time**. If **start_time** is not supplied, the rule is assumed to apply ending at 23:59 (which is rounded up to midnight).

Example:

snaccess> add type=allow host=10.101.224.1 file_systems=snfs1 path=projects/current days=mon,tue,wed,thu,fri start_time=08:00 end_time=19:00

clear Remove all rules from the list.

delete *rule_number*
Remove the specified rule from the list.

disable *rule_number*
Deactivate the specified rule, but do not remove it from the list. A disabled rule remains in the rule list and can be reactivated using the **enable** command.

enable *rule_number*
Reactivate a rule previously deactivated using the **disable** command.

help [*command*]
Print a command summary, or details about the specified *command*.

history Show previously run commands.

insert {after | before} *rule_number* type={allow | deny} host=*hostspec rule_options*
Insert a new rule into the middle of the list. See description of *rule_options* under **add** above.

Example:

snaccess> insert after 2 type=allow host=10.101.224.1 file_systems=snfs1

move *rule_number* {after | before} *rule_number*
Change the location of a rule.

Examples:

snaccess> move 2 after 5

snaccess> move 8 before 8

quit[!] Exit the program. The trailing '!' is required to force exit when there are unsaved changes.

reload Discard all local changes and reload the access list from the server. See also the *-f* option to the *save* command.

replace *rule_number* type={allow | deny} host=*hostspec rule_options*
Replace the contents of the specified rule with new content. See description of *rule_options* under **add** above.

Example:

snaccess> replace 2 type=allow host=10.101.224.1 file_systems=snfs1

resume Turn StorNext Access feature back on after it was previously turned off with the **suspend** command. Note that like all other commands that update the configuration, the **save** command must also be used to have it take effect.

save [-f]
Write out pending rule modifications. This will also attempt to back up the previous **access.json** to a timestamped version under /usr/cvfs/config/config_history. When *-f* is specified, force the new access table to be written if it was updated by another user while local edits were being made. See also the *reload* command.

suspend
Turn off the StorNext Access feature without deleting the **snaccess.json** file. Note that like all other commands that update the configuration, the **save** command must also be used to have it take effect.

swap *rule_number rule_number*

Swap the positions of two rules.

Example:

snaccess> swap 2 5

test [-v] *host* fs *file_system* [path some/path] [at HH:MM] [on *day_of_the_week*]

Test whether a host has access to path on a given file_system at a specific time on particular day of the week. Host and fs are required arguments. The path defaults to the root of the file system, time defaults to the current time, and day_of_the_week defaults to the current day of the week. When *-v* is specified, details regarding each rule are displayed.

Examples:

snaccess> test host=10.101.224.1 fs=snfs1

snaccess> test host=10.101.224.1 fs=snfs1 path=projects/current day=monday time=10:00

SEE ALSO

snfs_access(7) **access.json**(5)

NAME

snacl − Display and modify ACLs

SYNOPSIS

snacl [-D] [-R] [-r] [-x] -l *file...*

snacl [-D] [-R] [-x] { +a | +ai | +a# *index* **| +ai#** *index* **}** *ACE file...*

snacl [-D] [-R] [-x] { -a *ACE* **| -a#** *index* **}** *file...*

snacl [-D] [-R] [-x] { =a# | =ai# } *index ACE file...*

snacl [-D] [-R] [-x] -E

snacl [-D] [-R] { -C | -i | -I | -N } *file...*

DESCRIPTION

snacl is used to display and modify Windows-style ACLs for files and directories on a StorNext file system.

OPTIONS

-l *file* The **-l** option is used to display security information including file ownership, Posix permission bits, and ACLs. Use of **-l** requires read permission on the specified *file* which may be a file or directory.

+a *ACE file*
The **+a** option is used to add an ACE to an ACL while maintaining canonical order. If the ACL for the file is not currently in canonical order, the use of **+a** is not allowed and **+a#** must be used instead when adding ACEs. See the section CANONICAL ORDERING below.

+ai *ACE file*
The **+ai** option behaves the same as the **+a** option except that added ACEs are marked with the inherited bit.

+a# *index ACE file*
The **+a#** option is used to add an ACE to an ACL at the specified index.

+ai# *index ACE file*
The **+ai#** option behaves the same as the **+a#** option except that added ACEs ae marked with the inherited bit.

-a *ACE file*
The **-a** option is used to delete ACEs. Existing DACL entries matching *ACE* are deleted if they match exactly. If an existing entry contains a superset of the rights specified by *ACE* only, the listed rights are removed. When using **-a** on a directory, ACEs for the directory's descendants are not affected unless the **-R** option is also specified.

-a# *index file*
The **-a#** option is used to delete an ACE at the specified index.

=a# *index ACE file*
The **=a#** option is used to replace an ACE at the specified index.

=ai# *index ACE file*
The **=ai#** option behaves the same as the **=a#** option except that added ACEs are marked with the inherited bit.

−E The **−E** option is used to assign ACEs to a file by reading values from stdin separated by newlines. If the file has an existing ACL, it is first removed.

−C The **−C** option is used to determine whether files contain ACLs in non-canonical order. When **-C** is used, *snacl* exits with a value of 0 if any of the specified files contains an ACL in non-canonical order. Otherwise, non-zero is returned.

−D The **−D** option enables debugging.

−i The **−i** option is used to remove the inherited bit from all ACEs in the specified files.

−I The **−I** option is used to remove all inherited ACEs from specified files.

−N The **−N** option is used to completely remove the ACL (all ACEs) from the specified files.

−r The **−r** option forces snacl to display raw SIDs when listing ACEs instead of mapping them to user and group names.

−R The **−R** option causes snacl to perform the requested operation recursively.

−x The **−x** option causes snacl to operate in "expert" mode when displaying security information so that additional, low-level information about the security descriptor is shown. This option may be removed in a future release or its output format changed. This option can also be used when modifying ACLs. In this context, it allows ACLs to be updated even when the security model for the file system is not set to "acl" (which is normally required).

ACE FORMAT

ACEs are specified using the following syntax:

principal { allow | deny } *permissions_and_inheritance_flags*

The *principal* is the name of a user or group. In cases where a user and group exist with the same name, it must be prefixed with "user:" or "group:" to remove ambiguity. If a user or group name contains a space, use colons (:) as a delimeter. For example: **"group:domain users:allow:read"**

The *permissions_and_inheritance_flags* field is a comma-separated list of permissions and inheritance flags that may be mixed. Each ACE must have at least one permission specified.

The following section provides all the permission keywords and a description of the actions they grant.

The following permissions apply to both files and directories:

delete Remove the file or directory. Deletion is granted either by this permission or the delete_child on the parent directory.

readattr Read the basic attributes from a file or directory. This is implicitly granted if the file or directory can be looked up and it is not explicitly denied.

writeattr Update basic attributes for a file or directory.

readextattr List or read the extended attributes for a file or directory.

writeextattr Update extended attributes for a file or directory.

readsecurity Read the security information for a file or directory.

writesecurity Write the security information for a file or directory.

chown Change the ownership of a file or directory.

The following permissions apply only to directories:

list List entries (readdir).

search Look up files by name. In addition to stat(2), this permission is required for name space operations such as file creation, deletion, and rename.

add_file Create a file

add_subdirectory
Create a subdirectory

delete_child Delete a file or subdirectory. Deletion is granted either by this permission on the parent or delete permission on the target.

The following permissions apply to files only:

read Open for reading.

write Open for writing.

append Open a file for appending writes.

execute Execute the file.

The following are inheritance flags which only apply to directories:

file_inherit ACE should inherit to (non-directory) files.

directory_inherit ACE should inherit to directories.

limit_inherit This flag prevents newly created subdirectories which inherited the specified ACE from its parent from further propagating the ACE to its children.

only_inherit This flag causes the ACE to be inherited to files and subdirectories but not used for permission processing on the directory. Note that the only_inherit flag is never inherited.

There are also special flags for generic ACL combinations:

read_access
This is shorthand for: list,search,readattr,readextattr,readsecurity,file_inherit,directory_inherit

modify_access
This is shorthand for: list,add_file,search,delete,add_subdirectory,readattr,writeattr,readextattr,writeextattr,readsecurity,file_inherit,directory_inherit

full_access
This is shorthand for: list,add_file,search,delete,add_subdirectory,delete_child,readattr,writeattr,readextattr,writeextattr,readsecurity,writesecurity,chown,file_inherit,directory_inherit

CANONICAL ORDERING

ACEs are always processed in the order they are listed in the ACL. However, there is a preferred order for ACEs that typically results in the simplest ACL that achieves the desired result. This is called canonical order and its use is a best practice. The order is:

1. Explicit Denied ACEs (if any)
2. Explicit Allowed ACEs (if any)
3. Inherited Denied ACEs (if any)
4. Inherited Allowed ACEs (if any)

When ACEs are always added using **snacl +a**, canonical ordering is preserved. If ACEs are added using **snacl +a#** or **snacl +ai#** or if they are modified using **snacl =a#**, canonical ordering may be broken. Also, ACEs are inherited in the same order as the parent so if the ACL on the parent directory does not use canonical ordering, the child may inherit ACEs in non-canonical order. The ACL on a file can be tested for canonical order using **snacl -C**.

EXAMPLES

List the ACL for a file.

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane allow write
 1: group:staff allow write,delete
```

Add an ACE to the ACL of a file, maintaining canonical order:

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane allow write
$ snacl +a "user:fred deny read,write" foo
$ snacl -l foo
-rw-r--r-- joe staff foo
```

```
 0: user:fred deny read,write
 1: user:jane allow write
```

Add an ACE to the ACL of a file, for a group containing a space. Because the space, colons (:) are used as delimiters.

```
$ snacl +a "group:domain users:allow:read" foo
$ snacl -l foo
-rw-r--r--+ 1 joe staff foo
 0: group:domain users allow read
```

Add an ACE to the ACL of a file, maintaining canonical order.

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane allow write
$ snacl +ai "user:fred allow write" foo
 0: user:jane allow write
 1: user:fred inherited allow write
```

Add an ACE to the ACL of a file at a particular index disregarding canonical ordering:

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane allow write
$ snacl +a# 1 "user:fred deny write" foo
 0: user:jane allow write
 1: user:fred deny write
```

Remove read permission for a user. Note that since the ACE contains other permissions, it is not removed completely.

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane allow write,delete
$ snacl -a "user:jane allow delete" foo
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane allow write
```

Remove a particular ACE from an ACL.

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane allow write,delete
 1: user:fred allow delete
 2: user:wendy allow write,delete
$ snacl -a# 1 foo
$ snacl -l foo
 0: user:jane allow write,delete
 1: user:wendy allow write,delete
```

Replace an ACE:

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane allow write,delete
 1: user:fred allow delete
 2: user:wendy allow write,delete
$ snacl =a# 1 "john allow write" foo
$ snacl -l foo
-rw-r--r-- joe staff foo
```

```
 0: user:jane allow write,delete
 1: user:john allow write
 2: user:wendy allow write,delete
```

Assign ACE values from stdin:

```
$ cat myacl.txt
user:jane allow write
user:joe deny read
$ snacl -E foo < myacl.txt
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane allow write
 1: user:joe deny read
```

Check for canonical order:

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:joe deny write
 1: user:jane allow write
$ snacl -C foo
$ echo $?
1
$ snacl -l bar
-rw-r--r-- joe staff foo
 0: user:jane allow write
 1: user:joe deny write
$ snacl -C bar
$ echo $?
0
```

Remove inherited bit from ACEs.

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane inherited allow write
 1: user:joe allow write
$ snacl -i foo
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane allow write
 1: user:joe allow write
```

Remove inherited ACEs:

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane inherited allow write
 1: user:joe allow write
$ snacl -I foo
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:joe allow write
```

Remove entire ACL:

```
$ snacl -l foo
-rw-r--r-- joe staff foo
 0: user:jane inherited allow write
```

```
 1: user:joe allow write
$ snacl -N foo
$ snacl -l foo
-rw-r--r-- joe staff foo
```

NOTES

Display and manipulation of ACLs requires identity mapping to work properly. Therefore, **snacl** will generate a warning or error when performing certain operations if the file system is not configured with the "acl" security model. The "-x" (expert) flag can be used to override this behavior and allow ACLs to be modified when the security model is not set to "acl". Also note that the "acl" security model is required for ACL enforcement on Linux/Unix.

SEE ALSO

snfs_config(5)

NAME

snaudit − StorNext Audit Reporting Utility

SYNOPSIS

snaudit [*action*] [*options*] [*pathname*]

DESCRIPTION

The **snaudit** command generates reports describing the changes that have occurred to specific files or directory trees within the *StorNext file system*.

In order for the audit feature to work, the **audit** file system configuration variable must be true. Also, **metadataArchive** and **metadataArchiveSearch** must be true and the **metadataArchiveDays** must be non-zero.

When the audit feature is enabled, the FSM asks the file system clients to send information about what file I/O they've performed. The FSM then records this information in the Metadata Archive. The **snaudit** tool queries the data from there. As a consequence of the data originating in the client, some information might not be gathered from older clients who don't know how to send the necessary data.

Sometimes for highly metadata-intensive workloads, the FSM can get behind applying update bundles to the Metadata Archive. It may take a few minutes for the system to catch up. The **snaudit** tool may block waiting for this to happen. You can use the **mdarchive status** command in **cvadmin** to check to see if there are bundles waiting to be applied.

For the **-H** action (or no action), the output of snaudit is a series of operations that have been performed on one or more files. The list of possible operations is:

create The creation of a new file.

link Creation of a hard link.

unlink Removal of the file or a link to the file.

alloc Allocation of blocks to a file.

read Read of a file.

write Write of a file.

rdwr A mixture of reads and writes.

rdwr_total
A mixture of reads and writes where the system doesn't have enough information to show the exact number of bytes transferred. Instead lifetime totals are shown.

rename Rename of a file.

modebits Change of a file's modebits (i.e. permissions)

uidgid Change of a file's user or group owner.

ntsd_key Change of a file's NT Security Descriptor Key and/or ACLs.

setsize Change in file size.

ACTIONS

-H This action causes **snaudit** to query the metadata archive of a file system and output a history of events for the given path. If the path is a regular file, the history for that file is returned. If the path is a directory, the histories for all files within that directory tree are returned. Only one path argument is accepted. This is the default action taken if no actions are specified.

-L This action returns time, user, and hostname of the latest access to the given file. If the **−v** option is also present, it returns the name of the process which generated the latest access and the cumulative reads, write, and opens for the file. This option only works for regular files.

-T This action causes snaudit to print the next TID to be generated by in a Metadata Archive. It is most useful in conjunction with the **-F** option to specify a live file system.

OPTIONS

−D Print out debugging information.

−d *pathname*

This option instructs **snaudit** to query for audit information directly from the metadata archive that exists in the directory specified by pathname rather than sending a request to an activated FSM to query the metadata archive. It is generally only useful if you have a metadata archive that has been separated from the file system which created it. When using this option, the pathnames you specify don't correspond to any mounted path. They are all relative to the file system root and should start with a slash. See the **EXAMPLES** section. This option is mutually exclusive with the **−F** and **−L** options.

−F *FsName*

In most cases while using **snaudit**, the tool will automatically figure out which file system you intend to query based on which files and directories you give it. When automatic detection doesn't work (the tool will let you know), use this option to specify the file system name. Using this option will cause snaudit to send a request to the file system's activated FSM to retrieve audit information from its metadata archive. This option is mutually exclusive with the **−d** option.

-h This option causes **snaudit** to print a friendly help message and exit.

-n *number*

When used with the -H (default) action, this option limits output to only *number* thousand events. This option can be used together with the **-t** option read out file system history in bite-sized chunks. Note that this event limit is only a rough, ball-park limit. Warning: A value greater than ten may cause **snaudit** to use an unwieldy amount of memory.

-o {**text**|**json**|**cjson**|**csv**}

Print output in **text**, **json**, **cjson** (compact JSON), or CSV format. The default is **text**.

-t *sTID[:eTID]*

When used with the -H (default) action, this option causes only events in a certain range of Transaction IDentifiers (TIDs) to be reported. TIDs are just positive integers. A single TID (such as "-t 42") is interpreted as a starting TID and all events with TIDs equal or greater to that number are reported. A colon followed by a TID (such as "-t :34253251") specifies an ending TID and all events with TIDs less than or equal to that TID are reported. A TID range (such as "-t 42:34253251") may also be specified.

−q *filename*

Output qustat counters to named file on completion.

-v Change the text output of the command to include exactly what changed for each event. For example, it includes the number of bytes written on write events.

−w *waittime*

Set the number of minutes that snaudit should wait for the FSM to respond to requests for audit history before timing out. By default, snaudit waits two minutes for the FSM to respond.

EXIT VALUES

snaudit will return 0 on success and non-zero on failure.

NOTES

Note there is fundamental trade-off made by **snaudit** and **MetaData Archive**: performance vs accuracy. To improve performance, the StorNext's FSM combines many transactions into one when adding them to the **MD Archive**. Many inode updates get combined into one. This means fewer operations and better performance. Of course, this means that the **MD Archive** doesn't contain every step that happens to a file or file system. So, snaudit can't always show every step.

There are a number of factors that influence when transactions are combined or not. Timing is a big one. Also, transactions from different machines or users are never combined. So, sometimes a bunch of updates made by one entity in rapid succession will be shown individually and sometimes they won't.

Snaudit tries to to break up combined updates and show better what happened. This involves looking at the previous state of the inode and seeing what changed. Then, individual events are synthesized. That idea doesn't work well with a file that is newly created. For example, sometimes a create, a data block allocations, and a change of permission bits get combined into one **snaudit** event.

Also, note that updates to the file system history can take a while to be recorded. For example, allocations events can be delayed by the buffer cache as it tries to perform efficient allocation. Also, read/write events are not seen by **MetaData Archive** until the client performing them closes the file.

Also, note that some applications have interesting behavior around updating files that will affect **snaudit** results. For example, when Vim saves a new version of a text file, it unlinks the old file and writes a new one. If you run **snaudit** on the file before and after the save, it looks like history on the original file disappears. In reality, they are two different files with different histories.

The application rsync has a similar behavior where it never overwrites a file. It writes a temporary file with a slightly different name and renames it over top of the original file. Depending on how the **MetaData Archive** transactions have been combinied, the rename step may or may not be visible in the **snaudit** output. In all cases, the history of the original file is no longer associated with the new file.

When using NFS or SMB clients, the host recorded by **snaudit** is the IP address of the NAS server and not the end client. Also, when I/O is performed by NAS-attached clients, the name of the recorded process is the name of the server daemon such as nfsd, smbd, or System.

The "user" captured by **snaudit** is the Unix UID of the process performing the operation. Therefore, when using StorNext Windows SAN or LAN clients, in order for the **snaudit** user to be meaningful, identity mapping must set up appropriately so that operations performed on Windows are recorded with a Unix UID corresponding to the appropriate account. One common way of setting up identity mapping is to use RFC2307 extensions to Active Directory. In addition to configuring identity mapping, for best results, systems where the **snaudit** command is run such as a Linux StorNext appliance should be joined to the directory service so that the raw UIDs that are captured by **snaudit** on Windows clients can be converted to user-friendly account names by the system. StorNext Apppliances can be joined to a directory service using the Connect "Manage NAS" application or using the "auth config" command in the StorNext NAS command line interface.

When buffered I/O is used, the UID recorded for allocations may correspond to a system account such as root even when files are being written by regular users. Because in such cases, the allocation is performed asynchronously by a background system thread.

EXAMPLES

By default, **snaudit** lists what happened to a file.

```
# cd /stornext/snfs1/linux/include/
# snaudit -v linux/net.h
        op  time                    user  host            name
        --  ----                    ----  ----            ----
    create  2017-02-06 16:19:04        0  10.65.177.100   /linux/include/linux/
  modebits  2017-02-06 16:19:04        0  10.65.177.100   /linux/include/linux/
            mode: 0100600 -> 0100644
    uidgid  2017-02-06 16:19:04        0  10.65.177.100   /linux/include/linux/
            uid: 0 -> 596, gid: 0 -> 400
    rename  2017-02-06 16:19:04        0  10.65.177.100   /linux/include/linux/n
            prev_name: /linux/include/linux/.net.h.VAgm6M
     alloc  2017-02-06 16:19:07        0  10.65.177.100   /linux/include/linux/n
     write  2017-02-06 16:19:09        0  10.65.177.100   /linux/include/linux/n
            bytes: 11K
      read  2017-02-06 16:37:25      596  10.65.188.123   /linux/include/linux/n
            bytes: 11K
```

The -L option causes **snaudit** to list current information about the file.

```
# snaudit -L -o json acpi.h
{
    "tid_start": 1,
    "tid_next": 4573,
    "auditData": [
        [
            {
                "op": "rdwr_total",
                "time": "1487196468.744789",
                "time_pretty": "2017-02-15 16:07:48",
                "host": "10.65.177.100",
                "user": 0,
                "name": "/linux/include/linux/acpi.h",
                "reads_total": 0,
                "writes_total": 34240,
                "num_opens": 1,
                "proc_name": "rsync",
                "detail": "bytes_read=0, bytes_written=33K, num_opens=1, proc_nam
            }
        ]
    ],
    "events": 1,
    "returnCode": 0,
    "more": false
}
```

The -d option specifies a path to a metadata archive that is not associated with an active FSM. This option should not be used to query the metadata archive of an active file system.

Pathnames to user files or directories are relative to the root of the file system. They always start with slash (/) and don't include the mount point.

```
# snaudit -d /<Path-to-static-mdarchive>/snfs1-mdarchive /linux/Makefile
        op  time                    user  host            name
        --  ----                    ----  ----            ----
    create  2017-02-06 16:16:36        0  10.65.177.100   /linux/.Makefile.VCAWY
  modebits  2017-02-06 16:16:36        0  10.65.177.100   /linux/.Makefile.VCAWY
    uidgid  2017-02-06 16:16:36        0  10.65.177.100   /linux/.Makefile.VCAWY
    rename  2017-02-06 16:16:36        0  10.65.177.100   /linux/Makefile
     alloc  2017-02-06 16:16:37        0  10.65.177.100   /linux/Makefile
     write  2017-02-06 16:16:42        0  10.65.177.100   /linux/Makefile
```

SEE ALSO

cvadmin(8), **snfs_config**(5), **snhistory**(8)

NAME

sncfgconvert − Convert StorNext File System configuration file format

SYNOPSIS

sncfgconvert -n *fsname* [**-F** *output_format*] [**-v**] **-f** *ConfigFile*

sncfgconvert -n *fsname* [**-F** *output_format*] [**-v**] *ConfigFile*

sncfgconvert -h

DESCRIPTION

The **sncfgconvert** program will convert a StorNext file system configuration file between the old pre-4.0 config and the new XML config file formats (on supported platforms).

The input file need not have a .cfg or .cfgx suffix - the converter will automatically figure out the format of the input file. Converted output is displayed on stdout.

Old format output files should have the .cfg suffix and XML formatted files should have the .cfgx suffix when writing them to the */usr/cvfs/config* directory. See **snfs_config**(5) for more information.

OPTIONS

−F *output_format*
: The format to convert to. See **-h** for applicable formats. This can also be used to "convert" from one format to the same format. Defaults to the latest format available for the system.

−h Display usage, including available formats to convert to.

−n *fsname*
: Required. The name of the file system whose config is to be converted.

−f *ConfigFile*
: Specify the config file.

−v Increase verbosity of messages from conversion process

EXIT VALUES

sncfgconvert will return 0 on success and non-zero on failure.

EXAMPLES

List usage:

```
# sncfgconvert -h
Usage: sncfgconvert -n <fsname> [-F <output_format>] ([-f <filename>] | <filename
  -F <output_format>    Valid formats:
                        XML (default)
                        ASCII
  -f <filename> Config file to convert
  -n <fsname>   File system name
  -v            increase verbosity (specify multiple -v's for more verbosity)
```

This also displays that the default format is XML.

Convert a config in a temporary location to XML and write it to a different tempfile (see **sncfginstall**(8) for details on installing the config to the proper location.

```
# sncfgconvert -F XML -n snfs1 -f /tmp/fs1.copy > /tmp/fs1.xml
```

Convert a config in a temporary location to the old ASCII format and display it on stdout:

```
# sncfgconvert -F ASCII -n test /tmp/test.cfgx
# Globals

AllocationStrategy Round
HaFsType HaUnmonitored
```

```
        FileLocks No
        BrlResyncTimeout 20
         ...
        RtTokenTimeout 0
        MultiPathMethod Rotate
        Node CvfsDisk_jbod0203 0
```

SEE ALSO

sncfginstall(8), **snfs_config**(5)

NAME

sncfgedit − Edit StorNext File System configuration file

SYNOPSIS

sncfgedit -n *FsName* [**-f** *FailedTemporaryConfigFile*]

sncfgedit -h

DESCRIPTION

The **sncfgedit** program provides convenient method to edit the StorNext file system configuration file for the named file system and validate the new configuration file before overwriting the old one. To do so, **sncfgedit** creates a temporary copy of the configuration file, and then opens it in an editor.

sncfgedit uses the the EDITOR environment variable to choose the program used to edit the configuration file, or **vi** if EDITOR is not set.

Validation includes standalone validation of the new configuration file as well as validation of changes between original configuration file and edited file if a configuration file previously existed. If no previous configuration file exists, a template will be loaded into the editor using default settings for all values.

Before an existing configuration file is overwritten, it is copied to */usr/cvfs/data/*FsName*/config_history/* for future reference. A timestamp is appended to the filename to allow for the existence of multiple backup copies.

OPTIONS

−h Display usage.

−n *FsName*
Required. The name of the file system whose config is to be edited or created.

−f *FailedTemporaryConfigFile*
Specify the path to the temporary config file from a previous run where the proposed changes were disallowed. This gives you another chance without throwing away all of the previous changes.

EXIT VALUES

sncfgedit will return 0 on success and non-zero on failure.

ENVIRONMENT VARIABLES

EDITOR
Allows the user to choose which editor is used to modify the configuration file. (default: vi)

FILES

*/usr/cvfs/data/*FsName*/config_history/**

SEE ALSO

snfs_config(5), **snfs.cfgx**(5), **snfs.cfg**(5)

NAME

sncfggen − Generate a StorNext File System configuration file from a reference configuration and a json input file

SYNOPSIS

sncfggen -f *json_file* **-n** *FsName* **-r** *refconfig* **-p** *refpath* [**-d**]

sncfggen -h

DESCRIPTION

The **sncfggen** program will generate a StorNext file system configuration file from a reference configuration file system and a json input file. The json file is written in JSON which is an open standard light weight data exchange language.

The json file is required to contain a **fileSystems** section which contains an entry with a name that matches the file system name. Global variables are specified as key/value pairs. These will replace the settings of those variables in the reference configuration. The json file must also contain a **stripeGroups** section. The **stripeGroups** array contains one entry for each stripe group and must contain the name of the stripe group. In each stripe group the **sgDisks** array of disk labels must be present. **Sncfggen** will generate a configuration file with the file system name filled in, changes made to global variables and disk labels updated for the disks in the stripe groups.

The new config file will be written to */usr/cvfs/config.*

OPTIONS

−h Display usage.

−f *json_file*
Required. Specify the json input file.

−r *refconfig*
Required. Specify the fs name of the reference configuration file system.

−p *refpath*
Required. Specify the file path to reference configuration file system.

−n *FsName*
Required. The name of the file system whose config is to be generated.

-d Optional. Turn on debug mode.

EXIT VALUES

sncfggen will return 0 on success and non-zero on failure.

EXAMPLES

List usage:

```
# sncfggen -h
usage sncfggen -h
      sncfggen -f json file -r refconfig -p refpath -n fsname -d
               -h              # display usage
               -f json_file    # json file to load
               -r refconfig    # refconfig fs name
               -p refpath      # refconfig file path
               -n fsname       # new file system name
               -d              # turn on debug mode
#
```

Generate */usr/cvfs/config/sl1.cfgx* from */tmp/refconfig.cfgx* and json input file *json_file.*

```
# sncfggen -f /tmp/json_file -r refconfig -p /tmp/refconfig.cfgx -n sl1
Successfully generated configuration file /usr/cvfs/config/sl1.cfgx.
#
```

Example json file.

```
# cat json_file
{
    "fileSystems": [
        {
            "name": "sl1",
            "bufferCacheSize": 4294967296,
            "fileLocks": false,
            "quotas": true,
            "stripeGroups": [
                {
                    "sgName": "sg0",
                    "sgDisks": [
                        "sl_0005"
                    ]
                },
                {
                    "sgName": "sg1",
                    "sgDisks": [
                        "sl_0006"
                    ]
                },
                {
                    "sgName": "sg2",
                    "sgDisks": [
                        "sl_0007",
                        "sl_0008"
                    ]
                }
            ]
    }
    ]
}
#
```

SEE ALSO

snfs_config(5)

NAME

sncfginstall − Install StorNext File System configuration file

SYNOPSIS

sncfginstall -n *FsName* **-f** *config_file* [**-M** *msg_format*]

sncfginstall -h

DESCRIPTION

The **sncfginstall** program will install a StorNext file system configuration file into the proper location, after validating that it is a valid config file, and comparing it against an already existing config file for the named file system if one exists.

The input file need not have a .cfg or .cfgx suffix - the converter will automatically figure out the format of the input file.

OPTIONS

−h Display usage.

−n *FsName*
Required. The name of the file system whose config is to be installed.

−f *config_file*
Specify the config file to install.

−M *msg_format*
Specify *XML* if you want messages to come back in XML format. This defaults to ASCII.

EXIT VALUES

sncfginstall will return 0 on success and non-zero on failure.

EXAMPLES

List usage:

```
# sncfginstall -h
Usage: sncfginstall -f <config_path> -n <fs_name> [-M <fmt>]
  -f            Path to file to install
  -n            Name of file system to install
  -M            Message format: ASCII, XML (default ASCII)
  -h            This usage
```

Install a config in a temporary location to the StorNext configuration directory:

```
# sncfginstall -n snfs1 -f /tmp/fs1.copy
'snfs1' installed
```

FILES

*/usr/cvfs/data/*FsName*/config_history/*

SEE ALSO

snfs_config(5)

NAME

sncfgquery − Query StorNext File System configuration files

SYNOPSIS

sncfgquery -d -f *config_path*

sncfgquery -d -n *fsname*

sncfgquery -l [**-s** *section*]

sncfgquery -q -s *section* **-k** *keyword* **-f** *config_path*

sncfgquery -q -s *section* **-k** *keyword* **-n** *fsname*

sncfgquery -h

DESCRIPTION

Query a given file system for configuration file settings. Only a subset of configuration file settings are supported.

ACTION OPTIONS

-d Dump configuration

-l List recognized keywords. Without any query options, this returns the recognized sections. If the **-s** flag is specified, it lists the recognized keywords within that section.

-q Query configuration

-h Print usage

QUERY OPTIONS

-s *section*
Select the specified section

-k *keyword*
Select the specified keyword within a section

LOCATION OPTIONS

-f *config_path*
Find information based on config file path

-n *fsname*
Find information based on file system name

EXIT VALUES

sncfgquery will return 0 on success and non-zero on failure.

EXAMPLES

List usage:

```
# sncfgquery -h
Usage: sncfgquery <actions> <query>

Locate config:
 -f <config>    Find information based on config file location
 -n <fsname>    Find information based on file system name
actions:
 -d             Dump config
 -l             List keywords
 -q             Query config
 -h             Print this help info
options:
 -k <keyword>   keyword
 -s <section>   section
```

List the keywords in the *globals* section:

```
# sncfgquery -l -s globals
Valid keywords for the globals section:
        BrlResyncTimeout
        DrCopy
        EventFileDir
        GlobalSuperUser
        HaFsType
        RenameTracking
        RestoreJournal
        MetadataArchive
        StorageManager
        AffinityPreference
```

Query the HaFsType of a filesystem

```
# sncfgquery -q -s globals -k HaFsType -n snfs1
HaFsType HaUnmonitored
```

SEE ALSO

snfs_config(5)

NAME

sncfgremove − Remove a StorNext File System configuration file

SYNOPSIS

sncfgremove -n *FsName* [**-M** *output_format*]

sncfgremove -h

DESCRIPTION

The **sncfgremove** program will remove a StorNext file system configuration file, archiving it to */usr/cvfs/data/*FsName*/config_history/* in the process.

OPTIONS

−h Display usage.

−n *FsName*
Required. The name of the file system whose config is to be removed.

−M *msg_format*
Specify *XML* if you want messages to come back in XML format. This defaults to ASCII.

EXIT VALUES

sncfgremove will return 0 on success and non-zero on failure.

EXAMPLES

List usage:

```
# sncfgremove -h
Usage: sncfgremove -n <fs_name> [-M <fmt>]
  -n            Name of file system to remove
  -M            Message format: ASCII, XML (default ASCII)
  -h            This usage
```

Remove a config:

```
# sncfgremove -n snfs1
'snfs1' successfully removed
```

FILES

*/usr/cvfs/data/*FsName*/config_history/*

SEE ALSO

snfs_config(5)

NAME

sncfgtemplate − Output a StorNext File System configuration file

SYNOPSIS

sncfgtemplate -n *FsName* [**-M** *output_format*]

sncfgtemplate -h

DESCRIPTION

The **sncfgtemplate** program will output a template StorNext file system configuration file with the given name to stdout. This can be redirected to a file and edited. When the configuration file is correct for the file system being created, it can be installed using **sncfginstall**(8), and made with **cvmkfs**(8).

Note: the standard way to create a new configuration file on the commandline is via **sncfgedit**(8).

OPTIONS

−h Display usage.

−n *FsName*
Required. The name of the file system whose config is to be created.

−M *msg_format*
Specify *XML* if you want messages to come back in XML format. This defaults to ASCII.

EXIT VALUES

sncfgtemplate will return 0 on success and non-zero on failure.

EXAMPLES

List usage:

```
# sncfgtemplate -h
Usage: sncfgtemplate -n <fs_name>
  -n            Name of file system
  -M            Message format (default ASCII)
  -h            This usage
```

Dump a template config to a temporary file:

```
# sncfgtemplate -n snfs1 > /tmp/myconfig
```

SEE ALSO

sncfginstall(8), **cvmkfs**(8), **snfs_config**(5)

NAME

sncfgtransform − Check two StorNext File System configuration files for a valid transformation

SYNOPSIS

sncfgtransform [-h] -n *FsName caller file1 file2*

DESCRIPTION

The **sncfgtransform** program will validate the transformation between two StorNext file system configuration files for the given "caller" and file system.

The valid *caller* values are:

fsm The transform is checked for a file system manager (fsm) restart. This is the usual transform that administrators are looking for since a change in the config file is usually followed by a file system restart.

cvupdatefs
: The transform is checked for a run of the cvupdatefs command. This can be used when adding a stripe group or for stripe group expansion.

updatefs
: Same as cvupdatefs.

cvfsck The transform is checked for a run of cvfsck.

dbg The transform is checked for the cvfsdb command.

cvntfscfg
: The transform is checked with config files in ASCII format, typically used only on Windows MDCs.

snadmin
: The transform is checked with callers of the snadmin library API.

cvmkfs The transform is checked for the cvmkfs command.

cvmkfsr
: The transform is checked for the cvmkfs command with the -r option which means that the file system meta data is being restored from a database created because the restore journal was configured.

convert The transform is checked for the sncfgconvert command.

fsmpm The transform is checked for the fsmpm utility.

mdarchive
: The transform is checked for the snhistory and snaudit commands and other logic that interacts with the metadata archive.

cfggen The transform is checked for the sncfggen command.

dcap The transform is checked for the disk capacity license logic.

sgmanage
: The transform is checked for sgmanage and other stripe group management commands.

validate
: The transform is checked for the sncfgvalidate command.

Two configurations files must be given and the transformation is assumed to be from *file1* to *file2*.

Each configuration file is parsed and may fail. If they both succeed parsing, the transformation is checked with any errors being displayed.

OPTIONS

−h Display usage.

−n *FsName*

Required. The name of the file system whose config files are given.

EXIT VALUES

sncfgtransform will return 0 on success and non-zero on failure.

EXAMPLES

List usage:

```
# sncfgtransform -h
Usage: sncfgtransform [-h] -n <fsname> <caller> <file1> <file2>
  -h        This usage
  -n        Name of file system to validate
  caller
        fsm
        cvfsck
        cvmkfs
        cvmkfsr
        dbg
        cvntfscfg
        snadmin
        sncfgconvert
        fsmpm
        cvupdatefs
        updatefs
        mdarchive
        cfggen
        dcap
        sgmange
        validate
  file1     current configuration file
  file2     proposed configuration file
```

Check a new config file under consideration for snfs1 with a copy in /tmp (OK):

```
# sncfgtransform -n snfs1 fsm /tmp/snfs1.cfgx /tmp/snfs1.new.cfgx
'snfs1' transformation OK
```

Check a new config file under consideration for cvupdatefs of snfs1 (OK):

```
# sncfgtransform -n snfs1 updatefs /tmp/snfs1.cfgx /tmp/snfs1.new.cfgx
'snfs1' transformation OK
```

Check a modified config against the installed config (with a valid change for cvupdatefs))

```
# sncfgtransform -n snfs1 updatefs /usr/cvfs/config/snfs1.cfgx /tmp/cfgx
'snfs1' transformation OK
```

Check a modified config against the installed config (with an invalid change)

```
# sncfgtransform -n snfs1 fsm /tmp/snfs1.cfgx /tmp/snfs1.new.cfgx
transformation failed for /tmp/snfs1.cfgx -> /tmp/snfs1.new.cfgx with -1
transformation for /tmp/snfs1.cfgx -> /tmp/snfs1.new.cfgx -- error: Stripe group

./sncfgtransform transformation failed
```

SEE ALSO

cvfsdb(8) **fsmpm**(8) **snaudit**(8) **snfs_config**(5) **sncfgconvert**(8) **sncfgtransform**(8) **sncfgvalidate**(8) **sncfggen**(8) **sgmanage**(8) **snhistory**(8)

NAME

sncfgvalidate − Validate a StorNext File System configuration file

SYNOPSIS

sncfgvalidate -n *FsName* [**-f** *config_file*] [**-M** *msg_format*]

sncfgvalidate -h

DESCRIPTION

The **sncfgvalidate** program will validate a StorNext file system configuration file.

If only an *FsName* is specified, **sncfgvalidate** looks for an already installed configuration file for the named file system, and validates its syntax if it exists.

If an *FsName* and a *config_file* are specified, the specified config file's internal syntax is validated, and it is compared against the currently installed configuration file for the named file system (if it exists).

OPTIONS

−h Display usage.

−n *FsName*
Required. The name of the file system whose config is to be validated.

−f *config_file*
Specify the config file to validate.

−M *msg_format*
Specify *XML* if you want messages to come back in XML format. This defaults to ASCII.

EXIT VALUES

sncfgvalidate will return 0 on success and non-zero on failure.

EXAMPLES

List usage:

```
# sncfgvalidate -h
Usage: sncfgvalidate -n <fs_name> [-f <config_path>] [-M <fmt>]
  -f            Path to file to validate
  -n            Name of file system to validate
  -M            Message format: ASCII, XML (default ASCII)
  -h            This usage
```

validate an installed config:

```
# sncfgvalidate -n snfs1
'snfs1' validated
```

validate a new config

```
# sncfgvalidate -n snfs2 -f /tmp/mysnfs2
'snfs' validated
```

validate a modified config against the installed config (with an invalid change)

```
# sncfgvalidate -n snfs1 -f /tmp/mysnfs1
warning: Journal Size of 16M is less than recommended minimum value; It must be a
warning: new fsBlockSize (32768) does not match existing fsBlockSize (16384), fi
```

SEE ALSO

snfs_config(5)

NAME

snconfig_impexp − StorNext Configuration Import/Export Tool

SYNOPSIS

snconfig_impexp *action* [*options*]

DESCRIPTION

The **snconfig_impexp** command allows machine configuration of some of the various configuration files that StorNext uses. Current values can be returned in JSON format via stdout. A JSON object can be passed to stdin to cause the tool to update the configuration. All this is done as directed by the actions described below.

In almost all cases, a JSON object is written to stdout. It will have keys describing the success or failure of the requested action. The **return_code** key is zero if the action succeeded or one if the action failed. If the action failed, an **error** key will be present describing the problem. On a non-fatal error, a **warnings** key will be present. These keys will be ignored in a JSON object passed to **snconfig_impexp**.

When **snconfig_impexp** modifies files, it first places backup-ups of the files into the /usr/cvfs/config_history directory.

ACTIONS

-R This action causes **snconfig_impexp** to read various StorNext configuration files and return their contents in a JSON object written to stdout.

-W This action takes a JSON object read from stdin, parses it, and writes its contents into the appropriate configuration files. The format of the JSON object is the same as that returned by the **-R** action. All top-level keys (other than the **snconfig_impexp_version** key), are optional. If a top-level key is present, the key's values are written to the appropriate configuration files, replacing anything that was there. If a top-level key is not present, then no changes are made to the configuration represent by the key. After all configuration changes are made, the current state is returned exactly as if **-R** was specified.

OPTIONS

-h This option causes **snconfig_impexp** to print a friendly help message and exit. This is the only output which is not in JSON format.

-v Format JSON output in a human-readable fashion.

EXIT VALUES

snconfig_impexp will return 0 on success and non-zero on failure.

EXAMPLES

A fully-populated output of **snconfig_impexp** looks like:

```
# snconfig_impexp -vR
{
    "snconfig_impexp_version": 1,
    "nss_coordinators": [
        {
            "ip": "10.11.12.13"
        },
        {
            "ip": "10.11.12.14",
            "cluster": "cluster12",
            "addom": "my_addom"
        },
        {
            "ip": "192.168.20.5",
            "masklen": 29
        }
```

```
        ],
        "cluster": {
            "default_cluster": "cluster12",
            "default_addom": "my_addom"
        },
        "fsms": [
            {
                "fsname": "snfs1"
            },
            {
                "fsname": "snfs2"
            },
            {
                "fsname": "snfs3",
                "cluster": "cluster12",
                "addom": "my_addom",
                "priority": 100
            }
        ],
        "return_code": 0
    }
```

An unsuccessful run (because it was run as a non-privileged user) looks like:

```
    # snconfig_impexp -R | snconfig_impexp -vW
    {
        "snconfig_impexp_version": 1,
        "return_code": 1,
        "error": "nss_coordinators: error opening file ’/usr/cvfs/config/fsnameserver
    }
```

SEE ALSO

fsmcluster(4), **fsmlist**(4), **fsnameservers**(4)

NAME

sndiskmove − StorNext File System Disk Mover Utility

SYNOPSIS

sndiskmove [**-fqv**] [**-s** *suffix*] [**-b** *bufMB*] [**-S** *file*] *SourceVolume DestinationVolume*

DESCRIPTION

sndiskmove is a utility for migrating the contents of a disk to a new disk. This is accomplished by copying the data from the disk and relabeling the destination disk with the name of the source disk. The source disk is also relabeled with its original name with a suffix added (this is ".old" by default but this can be changed with the **-s** option.

WARNING: Using this command incorrectly can cause loss of data. Please ensure that you understand the procedures to safely execute this command completely. Also, this command is not supported for filesystem disks that contain data, it must be used for Exclusive Metadata or Journal disks only.

Attempting to run this command on a disk that is being used on an actively running filesystem will cause data corruption. All referencing filesystems must be stopped and remain stopped during the processing of this command.

If successful, the command will cause the local **fsmpm** process to rescan the disks on the host where the command is executed. If there are any **FSMs** configured on other hosts, it is critical to request a rescan of the disks before reenabling the **FSM** on those servers by invoking

cvadmin -e 'disks refresh'

on those hosts.

OPTIONS

-f Forces the disk data to be moved without confirmation from the user. **WARNING: Use this flag with extreme caution!**

-q Run in quiet mode. This disables the progress display.

-v Causes **sndiskmove** to be verbose.

-s *suffix* Uses the supplied *suffix* to relabel the source disk.

-b *bufsizeMB*
Specifies the buffer size to use for the copying of disk data (in megabytes). The default value is 4MB.

-S *file* Writes status monitoring information in the supplied file. This is used internally by StorNext and the format of this file may change.

NOTES

The source and destination disks must contain EFI labels. VTOC labels are not supported. Also, the sector sizes used by the source and destination disks must be the same.

SEE ALSO

cvadmin(8), **cvlabel**(8)

NAME

sndpscfg − StorNext File System Proxy Server Configuration Utility

SYNOPSIS

sndpscfg -e

sndpscfg -E *FsName*

sndpscfg -a

DESCRIPTION

The StorNext File System (SNFS) **sndpscfg** command is a utility used to generate and modify SNFS Proxy Server configuration files on Linux systems. (NOTE: to view and adjust the Proxy Server settings on Windows systems, use the LAN Client/Gateway tab in the Client Configuration tool instead.)

SYNTAX

The **-e** option is used to edit the default **dpserver** configuration file. If no default **dpserver** file exists, a template file will be generated. The template file contains commented-out entries for each of the network interfaces on the system, and commented-out entries for each of the tunable parameters, specifying default values.

The **-E** *FsName* option is used to edit the file-system-specific **dpserver** configuration file for the specified file system. As with **-e**, a template file will be generated if no file-system-specific **dpserver** file exists.

The **-a** option is used to print a template **dpserver** configuration file to standard output.

LIMITATIONS

Only the **Linux** platform is supported with sndpscfg commands

To view and adjust the Proxy Server settings on Windows systems, user the LAN Client/Gateway table in the Client Configuration tool instead.

FILES

/usr/cvfs/config/dpserver
*/usr/cvfs/config/dpserver.*FsName

SEE ALSO

mount_cvfs(8), **dpserver**(4)

NAME

WebService access to Storage Manager file operations

SYNOPSIS

snretrieve [-h] [-v] [-a] file [file ...]

snstore [-h] [-v] [-a] file [file ...]

snrmdiskcopy [-h] [-v] [-a] file [file ...]

snfileinfo [-h] [-a] file [file ...]

snjobinfo [-h] -m MOUNT job [job ...]

DESCRIPTION

Commands to allow remote client access to Storage Manager file management commands. For managed files allows control of store, retrieve and truncate operations. For file systems where remote web service access has been configured, these commands provide similar functionality to the storage manager commands fsretrieve, fsrmdiskcopy, fsstore and fsfileinfo. Commands run synchronously by default, the command waits for actions to complete and optionally reports on the results using the **-v** option. The **-a** option can be used to make the command asynchronous. In this case one or more job numbers are reported and snjobinfo can be used to later wait for and report the results.

The commands can take files or directories as arguments. If a directory is specified, then its contents are processed recursively. If a mix of files and directories are specified then more than one job will be used to run the commands.

If a file is specified with @ in front of its name, then this file is interpreted as a list of the files to be processed one entry per line. This allows a preselected set of content to be input to the command.

The snjobinfo command requires that a pathname in a filesystem be provided, this is so that the location of the web server can be determined. The snjobinfo command can be used repeatedly to look at the results of past commands.

These commands require python 2.7.9 or later version.

EXIT VALUES

Commands will exit with zero on successful runs and non-zero otherwise.

NAME

snfs.cfg − StorNext File System Configuration File

SYNOPSIS

This page describes the old File System configuration file format, used prior to StorNext 4.0.

A file system name is associated to its configuration file by the file's prefix. For example, if the file system were named **projecta**, then its configuration file would be */usr/cvfs/config/projecta.cfg*. There may be multiple file systems simultaneously mounted, with an FSM program running for each active file system. Configuration files must reside on the same system as the FSM processes that use them.

DEPRECATED

This format has been deprecated in favor of the cfgx format on all platforms except Windows - see **snfs.cfgx**(5) for details on the new format.

SYNTAX

Each configuration file has several section headers and section bodies. A section header is enclosed by square brackets as follows:

```
[ Keyword Name ]
```

A section body is the non-bracketed lines of configuration between section headers.

Every configuration file begins with a **Global** section body; a **Global** section header is implied. In addition to the implicitly named **Global** section, other section-header keyword values are **AutoAffinity**, **NoAffinity**, **DiskType**, **Disk** and **StripeGroup**. Section-header keywords are case insensitive.

Section names consist of case-sensitive letters, numbers, underscores (_), and dashes (-).

Each line of a section body has the following syntax:

```
Keyword Value
```

Section-body keyword names are section dependent as described below.

Section-body values can be a number, name or any combination of characters enclosed by double quotes ("). A number can be in octal, decimal or hexadecimal. Octal numbers are represented by prefixing the number with **0**, and hexadecimal numbers are represented by prefixing the number with **0x**.

A suffix of **m** or **M** indicates a **mega** multiplier, which is 2^20 = 1,048,576. For example the value **3M** results in an integer value of 3,145,728.

Following is the list of case-insensitive multiplier suffixes:

```
Suffix    Name    Multiplier
------    ----    ----------
  K       kilo    1,024
  M       mega    1,048,576
  G       giga    1,073,741,824
  T       tera    1,099,511,627,776
```

In some cases of specifying disk space, a suffix changes the meaning of the parameter. A number alone implies blocks while a number with a suffix implies bytes. Following are the keywords configured in blocks by default, or bytes when a multiplier suffix is used:

```
Keyword
-------
InodeExpandMin
InodeExpandInc
InodeExpandMax
InodeStripeWidth
PerfectFitSize
```

```
StripeAlignSize
StripeBreadth
```

Comments start anywhere a pound sign (**#**) occurs, and continue to the end of that line. The following are valid comment strings:

```
# This is a comment line
#     Text after and including a # sign on any line is ignored.
#               vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
FsBlockSize 4k  # The file system block size is 4096 bytes.
```

SECTIONS SUMMARY

The configuration file has six section types that are delimited by square-bracketed section-header lines as described above. The first section type is the **Global** section, which does not have a section-header line. It occurs only once, at the start of the file, and ends with the first bracketed section-header line. The Global section defines configuration attributes that affect table sizes, file system caches and performance factors for the file system.

The remaining sections are **AutoAffinity**, **NoAffinity**, **DiskTypes**, **Disk** and **StripeGroup**. These may occur any number of times and in any order.

Each **AutoAffinity** section defines a mapping of extensions to the given **Affinity**.

Each **NoAffinity** section defines a mapping of extensions to no affinity, i.e. an affinity value of 0.

Each **DiskTypes** section defines an instance of a valid disk type. This is simply a category of disk device and the disk device's size in sectors.

The **Disk** section describes individual disk devices that are configured in the file system. The name of the disk device must be placed in the volume header label using **cvlabel**(8). This is how the disk devices become associated with an **SNFS** disk name.

The **StripeGroup** section describes a group of **Disk** entities that comprise a stripe group. One or more stripe groups describe the entire file system. Best performance is attained with a minimum of 2 stripe groups per file system with one stripe group used exclusively for metadata/journal and the other for user data. Metadata has an I/O pattern of small random I/O whereas user data is typically of much larger size. Splitting apart metadata and journal into 2 stripe groups is also recommended particularly if latency for file creation, removal and allocation of space is important. A stripe group is the smallest entity that can be manipulated by administration commands, such as **cvadmin**(8). Even though disk devices may be brought up or down, this has the effect of upping or downing the entire stripe group. The **StripeGroup** can be either read or write disabled. The stripe group is still up, but one or more of the related group's disks are disabled. When a stripe group is write disabled, no further allocations are permitted on the group.

NOTE: Several releases ago, there was a section called **MediaType** that defined **Affinity** names before they were used in a **StripeGroup** section. **Affinities** no longer need to be defined separately from the **StripeGroup**, and if upgrading from a file system that used the **MediaType** section, it must be removed or it will cause parser errors.

DEFAULT GLOBAL VALUES

The following defaults, minimums, and maximums are allowed for modifiable global variables. *NOTE:* the Default Value is in effect when the global variable is absent from the configuration file. Graphical user interfaces and command-line tools such as sncfgedit may create configuration files containing globals variables with initial values that differ from the Default Value below.

Variable Name	Default	Min	Max
AffinityPreference	NO	NO	YES
AllocationStrategy	Round		Round, Fill, Balance
AllocSessionReservationSize	1024M	128M	1048576M
BrlResyncTimeOut†	20	0	60
BufferCacheSize	256M	32M	500G

CaseInsensitive	NO	NO	YES
CvRootDir†	"/"	valid dir with < 1024 chars	
Debug	0	0	0xFFFFFFFF
DirWarp°	YES	NO	YES
EnableSpotlight	NO	NO	YES
EventFiles†	YES	NO	YES
EventFileDir†	special		valid dir with < 1024 chars
ExtentCountThreshold	49152	0	0x1FFFC00
FileLocks	NO	NO	YES
ForcePerfectFit†	NO	NO	YES
FsCapacityThreshold	0	0	100
FSMRealtime	NO	NO	YES
FSMMemlock	NO	NO	YES
GlobalShareMode	NO	NO	YES
GlobalSuperUser	NO	NO	YES
HaFsType	HaUnmonitored (values in snfs_config(5))		
InodeCacheSize	128K	4K	512K
InodeDeleteMax	special	10	0xFFFFFFFF
InodeExpandInc°	0	1	32767
InodeExpandMax°	0	1	32768
InodeExpandMin°	0	1	32768
InodeStripeWidth	4096M	0	1099511627776
IoTokens	YES	NO	YES
JournalSize	256M	16M	512M
MaintenanceMode	NO	NO	YES
MaxLogs	4	1	1000
MaxLogSize	16M	1M	1073741824
NamedStreams	NO	NO	YES
OpHangLimitSecs	180	0	0xFFFFFFFF
PerfectFitSize	8	1	32768
Quotas	NO	NO	YES
QuotaHistoryDays	7	0	3650
RemoteNotification†	NO	NO	YES
RenameTracking	NO	NO	YES
ReservedSpace†	YES	NO	YES
MetadataArchive†	NO	NO	YES
MetadataArchiveDir†	special	valid dir with < 1024 chars	
MetadataArchiveSearch†	YES	NO	YES
MetadataArchiveCache†	2GB	1GB	500GB
MetadataArchiveDays†	0	0	366
SecurityModel	legacy	legacy	legacy, acl, unixpermbits
StripeAlignSize	-1	-1	0xFFFFFFFF
TrimOnClose†	0	0	(2^64)-1
UnixDirectoryCreationModeOnWindows	0755	0	0777
UnixFileCreationModeOnWindows	0644	0	0777
UnixIdMapping	none		none, algorithmic, winbind
UnixIdFabricationOnWindows	*	NO	YES
UnixNobodyGidOnWindows	60001	0	0x7FFFFFFF
UnixNobodyUidOnWindows	60001	0	0x7FFFFFFF
UseL2bufferCache	YES	NO	YES
WindowsIdMapping	ldap		ldap, mdc, none
Security Model Variables			
UseActiveDirectorySFU	YES	NO	YES

WindowsSecurity	YES	NO	YES
XSan Only Variables			
EnforceACLs	NO	NO	YES

* -- UnixIdFabricationOnWindows default value is YES on Xsan and NO for all other platforms.

† *NOTE*: Not intended for general use. Only use when recommended by Quantum Support.

° *NOTE:* Deprecated and will no longer be valid in a future release

AUTOAFFINITY SECTION

Following is the format for a **AutoAffinity** section:

[AutoAffinity <Affinity>]

Extension <extension name>
[One Extension per extension in this mapping]

NOAFFINITY SECTION

Following is the format for a **NoAffinity** section:

[NoAffinity]

Extension <extension name>
[One Extension per extension in this mapping]

DISKTYPE SECTION

Following is the format for a **DiskType** section:

[DiskType <name>]

Sectors <sectors_per_disk>

SectorSize <sector_size>

DISK SECTION

The **Disk** section syntax is as follows:

[Disk <name>]

Status <UP | DOWN>

Type <disktype_name>

STRIPEGROUP SECTION

The **StripeGroup** section format is as follows:

[StripeGroup <name>]

Status <UP | DOWN>

Exclusive <Yes | No>

Metadata <Yes | No>

Journal <Yes | No>

Affinity <eight_character_string>

[Zero or more affinity entries are allowed]

Read <Enabled | Disabled>

Write <Enabled | Disabled>

Alloc <Enabled | Disabled>

StripeBreadth <number_of_blocks_per_disk>

MultiPathMethod <Rotate|Static|Sticky|Balance|Cycle>

Node <disk_name> <stripe_group_unit_number>
[One Node per disk in stripe group]

- **Status**
  If **Up**, the stripe group is available, if **Down** it is not.

- **StripeBreadth**
  Describes the number of file system blocks or bytes that are appended to a file before switching to the next disk in the group. When the value is specified without a multiplier suffix, it is a number of file system blocks. When specified with a multiplier, it is bytes.

- **Metadata**
  If **Yes**, this stripe group contains metadata. If **No**, it does not.

- **Journal**
  If **Yes**, this stripe group contains the journal. If **No**, it does not. Only one stripe group may contain a journal per file system.

- **Exclusive**
  When the **Exclusive** variable is set to **YES** on a stripe group that has **Metadata** or **Journal** set to YES, no userdata may reside on that stripe group.

  When the **Exclusive** variable is set to **YES** on a stripe group that does not have either **Metadata** or **Journal** set to **YES**, and does have **Affinity** values declared, only the **Affinities** declared in its **StripeGroup** section are allowed to reside on this stripe group. This may be preferable for high-bandwidth applications. Because **Exclusive** is used in two ways, a stripe group cannot have both exclusive **Affinity** declarations and metadata.

  The **Exclusive** helps to optimize disk striping strategies. For example, in a broadcast video application, an NTSC field is blocked out at 352,256 bytes per field. The optimal and required stripe breadth for a 525 line (NTSC broadcast) stripe group is two fields (a frame), or 43, 16k file system blocks. However, a 625 line (PAL broadcast) stripe group uses 417,792 bytes per field, and the optimal stripe breadth is 51, 16k file system blocks per disk. In order for allocation to work in a mixed mode environment, it is necessary to have one stripe group set up for 525 (NTSC) and another set up for 625 (PAL). See the configuration file example below to see two stripe groups, each configured optimally for NTSC and PAL modes.

- **Affinity**
  In conjunction with the **Exclusive** variable, the **Affinity** variable helps the FSM determine what type of allocation may occur on a stripe group. When **Exclusive** is **YES**, only files with matching **Affinity** values can be allocated. When **Exclusive** is **NO**, it is possible that other file types would be allocated on the stripe group. This could have fragmenting effects over time and eventually cause high-bandwidth performance problems. It is recommended that stripe groups are developed to be specialized for each file type used.

  The **Affinity** value can be any string of 8 characters or less.

- **Rtios**
  The **Rtios** variable defines the maximum number of disk I/O operations per second (IOs/s) available to real-time applications for the stripe group using the **Quality of Service (QoS)** API. If both **Rtios** and **Rtmb** are set to **0** (the default value), **QoS** is disabled. If both **Rtios** and **Rtmb** are defined, the smaller value (after **Rtmb** is converted to IOs/s internally) is used. This value should be obtained by real measurement using IO benchmark tool.

- **RtiosReserve**
  The **RtiosReserve** variable defines the minimum number of disk I/O operations per second reserved for non-realtime applications when realtime operations have been enabled using the **QoS** API. This prevents non-realtime clients from IO starvation. If both **RtiosReserve** and **RtmbReserve** are set to **0** (the default value), the actual bandwidth reserved is 1MB/s (converting to IO/s), at least 1 IO/s is reserved. If both parameters are defined, the smaller value (**RtmbReserve** is converted to IOs/s internally) is used. This value should not be greater than **Rtios** or **Rtmb** (after being converted to IOs/s).

- **Rtmb**
  The **Rtmb** variable defines the maximum number of MBs per second available to real-time applications for the stripe group using the **QoS** API. Internally the bandwidth is converted to IOs/s based on the size of a well-formed I/Os, i.e. the size of a stripe width. If both **Rtios** and **Rtmb** are set to **0** (the default value), **QoS** is disabled. If both are defined, the smaller value (**RtmbReserve** is converted to IOs/s internally) is used. This value should be obtained through real measurement using I/O benchmark tools. Note: since the system uses IOs/s internally to throttle I/Os, it is recommended to specify **Rtmb** only if all I/Os are well formed. Otherwise, the conversion between MB/s and IOs/s using well-formed IOs could lead to unexpected results.

- **RtmbReserve**
  The **RtmbReserve** variable defines the minimum number of MBs per second reserved for non-realtime applications when realtime operations have been enabled using the **QoS** API. This prevents non-realtime clients from IO starvation. Internally this parameter is converted to IOs/s based on the size of a well-formed IO. If both **RtiosReserve** and **RtmbReserve** are set to **0** (the default value), the actual bandwidth reserved is 1MB/s (converting to IOs/s), at least 1 IO/s is reserved. If both parameters are defined, the smaller value (**RtmbReserve** is converted to IOs/s internally) is used. This value should not be greater than the value of **Rtmb**.

- **RtTokenTimeout**
  The **RtTokenTimeout** variable defines the number of seconds for the **FSM** server to wait for clients to respond to a **QoS** token callback before timeout. If this parameter is not set or it is set to 0, the default value is 1.5 (seconds). This value may need to be changed for a SAN that has a mixture of client machine types (Linux, Windows, etc.) that all have different TCP/IP characteristics. Also, large numbers of clients (greater than 32) may also require increasing the value of this parameter.

EXAMPLE CONFIGURATION FILE

The following is an example of a fairly complex SNFS file system that supports multiple broadcast video/audio file formats, seven stripe groups and eighteen disk drives.This file system is set up to support both 525 and 625 real-time broadcast formats.

```
#
# StorNext File System Configuration File Example
#
#
# Names can be of [A-Z][a-Z][0-9] hyphen (-) and a under-bar (_)
#
# Other things, like user defined strings and pathnames must be enclosed
# by double quotes (").
#
# The comment character (#) may start anywhere and persists to the end
```

```
#  of line.
#

# ************************************************************************
# A global section for defining file system-wide parameters.
#
# For Explanations of Values in this file see the following:
#
# UNIX Users:    man snfs_config
#
# Windows Users: Start > Programs >
#                     StorNext File System > Help >
#                         Configuration File Format
#
# ************************************************************************

AffinityPreference      No
AllocationStrategy      Round
BufferCacheSize         256M
DataMigration           No      # SNMS Managed File Systems Only
Debug                   0x0
FileLocks               No
FsBlockSize             4K
GlobalShareMode         No
GlobalSuperUser         Yes     # Set to Yes for SNMS Managed File Systems
InodeCacheSize          128K    # 800-1000 bytes each
IoTokens                Yes     # Data coherency model uses tokens
JournalSize             256M
MaxLogs                 4
MaxLogSize              16M
OpHangLimitSecs         300     # Default is 180 secs
Quotas                  No
QuotaHistoryDays        7
UnixDirectoryCreationModeOnWindows  0755
UnixFileCreationModeOnWindows       0644
UnixIdFabricationOnWindows          No
UnixNobodyGidOnWindows  60001
UnixNobodyUidOnWindows  60001
WindowsSecurity         Yes
SecurityModel           legacy
WindowsIdMapping        ldap
UseActiveDirectorySFU   Yes

[AutoAffinity NTSC]
Extension mov
Extension dpx

[AutoAffinity PALAud]
Extension mp3
Extension WAV

[NoAffinity]
Extension html
Extension txt
```

```
# ************************************************************************
# A disktype section for defining disk hardware parameters.
# ************************************************************************
[DiskType MetaDrive] ##1+1 Raid 1 Mirrored Pair##
Sectors    99999999      ## Sectors Per Disk From Command "cvlabel -l" ##
SectorSize 512

[DiskType JournalDrive] ##1+1 Raid 1 Mirrored Pair##
Sectors    99999999      ## Sectors Per Disk From Command "cvlabel -l" ##
SectorSize 512

[DiskType VideoDrive] ##8+1 Raid 5 LUN for Video##
Sectors    99999999      ## Sectors Per Disk From Command "cvlabel -l" ##
SectorSize 512

[DiskType AudioDrive] ##4+1 Raid 3 LUN for Audio##
Sectors    99999999      ## Sectors Per Disk From Command "cvlabel -l" ##
SectorSize 512

[DiskType DataDrive] ##4+1 Raid 5 LUN for Regular Data##
Sectors    99999999      ## Sectors Per Disk From Command "cvlabel -l" ##
SectorSize 512




# ************************************************************************
# A disk section for defining disks in the hardware configuration.
# ************************************************************************

[Disk CvfsDisk0]
Status UP
Type MetaDrive

[Disk CvfsDisk1]
Status UP
Type JournalDrive

[Disk CvfsDisk2]
Status UP
Type VideoDrive

[Disk CvfsDisk3]
Status UP
Type VideoDrive

[Disk CvfsDisk4]
Status UP
Type VideoDrive

[Disk CvfsDisk5]
Status UP
Type VideoDrive
```

```
[Disk CvfsDisk6]
Status UP
Type VideoDrive

[Disk CvfsDisk7]
Status UP
Type VideoDrive

[Disk CvfsDisk8]
Status UP
Type VideoDrive

[Disk CvfsDisk9]
Status UP
Type VideoDrive

[Disk CvfsDisk10]
Status UP
Type AudioDrive

[Disk CvfsDisk11]
Status UP
Type AudioDrive

[Disk CvfsDisk12]
Status UP
Type AudioDrive

[Disk CvfsDisk13]
Status UP
Type AudioDrive

[Disk CvfsDisk14]
Status UP
Type DataDrive

[Disk CvfsDisk15]
Status UP
Type DataDrive

[Disk CvfsDisk16]
Status UP
Type DataDrive

[Disk CvfsDisk17]
Status UP
Type DataDrive



# ****************************************************************************
# A stripe section for defining stripe groups.
# ****************************************************************************
```

```
[StripeGroup MetaFiles]
Status UP
MetaData Yes
Journal No
Exclusive Yes
Read Enabled
Write Enabled
StripeBreadth 256K
MultiPathMethod Rotate
Node CvfsDisk0 0


[StripeGroup JournFiles]
Status UP
Journal Yes
MetaData No
Exclusive Yes
Read Enabled
Write Enabled
StripeBreadth 256K
MultiPathMethod Rotate
Node CvfsDisk1 0


[StripeGroup NTSCFiles]
Status UP
Exclusive Yes                       ##Exclusive StripeGroup for Video Files Only##
Affinity NTSC                       ##8 character limit##
Read Enabled
Write Enabled
Alloc Enabled
StripeBreadth 688k          ## NTSC frame size
MultiPathMethod Rotate
Node CvfsDisk2 0
Node CvfsDisk3 1
Node CvfsDisk4 2
Node CvfsDisk5 3


[StripeGroup PALFiles]
Status UP
Exclusive Yes                       ##Exclusive StripeGroup for Video Files Only##
Affinity PAL                        ##8 character limit##
Read Enabled
Write Enabled
Alloc Enabled
StripeBreadth 816k      ## PAL frame size
Node CvfsDisk6 4
Node CvfsDisk7 5
Node CvfsDisk8 6
Node CvfsDisk9 7


## CCIR-601 525 Audio is read/written in 65536 byte blocks
```

```
[StripeGroup AudioFiles1]
Status UP
Exclusive Yes                       ##Exclusive StripeGroup for Audio File Only##
Affinity NTSCAud           ##8 character limit##
Read Enabled
Write Enabled
Alloc Enabled
StripeBreadth 64k
MultiPathMethod Rotate
Node CvfsDisk10 0
Node CvfsDisk11 1


## CCIR-601 625 Audio is read/written in 61440 byte blocks
## We put 4 blocks per stripe so that it’s divisible by 16k fs block size
[StripeGroup AudioFiles2]
Status UP
Exclusive Yes                       ##Exclusive StripeGroup for Audio File Only##
Affinity PALAud                     ##8 character limit##
Read Enabled
Write Enabled
Alloc Enabled
StripeBreadth 240k
Node CvfsDisk12 2
Node CvfsDisk13 3


[StripeGroup RegularFiles]
Status UP
Exclusive No                        ##Non-Exclusive StripeGroup for all Files##
Read Enabled
Write Enabled
Alloc Enabled
StripeBreadth 256K
MultiPathMethod Rotate
Node CvfsDisk14 0
Node CvfsDisk15 1
Node CvfsDisk16 2
Node CvfsDisk17 3

#
# End
#
```

FILES

/usr/cvfs/config/.cfg*
/usr/cvfs/data/<file_system_name>*/config_history/*.cfg.*<TIMESTAMP>

SEE ALSO

snfs_config(5), **snfs.cfgx**(5)

NAME

snfs.cfgx − StorNext File System Configuration File

SYNOPSIS

This page describes the XML-format file system configuration file first introduced in StorNext 4.0 (the configDoc element will have a version attribute of "1.0"). It is an XML 1.0 compliant format, and is hierarchical in nature. All elements and attributes are case-sensitive.

See **snfs_config**(5) for details and descriptions of specific fields in this file and for a more general overview of file system configuration.

See **sncfgedit**(8) for the best way to edit a configuration file from the commandline.

A file system name is associated to its configuration file by the file's prefix. For example, if the file system were named **projecta**, then its configuration file would be */usr/cvfs/config/projecta.cfgx*. There may be multiple file systems simultaneously mounted, with an FSM program running for each active file system. Configuration files must reside on the same system as the FSM processes that use them.

ELEMENTS

The following describes the elements in hierarchical depth-first order. See EXAMPLE CONFIGURATION FILE to see all the elements together.

configDoc

The main element of the config is a **configDoc**. This sets up the XML namespace via the *xmlns* attribute and specifies the version of the configuration format via the *version* attribute. The configDoc contains all configuration information for the StorNext File System described by the file.

- *xmlns*

  Setup the xml namespace. If this is set to "snfs", no additional work is required. If it is setup like this:

  ```
  xmlns:snfs="http://www.quantum.com/snfs"
  ```

  each element in the document must be prefixed with "snfs:" to explicitly add them to the snfs namespace.

- *version*

  The format version. Currently must be "1.0".

  Currently, the only element the **configDoc** contains is a single **config** element.

config

Each **config** element contains one **globals** element, one **diskTypes** element, and one **stripeGroups** element. It also contains the following attributes:

- *configVersion*

  A generation number for the configuration file. This typically increases by one every time a changed version of the configuration is written to disk.

- *fsMade*

  Not used in this release

- *requestType*

  Not used in this release

- *name*

  A string denoting the name of the file system

- *fsBlockSize*

  The block size of the file system. As of StorNext 5, the block size is fixed at 4096. A value other than 4096 may be specified for a file system that has been upgraded, in which case the size when the file system was created is used.

- *journalSize*
  The size of the file system's journal. Must be at least 1024 times larger than the fsBlockSize.

globals

The **globals** element contains all global variable elements.

The following table lists the globals, their default values, and the valid range of values for each:

Variable Name	Default	Min	Max
affinityPreference	false	false	true
allocationStrategy	round		round, fill, balance
allocSessionReservationSize	1073741824	134217728	1099511627776
bufferCacheSize	256M	32M	500G
caseInsensitive	false	false	true
cvRootDir†	"/"	valid dir with < 1024 chars	
debug	00000000	00000000	FFFFFFFF
dirWarp°	true	false	true
enableSpotlight	false	false	true
eventFiles†	true	false	true
eventFileDir†	special		valid dir with < 1024 chars
extentCountThreshold	49152	0	0x1FFFC00
fileLockResyncTimeOut†	20	0	60
filelocks	false	false	true
forcePerfectFit†	false	false	true
fsCapacityThreshold	0	0	100
fsmRealTime	false	false	true
fsmMemlock	false	false	true
globalShareMode	false	false	true
globalSuperUser	false	false	true
haFsType	HaUnmonitored (values in snfs_config(5))		
inodeCacheSize	131072	4096	524288
inodeDeleteMax	special	10	0xFFFFFFFF
inodeExpandInc°	0	1	17179869184
inodeExpandMax°	0	1	17179869184
inodeExpandMin°	0	1	17179869184
InodeStripeWidth	4294967296	0	1099511627776
ioTokens	true	false	true
maintenanceMode	false	false	true
maxLogs	4	1	1000
maxLogSize	16777216	1048576	1073741824
namedStreams	false	false	true
opHangLimitSecs	180	0	0xFFFFFFFF
perfectFitSize	32768	4096	17179869184
quotas	false	false	true
quotaHistoryDays	7	0	3650
remoteNotification†	false	false	true
renameTracking	false	false	true
reservedSpace†	true	false	true
metadataArchive†	false	false	true
metadataArchiveDir†	special		valid dir with < 1024 chars
metadataArchiveSearch†	YES	NO	YES
metadataArchiveDays†	0	0	366
metadataArchiveCache†	2GB	1GB	500GB
securityModel	legacy	legacy, acl, unixpermbits	
spotlightUseProxy	false	false	true

storageManager	false	false	true
stripeAlignSize	-1	-1	0xFFFFFFFF
trimOnClose†	0	0	(2^64)-1
unixIdMapping	algorithmic		algorithmic, ldap
unixDirectoryCreationModeOnWindows	0755	0	0777
unixFileCreationModeOnWindows	0644	0	0777
unixIdFabricationOnWindows	*	false	true
unixIdMapping	algorithmic		none, algorithmic, winbind
unixNobodyGidOnWindows	60001	0	0x7FFFFFFF
unixNobodyUidOnWindows	60001	0	0x7FFFFFFF
useL2bufferCache	true	false	true
Security Model Variables			
useActiveDirectorySFU	true	false	true
windowsSecurity	true	false	true
windowsIdMapping	ldap		ldap, mdc, none
XSan-specific Variables			
enforceACLs	false	false	true
spotlightSearchLevel	ReadWrite	FsSearch	ReadWrite

* -- UnixIdFabricationOnWindows default value is YES on Xsan and NO for all other platforms.

† *NOTE*: Not intended for general use. Only use when recommended by Quantum Support.

° *NOTE:* Deprecated and will no longer be valid in a future release

Deprecated global options

The following global option has been deprecated:

AllocSessionReservation <Yes|No>

The **AllocSessionReservation** parameter has been replaced by the **allocSessionReservationSize** parameter. The old parameter is ignored but warnings are issued to indicate what to do in some cases.

autoAffinities

The **autoAffinities** element contains one or more **autoAffinity** and/or **noAffinity** elements.

autoAffinity

The **autoAffinity** element defines a mapping of extensions to the given affinity. It contains one or more **extension** elements and has one attribute:

- *affinity*

  The **Affinity** for this mapping.

extension

Each extension element within the **autoAffinity** element contains a file name extension to map to this **Affinity**. The extension string is case insensitive. The extension string can be empty which means all files not matching any extension in any mapping.

Put together it looks like this:

```
<autoAffinity affinity="Video">
   <extension>dpx</extension>
   <extension>mov</extension>
</autoAffinity>
<autoAffinity affinity="Audio">
   <extension>mp3</extension>
   <extension>wav</extension>
</autoAffinity>
<autoAffinity affinity="Other">
   <extension></extension>
```

```
    </autoAffinity>
```

noAffinity
The **noAffinity** element defines a mapping of extensions to no affinity, i.e. an affinity value of 0. It contains one or more **extension** elements in the same format as **autoAffinity**.

Put together it looks like this:

```
<noAffinity>
   <extension>txt</extension>
   <extension>html</extension>
</noAffinity>
```

diskTypes
The **diskTypes** element contains one or more **diskType** elements.

diskType
The **diskType** element defines a single disk type. It has three attributes:

- *typeName*
  The name by which this disk type will be referenced in subsequent **disk** elements

- *sectors*
  The number of sectors this disk type contains

- *sectorSize*
  The size of each sector for this disk type

  Put together it looks like this:

```
<diskType typeName="MetaDrive" sectors="99999999" sectorSize="512"/>
```

stripeGroups
The **stripeGroups** element contains one or more **stripeGroup** elements.

stripeGroup
The **stripeGroup** element contains a stripe group definition. A stripegroup element contains an optional **affinities** element and one or more **disk** elements. It also has several attributes associated with it:

- *index*
  A non-negative integer denoting the order of the stripe group within the file system.

- *name*
  A string containing the name of the stripe group

- *status*
  up or **down**

- *metadata*
  true if the stripe group contains metadata, **false** otherwise.

- *journal*
  true if the stripe group contains the journal, **false** otherwise. Only one stripe group per file system may contain a journal.

- *userdata*
  true if the stripe group contains userdata, **false** otherwise.

- *stripeBreadth*

The number of bytes to write to each disk in the stripe group before moving to the next disk.

- *multipathMethod*
  One of the following multipath methods: rotate|static|sticky|balance|cycle

- *read*
  true or **false**.

- *write*
  true to enable writes and new allocations to the stripe group, or **false** to disable writes and allocations.

- *alloc*
  true to enable new allocations to the stripe group, or **false** to disable allocations.

- *realTimeIOs*
  Maximum number of I/O operations per second available to real-time applications for the stripe group using the **Quality of Service (QoS)** API.

- *realTimeIOsReserve*
  I/Os that should be reserved for applications not using the QoS API.

- *realTimeMB*
  Maximum number of MBs per second available to real-time applications for the stripe group using the **QoS** API.

- *realTimeMBReserve*
  MBs per second that should be reserved for applications not using the QoS API.

- *realTimeTokenTimeout*
  A non-negative integer indicating the number of seconds for the **FSM** server to wait for clients to respond to a **QoS** token callback before timeout.

  A stripegroup element looks like the following:

```
<stripeGroup index="0" name="MyStripeGroup" status="up" stripeBreadth="4194304" r
  </stripeGroup>
```

affinities
The **affinities** element is only valid in stripe groups that have *userdata*="true". It contains one or more **affinity** elements. It has one attribute.

- *exclusive*
  If *exclusive* is **true**, only files that have the affinities defined for the stripe group associated with them will be allocated in the stripe group. If *exclusive* is **false**, file with the associated affinities will be steered to this stripe group but other files may be allocated in this stripe group as well.

affinity
The **affinity** element defines an affinity to be associated with the stripe group. An affinity is a sequence of up to 8 characters. Up to 512 affinities may be associated with a given stripe group.

For example:

```
<affinity>MyAff1</affinity>
```

disk
A **disk** element defines a disk to use in the stripe group. It contains the following attributes:

- *index*
  Defines the order within the stripe group. Cannot be changed after the file system is made.

- *diskLabel*
  The label of the disk. See **cvlabel**(8) for details on how to create labels.

- *diskType*
  The name of a defined disk type

- *ordinal*
  The global order of the disks in the file system configuration. Cannot be changed after the file system is made.

  For example:

```
<disk index="0" diskLabel="CvfsDisk2" diskType="VideoDrive" ordinal="0"/>
```

EXAMPLE CONFIGURATION FILE

```
<?xml version="1.0" encoding="UTF-8"?>
<configDoc xmlns="http://www.quantum.com/snfs" version="1.0">
  <config configVersion="0" name="example" fsBlockSize="4096" journalSize="268435456">
    <globals>
      <abmFreeLimit>false</abmFreeLimit>
      <affinityPreference>false</affinityPreference>
      <allocationStrategy>round</allocationStrategy>
      <haFsType>HaUnmonitored</haFsType>
      <bufferCacheSize>33554432</bufferCacheSize>
      <cvRootDir>/</cvRootDir>
      <storageManager>false</storageManager>
      <debug>00000000</debug>
      <extentCountThreshold>49152</extentCountThreshold>
      <enableSpotlight>false</enableSpotlight>
      <enforceAcls>false</enforceAcls>
      <fileLocks>false</fileLocks>
      <fileLockResyncTimeOut>20</fileLockResyncTimeOut>
      <forcePerfectFit>false</forcePerfectFit>
      <fsCapacityThreshold>0</fsCapacityThreshold>
      <globalSuperUser>true</globalSuperUser>
      <inodeCacheSize>32768</inodeCacheSize>
      <inodeExpandMin>0</inodeExpandMin>
      <inodeExpandInc>0</inodeExpandInc>
      <inodeExpandMax>0</inodeExpandMax>
      <inodeDeleteMax>0</inodeDeleteMax>
      <inodeStripeWidth>4294967296</inodeStripeWidth>
      <IoTokens>true</IoTokens>
      <maxLogs>4</maxLogs>
      <remoteNotification>false</remoteNotification>
      <fsmRealTime>false</fsmRealTime>
      <fsmMemLocked>false</fsmMemLocked>
      <opHangLimitSecs>300</opHangLimitSecs>
      <perfectFitSize>131072</perfectFitSize>
      <quotas>false</quotas>
      <quotaHistoryDays>7</quotaHistoryDays>
      <renameTracking>false</renameTracking>
      <metadataArchive>false</metadataArchive>
      <metadataArchiveDir></metadataArchiveDir>
```

```
    <metadataArchiveDays>0</metadataArchiveDays>
    <spotlightUseProxy>false</spotlightUseProxy>
    <stripeAlignSize>-1</stripeAlignSize>
    <trimOnClose>0</trimOnClose>
    <useL2BufferCache>true</useL2BufferCache>
    <unixDirectoryCreationModeOnWindows>644</unixDirectoryCreationModeOnWindows>
    <unixIdFabricationOnWindows>false</unixIdFabricationOnWindows>
    <unixFileCreationModeOnWindows>755</unixFileCreationModeOnWindows>
    <unixNobodyUidOnWindows>60001</unixNobodyUidOnWindows>
    <unixNobodyGidOnWindows>60001</unixNobodyGidOnWindows>
    <windowsSecurity>true</windowsSecurity>
    <securityModel>legacy</securityModel>
    <windowsIdMapping>ldap</windowsIdMapping>
    <globalShareMode>false</globalShareMode>
    <useActiveDirectorySFU>true</useActiveDirectorySFU>
    <eventFiles>true</eventFiles>
    <eventFileDir></eventFileDir>
    <allocSessionReservationSize>1073741824</allocSessionReservationSize>
  </globals>
  <autoAffinities>
   <autoAffinity affinity="Video">
      <extension>dpx</extension>
      <extension>mov</extension>
   </autoAffinity>
   <autoAffinity affinity="Audio">
      <extension>mp3</extension>
      <extension>wav</extension>
   </autoAffinity>
   <noAffinity>
      <extension>txt</extension>
      <extension>html</extension>
   </noAffinity>
  </autoAffinities>
  <diskTypes>
    <diskType typeName="MetaDrive" sectors="99999999" sectorSize="512"/>
    <diskType typeName="JournalDrive" sectors="99999999" sectorSize="512"/>
    <diskType typeName="VideoDrive" sectors="99999999" sectorSize="512"/>
    <diskType typeName="AudioDrive" sectors="99999999" sectorSize="512"/>
    <diskType typeName="DataDrive" sectors="99999999" sectorSize="512"/>
  </diskTypes>
  <stripeGroups>
    <stripeGroup index="0" name="MetaFiles" status="up" stripeBreadth="262144" read="tru
      <disk index="0" diskLabel="CvfsDisk0" diskType="MetaDrive" ordinal="0"/>
    </stripeGroup>
    <stripeGroup index="1" name="JournFiles" status="up" stripeBreadth="262144" read="tr
      <disk index="0" diskLabel="CvfsDisk1" diskType="JournalDrive" ordinal="1"/>
    </stripeGroup>
    <stripeGroup index="2" name="VideoFiles" status="up" stripeBreadth="4194304" read="t
      <affinities exclusive="true">
        <affinity>Video</affinity>
      </affinities>
      <disk index="0" diskLabel="CvfsDisk2" diskType="VideoDrive" ordinal="2"/>
      <disk index="1" diskLabel="CvfsDisk3" diskType="VideoDrive" ordinal="3"/>
      <disk index="2" diskLabel="CvfsDisk4" diskType="VideoDrive" ordinal="4"/>
```

```
        <disk index="3" diskLabel="CvfsDisk5" diskType="VideoDrive" ordinal="5"/>
        <disk index="4" diskLabel="CvfsDisk6" diskType="VideoDrive" ordinal="6"/>
        <disk index="5" diskLabel="CvfsDisk7" diskType="VideoDrive" ordinal="7"/>
        <disk index="6" diskLabel="CvfsDisk8" diskType="VideoDrive" ordinal="8"/>
        <disk index="7" diskLabel="CvfsDisk9" diskType="VideoDrive" ordinal="9"/>
      </stripeGroup>
      <stripeGroup index="3" name="AudioFiles" status="up" stripeBreadth="1048576" read="t
        <affinities exclusive="true">
          <affinity>Audio</affinity>
        </affinities>
        <disk index="0" diskLabel="CvfsDisk10" diskType="AudioDrive" ordinal="10"/>
        <disk index="1" diskLabel="CvfsDisk11" diskType="AudioDrive" ordinal="11"/>
        <disk index="2" diskLabel="CvfsDisk12" diskType="AudioDrive" ordinal="12"/>
        <disk index="3" diskLabel="CvfsDisk13" diskType="AudioDrive" ordinal="13"/>
      </stripeGroup>
      <stripeGroup index="4" name="RegularFiles" status="up" stripeBreadth="262144" read="
        <disk index="0" diskLabel="CvfsDisk14" diskType="DataDrive" ordinal="14"/>
        <disk index="1" diskLabel="CvfsDisk15" diskType="DataDrive" ordinal="15"/>
        <disk index="2" diskLabel="CvfsDisk16" diskType="DataDrive" ordinal="16"/>
        <disk index="3" diskLabel="CvfsDisk17" diskType="DataDrive" ordinal="17"/>
      </stripeGroup>
    </stripeGroups>
  </config>
</configDoc>
```

FILES

/usr/cvfs/config/.cfgx*
/usr/cvfs/examples/example.cfgx
/usr/cvfs/data/<file_system_name>*/config_history/*.cfgx.*<TIMESTAMP>

SEE ALSO

snfs_config(5), **snfs.cfg**(5)

NAME

snfsdefrag − StorNext File System File Defrag Utility

SYNOPSIS

snfsdefrag [**-ADdFPqsv**] [**-G** *group*] [**-K** *key]* [**-k** *key*] [**-g** *group*] [**-m** *count*] [**-r**] [**-S** *file*] *Target* [*Target*...]

snfsdefrag -e [**-v**] [**-b**] [**-F**] [**-G** *group*] [**-K** *key]* [**-r**] [**-t**] [**-L**] [**-S** *file*] *Target* [*Target*...]

snfsdefrag -E [**-v**] [**-b**] [**-F**] [**-G** *group*] [**-K** *key]* [**-r**] [**-t**] [**-L**] [**-S** *file*] *Target* [*Target*...]

snfsdefrag -c [**-v**] [**-F**] [**-G** *group*] [**-K** *key*] [**-r**] [**-t**] [**-T**] [**-S** *file*] [**-A**] *Target* [*Target*...]

snfsdefrag -p [**-DPqv**] [**-F**] [**-G** *group*] [**-K** *key]* [**-r**] [**-S** *file*] [**-A**] *Target* [*Target*...]

snfsdefrag -l [**-Dv**] [**-F**] [**-G** *group*] [**-K** *key*] [**-m** *count*] [**-r**] [**-S** *file*] [**-A**] *Target* [*Target*...]

DESCRIPTION

snfsdefrag is a utility for defragmenting files on a StorNext file system by relocating the data in a file to a smaller set of extents. Reducing the number of extents in a file improves performance by minimizing disk head movement when performing I/O. In addition, with fewer extents, StorNext File System Manager (FSM) overhead is reduced.

snfsdefrag can be used to migrate files off of an existing stripe group and on to other stripe groups by using the **-G** option and setting the **-m** option to 0. If affinities are associated with a file that is being defragmented, new extents are created using the existing file affinity, unless being overridden by the **-k** option. If the **-k** option is specified, the files are moved to a stripe group with the specified affinity. Without **-k**, files are moved to any available stripe group. This migration capability can be especially useful when a stripe group is going out of service. See the use of the **-G** option in the EXAMPLES section below.

In addition to defragmenting and migrating files, **snfsdefrag** can be used to list the extents in a file (see the **-e** option) or to prune away unused space that has been preallocated for the file (see the **-p** option).

Note: Free space in a stripe group can also become fragmented. To address this issue, see the **sgdefrag** command.

Note: The **snfsdefrag** utility is no longer the preferred tool to prepare a stripe group for retirement. Use the **sgoffload** utility instead.

OPTIONS

-A Do not attempt to temporarily stop I/O on an open file by setting the administrative lock on the file.

-b Show extent size in blocks instead of kilobytes. Only useful with the **-e** and **-E** (list extents) options.

-c This option causes **snfsdefrag** to just display an extent count instead of defragmenting files. See also the **-t** and **-T** options.

-D Turns on debug messages.

-d Causes **snfsdefrag** to operate on files containing extents that have depths that are different than the current depth for the extent's stripe group. This option is useful for reclaiming disk space that has become "shadowed" after cvupdatefs has been run for stripe group expansion. Note that when **-d** is used, a file may be defragmented due to the stripe depth in one or more of its extents OR due to the file's extent count.

-e This option causes **snfsdefrag** to not actually attempt the defragmentation, but instead report the list of extents contained in the file. The extent information includes the starting file relative offset, starting and ending stripe group block addresses, the size of the extent, the depth of the extent, and the stripe group number. See also the **-t** option.

-E This option has the same effect as the **−e** option except that file relative offsets and starting and ending stripe group block addresses that are stripe-aligned are highlighted with an asterisk (*). Also, starting stripe group addresses that are equally misaligned with the file relative offset are

highlighted with a plus sign (+). See also the **-t** option.

-F This option causes **snfsdefrag** to skip resource forks for file systems on which the namedStreams option is enabled. This option has no effect on Windows.

-G *stripegroup*
This option causes **snfsdefrag** to only operate on files having at least one extent in *stripegroup*, which is the stripe group index. Use **"sgmanage --list"** to see the stripe group index. Note that multiple **-G** options can be specified to match files with an extent in at least one of the specified stripe groups.

-K *key* This option causes **snfsdefrag** to only operate on source files that have the supplied affinity *key*. If *key* is preceded by '!' then **snfsdefrag** will only operate on source files that do **not** have the affinity *key*. See EXAMPLES below.

-k *key* Forces the new extent for the file to be created on the stripe group specified by *key*. This option has the side effect of changing or creating the affinity on the affected files. If this is not desired, the **cvaffinity** command can be used to change or delete the affinity or the *-g* option can used instead.

-g *stripegroup*
Places the new extent on the stripe group corresponding to the specified index *stripegroup*. Unlike the *key* option, the file's affinity is not affected. Use **"sgmanage --list"** to see the stripe group index.

-l This option causes **snfsdefrag** to just list candidate files.

-L When used with the **-e** or **-E** option, this option causes **snfsdefrag** to also print out the physical location of each extent on disk.

-m *count*
This option tells **snfsdefrag** to only operate on files containing more than *count* extents. By default, the value of *count* is 1. A value of zero can be specified to operate on all files with at least one extent. This is useful for moving files off a stripe group.

-p Causes **snfsdefrag** to perform a prune operation instead of defragmenting the file. During a prune operation, blocks beyond EOF that have been preallocated either explicitly or as part of inode expansion are freed, thereby reducing disk usage. Files are otherwise unmodified. Note: While prune operations reclaim unused disk space, performing them regularly can lead to free space fragmentation.

-P Lists skipped files.

-q Causes **snfsdefrag** to be quiet.

-r This option instructs **snfsdefrag** to recurse through the Target and attempt to defragment each fragmented file that it finds. If Target is not specified, the current directory is assumed.

-s Causes snfsdefrag to perform allocations that are block-aligned. This can help performance in situations where the I/O size perfectly spans the width of the stripe group's disks.

-S *file* Writes status monitoring information in the supplied file. This is used internally by StorNext and the format of this file may change.

-t This option adds totals to the output of the **−c**, **−e**, or **−E** options. Output at the end indicates how many regular files were visited, how many total extents were found from all files, and the average # of extents per file. Also shown are the number of files with one extent, the number of files with more than one extent, and the largest number of extents in a single file.

-T This option acts like -t, except that with -c, only the summary output is presented. No information is provided for individual files.

-v Causes **snfsdefrag** to be verbose.

EXAMPLES

Count the extents in the file foo.

```
rock% snfsdefrag -c foo
```

Starting in directory, dir1, recursively count all the files and their extents and then print the grand total and average number of extents per file.

```
rock% snfsdefrag -r -c -t dir1
```

List the extents in the file foo.

```
rock% snfsdefrag -e foo
```

Defragment the file foo.

```
rock% snfsdefrag foo
```

Defragment the file foo if it contains more than 2 extents. Otherwise, do nothing.

```
rock% snfsdefrag -m 2 foo
```

Traverse the directory abc and its sub-directories and defragment every file found containing more than one extent.

```
rock% snfsdefrag -r abc
```

Traverse the directory abc and its sub-directories and defragment every file found having one or more extents whose depth differs from the current depth of extent's stripe group OR having more than one extent.

```
rock% snfsdefrag -rd abc
```

Traverse the directory abc and its sub-directories and only defragment files having one or more extents whose depth differs from the current depth of extent's stripe group. This situation would arise after cvupdatefs has been used to expand the depth of a stripe group. The high value for -m ensures that only extents with different depth values are defragmented.

```
rock% snfsdefrag -m 9999999999 -rd abc
```

Traverse the directory abc and recover unused preallocated disk space in every file visited.

```
rock% snfsdefrag -rp abc
```

Force the file foo to be relocated to the stripe group with the affinity key "fast"

```
rock% snfsdefrag -k fast -m 0 foo
```

If the file foo has the affinity **fast**, then move its data to a stripe group with the affinity **slow**.

```
rock% snfsdefrag -K fast -k slow -m 0 foo
```

If the file foo does NOT have the affinity **slow**, then move its data to a stripe group with the affinity **slow**.

```
rock% snfsdefrag -K '!slow' -k slow -m 0 foo
```

Traverse the directory abc and migrate any files containing at least one extent in stripe group 2 to any non-exclusive stripe group.

```
rock% snfsdefrag -r -G 2 -m 0 abc
```

Traverse the directory abc and migrate any files containing at least one extent in stripe group 2 to stripe groups with the affinity **slow**. It is advised that allocations to the source stripe group be disabled before

running the following command, if you wish to retire the source stripe group. On systems with Linux MDCs, use **sgmanage** to disable allocations. On Windows MDCs, edit the config file, insert **"Alloc Disabled"** in the stripe group section, and restart the FSM.

```
rock% snfsdefrag -r -G 2 -k slow -m 0 abc
```

Traverse the directory abc list any files that have the affinity **fast** and having at least one extent in stripe group 2. It is advised that allocations to the source stripe group be disabled before running the following command, if you wish to retire the source stripe group. On systems with Linux MDCs, use **sgmanage** to disable allocations. On Windows MDCs, edit the config file, insert **"Alloc Disabled"** in the stripe group section, and restart the FSM.

```
rock% snfsdefrag -r -G 2 -k fast -l -m 0 abc
```

NOTES

If **snfsdefrag** is run on a Windows client, the user must have read and write access to the file. If **snfsdefrag** is run on a Unix client, only the owner of a file or superuser is allowed to defragment a file. (To act as superuser on a StorNext file system, in addition to becoming the user **root**, the configuration option GlobalSuperUser must be enabled. See **snfs_config(5)** for more information.)

snfsdefrag attempts to set an administrative lock on a file before attempting the defrag operation, unless overridden by the **-A** option. This will stop I/O related operations on an open file and allow the defrag operation to proceed. When the defrag is complete, the client will refresh its view of the file such that an application will not be aware that the file's physical location on disk may have changed. If the administrative lock cannot be obtained and the file is open, **snfsdefrag** will skip the file.

snfsdefrag will not operate on files that have been modified in the past 10 seconds and files with modification times in the future. If a file is modified while defragmentation is in progress, **snfsdefrag** will abort and the file will be skipped.

snfsdefrag skips special files and files containing holes.

snfsdefrag does not follow symbolic links.

When operating on a file marked for PerfectFit allocations, **snfsdefrag** will "do the right thing" and preserve the PerfectFit attribute.

While performing defragmentation, **snfsdefrag** creates a temporary file named *TargetFile*__**defragtmp**. If the command is interrupted, **snfsdefrag** will attempt to remove this file. However, if **snfsdefrag** is killed or a power failure occurs, the temporary file may be left behind. If snfsdefrag is subsequently re-run and attempts defragmentation, it will clean up any stale temporary files encountered. But if snfsdefrag is not run again, it will be necessary to find and remove the temporary file as it will continue to consume space. Note that user files having the **__defragtmp** extension should not be created if **snfsdefrag** is to be run.

snfsdefrag will fail if it cannot locate a set of extents that would reduce the current extent count on a file.

When files being defragmented reside in a managed file system with stub files enabled and CLASS_STUB_READ_AHEAD is set in the fs_sysparams file, the operation could cause file retrieval.

By default, when using the **-r option, snfsdefrag** will sort directory entries before operating on them unless the size of the directory exceeds 1GiB. This threshold can be adjusted using the environment variable DEFRAG_MAX_DIR_SORT_SIZE.

On Linux and macOS platforms, when a file contains a resource fork and resides on a file system where the namedStreams feature is enabled, by default, **snfsdefrag** will attempt to operate on the resource fork as well as the main file. When this occurs, the resource fork will be displayed having the same name as the original file with the "/..namedfork/rsrc" suffix. For example, if the original file has the name "/stornext/snfs1/myfile", **snfsdefrag** will show "/stornext/snfs1/myfile/..namedfork/rsrc" to represent the resource fork "named stream." The -F option can be used to prevent **snfsdefrag** from operating on these named streams.

ADVANCED FRAGMENTATION ANALYSIS

There are two major types of fragmentation to note: file fragmentation and free space fragmentation. File fragmentation is measured by the number of file extents used to store a file. A file extent is a contiguous allocation unit within a file. When a large enough contiguous space cannot be found to allocate to a file, multiple smaller file extents are created. Each extent represents a different physical spot in a stripe group. Requiring multiple extents to address file data impacts performance in a number of ways. First, the file system must do more work looking up locations for a file's data. Also, having file data spread across many different locations in the file system requires the storage hardware to do more work while reading a file. On a disk there will be increased head movements, as the drive seeks around to read in each data extent. Many disks also attempt to optimize I/O performance, for example, by attempting to predict upcoming read locations. When a file's data is contiguous these optimizations work well. However, with a fragmented file the drive optimizations are not nearly as efficient.

A file's fragmentation should be viewed more as a percentage than as a hard number. While it's true that a file of nearly any size with 50000 fragments is extremely fragmented and should be defragmented, a file that has 500 fragments that are mostly one or two file system blocks (4096 bytes) in length is also very fragmented. Keeping files to under 10% fragmentation is the ideal, and how close you come to that ideal is a compromise based on real-world factors (file system use, file sizes and their life span, opportunities to run **snfsdefrag**, etc.).

In an attempt to reduce fragmentation (file and free space), Administrators can try using the Allocation Session Reservation feature. This feature is managed using the GUI or by modifying the **allocSessionReservationSize** parameter, see **snfs_config**(5). See also the StorNext Tuning Guide.

Some common causes of fragmentation are having very full stripe groups (possibly because of affinities), a file system that has a lot of fragmented free space (deleting a fragmented file produces fragmented free space), heavy use of CIFS or NFS which typically use out-of-order allocations resulting in unoptimized (uncoalesced) allocations, or an application that writes files in a random order.

snfsdefrag is designed to detect files which contain file fragmentation and coalesce that data onto a minimal number of file extents. The efficiency of **snfsdefrag** is dependent on the state of the file system's free data blocks, or free space.

The second type of fragmentation is free space fragmentation. The file system's free space is the pool of unallocated data blocks. Space allocation for new files, as well as allocations for extending existing files, comes from the file system's free space. Free space fragmentation is measured by the number of fragments of contiguous free blocks. Fragmentation in the file system's free space affects the file system's ability to allocate large extents. A file can only have an extent as large as the largest contiguous block of free space. Thus free space fragmentation can lead to file fragmentation in larger files. As **snfsdefrag** processes fragmented files it attempts to use large enough free space fragments to create a new defragmented file space. If free space is too fragmented **snfsdefrag** may not be able to allocate a large enough extent for the file's data. In the case that **snfsdefrag** must use multiple extents in the defragmented file, it will only proceed if the processed file will have fewer extents than the original. Otherwise **snfsdefrag** will abort that file's defrag process and move on to remaining defrag requests.

FRAGMENTATION ANALYSIS EXAMPLES

The following examples include reporting from **snfsdefrag** as well as **cvfsck**. Some examples require additional tools such as **awk** and **sort**.

Reporting a specific file's fragmentation (extent count).

```
# snfsdefrag -c <filename>
```

Report all files, their extents, the total # of files and extents, and the average number of extents per files. Beware that this command walks the entire file system so it can take a while and cause the performance of applications to degrade while running.

```
# snfsdefrag -r -c -t <mount point>
```

The following command will create a report showing each file's path, followed by extent count, with the re-

port sorted by extent count. Files with the greatest number of extents will show up at the top of the list.

Replace <fsname> in the following example with the name of your StorNext file system. The report is written to stdout and should be redirected to a file.

```
# cvfsck -x <fsname> | awk -F, '{if (NF == 14) \
  print($6", "$7)}' | sort -uk1 -t, | sort -nrk2 -t,
```

This next command will display all files with at least 10 extents and with a size of at least 1MB. Replace <fsname> in the following example with the name of your StorNext file system. The report is written to stdout and can be redirected to a file.

```
# echo "#extents  file size  av. extent size  filename"; \
  cvfsck -r <fsname> | awk '{if (NF == 8 && $03 > 1048576 && \
  $05 > 10) printf("%8d %10d %16d %10s\n", $5, $3, $03/$05, $8)}' \
  | sort -nr
```

The next command displays a report of free space fragmentation. This allows an administrator to see if free space fragmentation may affect future allocation fragmentation. See **cvfsck**(8) man page for description of report output.

```
# cvfsck -a -t -f <fsname>
```

The fragmentation detected RAS warning message may sometimes refer to an inode number instead of a file name. To find the file name associated with the inode number on non-Windows clients, fill the file system mount point and the decimal inum from the RAS message into the following find command. The file name can then be used to defragment the file. There may be more than one file that matches the 32-bit inode number.

```
# find <mount_point> -inum <decimal_inum>

# snfsdefrag <filename>
```

For Windows clients:

Using a DOS shell, CD to the directory containing the StorNext binaries and run the cvstat command as follows: The <fname> parameter is the drive letter:/mount point and the <inum> parameter has either the decimal or hexadecimal 64-bit inode number from the RAS message. For example:

```
c:\> cd c:\Program Files\StorNext\bin

c:\> cvstat fname=j:\ inum=0x1c0000004183da
```

FILES

/usr/cvfs/config/.cfgx*

SEE ALSO

cvfsck(8), **cvcp**(1), **cvmkfile**(1), **snfs_config**(5), **snfs.cfgx**(5), **snfs.cfg**(5), **cvaffinity**(1), **sgdefrag**(8), **sgoffload**(8), **sgmanage**(8)

NAME

SNFSNAMESCANNER − Scan for unusual names in a StorNext File System

SYNOPSIS

/usr/cvfs/lib/snfsnamescanner -ipu [**-L** 1**]** *pathname* |-h

DESCRIPTION

snfsnamescanner scans the StorNext file system starting at the specified *pathname* for file and directory names that are:

—

Illegal Windows names. Names that do not conform to the Windows naming convention.

—

Names with Unicode Private User Area (PUA) code points. Illegal Windows names that are created by Mac and Linux SMB clients.

—

Names with invalid UTF-8 values. Names created by StorNext Linux and Mac clients using invalid UTF-8 hex sequences.

If **snfsnamescanner** finds a name with one of the specified problems, it creates a script which can be used to change the name of the file.

pathname is the path to the root directory of a mount StorNext file system.

Snfsnamescanner can only be run on a Linux system. You may need to become user **root** to access all the files and directories in the file system. Run **snfsnamescanner** on the active MDC for the fastest scan times.

OPTIONS

-i Scan the specified file system for illegal Windows file names. The file names with illegal file names are placed in the file *illegalFileNames.sh*. This file can be used for changing the file name to a legal Windows file name.

-p Scan the specified file system for file names with PUA code points in the name. The file names with PUA code points are put in the file *puaFileNames.sh*. This file can be used for changing the file name to an equivalent name with ASCII characters.

-u Scan the specified file system for file names with invalid UTF-8 values in the name. The file names with invalid UTF-8 values are put in the file *utf8FileNames.sh*.

−L 1 Convert Latin-1 characters to UTF-8. Use in conjunction with the **-u** option. Treat any invalid UTF-8 byte sequence as a Latin-1 character. Translate each Latin-1 character to a valid UTF-8 sequence. Use this option only if you are sure the character encoding is Latin-1.

-h Display help

You can specify one or more of the -i, -p or -u options at the same time. Specifying more than one option is more efficient than running **snfsnamescanner** multiple times.

Examples:

To scan for illegal Windows file names:

snfsnamescanner -i /stornext/snfs1

The list of illegal file names is written to *./illegalFileNames.sh*. Each line in *illlegalFileNames.sh* will have identical source and target file names. You will have to decide on how to change the target file name to a legal Windows file name.

To scan for PUA code points in file system names:

snfsnamescanner -p /stornext/snfs1

The list of file names with PUA code points is written to *./puaFileNames.sh*. This script can be used without modification. Run this script on the same system it was created since it uses the mount point to the file

system. This script will replace each PUA code point in a file or directory name with the equivalent PUA code point ASCII character.

To scan for file names with invalid UTF-8 values:

snfsnamescanner -u /stornext/snfs1

The list of file names with invalid UTF-8 values is written to *./utf8FileNames.sh*. If the -L option is not used, each line in *utf8FileNames.sh* will have identical source and target file names. You will have to decide on how to change the invalid UTF-8 values to valid UTF-8 values.

The target names in the *illlegalFileNames.sh* file, and possibly the *utf8FileNames.sh* file, will need to be changed before running the script. There are many ways to do this. Here is an example on how to use the Linux sed(1) command to change the target file names in the *illlegalFileNames.sh* file.

To change the target file names in the *./illegalFileNames.sh* file, run the following command.

```
sed -i -f sedscript ./illegalFileNames.sh
```

Where the contents of the sedscript should looks something like this:

```
s-\(\t'/stornext/snfs1/.*\)<-\1#-
s-\(\t'/stornext/snfs1/.*\)>-\1@-
s-\(\t'/stornext/snfs1/.*\):-\1!-
s-\(\t'/stornext/snfs1/.*\)"-\1%-
s-\(\t'/stornext/snfs1/.*\)|-\1+-
s-\(\t'/stornext/snfs1/.*\)?-\1~-
s-\(\t'/stornext/snfs1/.*\)\*-\1A-
s-\(\t'/stornext/snfs1/.*\) '$-\1_'-
s-\(\t'/stornext/snfs1/.*\)\.'$-\1p'-
```

Change the string "/stornext/snfs1" to match the mount point in the *./illegalFileNames.sh* file. The substitution characters are just an example. You can change the sed substitution characters to any set of characters you like, although it would be best if each substitution is unique. You may need to run this sed script multiple times. Each pass will only change a single occurance of the character on a line. You may want to grep the *./illegalFileNames.sh* file to see which Windows reserved characters are present in the file and modify the sed script accordingly.

FILES

/usr/cvfs/lib/snfsnamescanner
./illegalFileNames.sh
./puaFileNames.sh
./utf8FileNames.sh

SEE ALSO

For more information about illegal Windows file name and Unicode Private User Area code points, see the "StorNext File Name Considerations" section in the StorNext User's Guide.

NAME

snfs access − The StorNext Access Control Feature

DESCRIPTION

The StorNext Access Control is an optional feature that provides a method of controlling which clients have access to a StorNext file system. This feature is similar to the StorNext nss_cctl(4) mechanism but has additional capabilities such as specifying which directories a client has access to and the time of day or days of the week access is granted.

The **access.json(5)** file is used to configure the Access Control feature. However, this file should not be edited directly. Instead, the **snaccess(8)** command should be used. There is only one **access.json(5)** file per MDC which controls access to all file systems running on the system and rules can target one or multiple file systems. Once created, the **access.json(5)** file is immediately consumed by a daemon running in each of the **FSM** processes.

Depending on the rules, at a particular time, the FSM will view each client as being in one of the following states:

fully allowed
: The client can mount the file system and access any part of it.

fully denied
: The client is not allowed to mount the file system. If the client already has the file system mounted, operations fail with EPERM.

directory restricted
: The client can mount the file system. However, the set of fully accessible directories is limited. If rules specify a client does not have access to a directory, only traversal and listing is allowed. Namespace and I/O operations fail with EPERM.

Clients may transition between states. This can occur when changes are made to the **access.json(5)** file. But this can also happen because a rule is specified to run only for a certain time range during the week and that range has just begun or finished. For some state transitions, the FSM will force the client to invalidate its caches to ensure the client loses access where required. For clients that are aware of the Access Control feature, this invalidation is transparent. However, when older, backrev clients have access changed or revoked, the FSM may disconnect the client to force its cache to be invalidated. In such cases, the following message may be seen in the **cvlog** file:

> [0226 10:00:19] 0x7fb39ae1a700 (Warning) SNConnect: SNFS Client [10.65.190.196] ip [10.65.190.196] disconnected unexpectedly from file system [snfs1], reason: [disconnecting to force client cache refresh]

NOTES

Care should be taken when restricting access, especially when using the scheduling aspect of the feature. No warnings or "grace periods" are provided and if a client suddenly is denied access, a user or process can fail to save in-progress work.

In some cases, if a client already has acquired certains kinds of access to an object, revoking it may not be possible. For example, if a process on a client succesfully opens a file and has taken posession of a byte range lock and is subsequently denied access, this lock will continue to be held until the process explictly unlocks the file, the file is closed, or StorNext is stopped on the client. Also, if an executable is already running when access is revoked, the process will continue to run.

If a process successfully performs a memory mapping operation on a file and access is later revoked, dirty pages may fail to be flushed. This behavior is expected, but may not be noticeable, depending on the error checking performed by the applications. For example, if a file is opened on Windows using "notepad" and access is later revoked to the client, performing a "Save" operation will appear to complete, but the file will not contain updates.

SEE ALSO

snaccess(8), **access.json**(5), **nss_cctl**(4), **fsm**(8)

NAME

snfs_config − StorNext File System Configuration File

SYNOPSIS

/usr/cvfs/config/*.cfgx

DESCRIPTION

The **StorNext File System** (**SNFS**) configuration file describes to the **File System Manager** (**FSM**) the physical and logical layout of an individual file system.

FORMAT OPTIONS

The StorNext File System uses the XML format for the configuration file (see **snfs.cfgx.5**). This is supported on linux MDCs and is required when using the Storage Manager web-based GUI. If the GUI is not used or not available, the **sncfgedit(8)** utility should be used to create or change the XML configuration file.

The old non-XML format (see **snfs.cfg.5**) used in previous versions of StorNext is required on Windows MDCs and is valid on linux MDCs, but the Storage Manager GUI will not recognize it.

Linux MDCs will automatically have their file system configuration files converted to the XML format on upgrade, if necessary. Old config files will be retained in the */usr/cvfs/data/*<file_system_name>*/config_history* directory.

When a file system system is created, the configuration file is stored in a compressed format in the metadata. Some StorNext components validate that if the configuration file has changed, it is still valid for the operation of that component. The components that do this are: **fsm(8)**, **cvupdatefs(1)**, and **cvfsck(1)**. If the configuration is invalid, the component terminates. If the configuration has changed and is valid, the old configuration is saved in */usr/cvfs/data/*<file_system_name>*/config_history/*.cfgx.<TIMESTAMP> and the new one replaces the old one in metadata.

This manpage seeks to describe the configuration file in general. Format specific information can be found in **snfs.cfgx.5** and **snfs.cfg.5**.

GLOBAL VARIABLES

The file system configuration has several global variables that affect the size, function and performance of the **StorNext File System Manager** (**FSM**). (The **FSM** is the controlling program that tracks file allocation and consistency across the multiple clients that have access to the file system via a Storage Area Network.) The following global variables can be modified.

- XML: **affinityPreference** <true/false>

  Old: **AffinityPreference** <Yes|No>

  The **AffinityPreference** variable instructs the FSM how to allocate space to a file with an **Affinity** in low space conditions. If space cannot be allocated on a stripe group with a matching **Affinity**, the system normally fails with ENOSPC. This occurs even if the file system has remaining space that could satisfy the allocation request. If this variable is set to true (Yes), instead of returning ENOSPC, the system attempts to allocate space on another stripe group with an **Affinity** of 0.

  With this preference mechanism, the file's **Affinity** is not changed so a subsequent allocation request will still try to use the original **Affinity** before retrying with an **Affinity** of 0.

  The default value of false (No) retains the behavior of returning ENOSPC instead of retrying the allocation request.

- XML: **allocationStrategy** <strategy>

  Old: **AllocationStrategy** <strategy>

  The **AllocationStrategy** variable selects a method for allocating new disk file blocks in the file system. There are three methods supported: **Round**, **Balance**, and **Fill**. These methods specify how, for each file, the allocator chooses an initial stripe group to allocate blocks from, and how the allocator chooses a new stripe group when it cannot honor an allocation request from a file's current stripe group.

  The default allocation strategy is **Round**. **Round** means that when there are multiple stripe groups of simi-

lar classes (for example two stripe groups for non-exclusive data), the space allocator should alternate (round robin) new files through the available stripe groups. Subsequent allocation requests for any one file are directed to the same stripe group. If insufficient space is available in that stripe group, the allocator will choose the next stripe group that can honor the allocation request.

When the strategy is **Balance**, the available blocks of each stripe group are analyzed, and the stripe group with the most total free blocks is chosen. Subsequent requests for the same file are directed to the same stripe group. If insufficient space is available in that stripe group, the allocator will choose the stripe group with the most available space.

When the strategy is **Fill**, the allocator will initially choose the stripe group that has the least amount of total free space. After that it will allocate from the same stripe group until the stripe group cannot honor a request. The allocator then reselects a stripe group using the original criteria.

To use a strategy other than **Round**, the Allocation Session Reservation feature must be disabled.

- XML: **fileLockResyncTimeOut** <value>

  Old: **BRLResyncTimeout** <value>

  NOTE: Not intended for general use. Only use when recommended by Quantum Support.

- XML: **allocSessionReservationSize** <value>

  Old: **AllocSessionReservationSize** <value>

  The Allocation Session Reservation (ASR) feature allows a file system to benefit from optimized allocation behavior for certain rich media streaming applications and most other workloads. The feature also focuses on reducing free space fragmentation.

  By default, this feature is enabled with a 1GB, 1073741824, size.

  An old, deprecated parameter, **AllocSessionReservation**, when set to yes would use a 1 GB segment size with no rounding. This old parameter is now ignored but can generate some warnings.

  allocSessionReservationSize allows you to specify the size this feature should use when allocating segments for a session. The value is expressed in bytes so a value of 2147483648 is 2 GBs. The value must be a multiple of MBs. The XML file format must be in bytes. The old configuration file format can use multipliers such as **m** for MBs or **g** for GBs. If the multiplier is omitted in the old configuration file, the value is interpreted as bytes as in the XML format.

  A value of 0 turns off this capability and falls back on the base allocator. When enabled, the value can range from 128 MB (134217728) to 1 TB (1099511627776). (The largest value would indicate segments are 1 TB in size, which is extremely large.) The feature starts with the specified size and then may use rounding to better handle user's requests. See also **InodeStripeWidth**.

  There are 3 session types: small, medium, and large. The type is determined by the file offset and requested allocation size. Small sessions are for sizes (offset+allocation size) smaller than 1MB. Medium sessions are for sizes 1MB through 1/10th of the **allocSessionReservationSize**. Large sessions are sizes bigger than medium.

  Here is another way to think of these three types: small sessions collect or organize all small files into small session chunks; medium sessions collect medium sized files by chunks using their parent directory; and large files collect their own chunks and are allocated independently of other files.

  All sessions are client specific. Multiple writers to the same directory or large file on different clients will use different sessions. Small files from different clients use different chunks by client.

  Small sessions use a smaller chunk size than the configured **allocSessionReservationSize**. The small chunk size is determined by dividing the configured size by 32. For 128 MB, the small chunk size is 4 MB. For 1 GB, the small chunk size is 32 MBs.

  Files can start using one session type and then move to another session type. If a file starts in a medium session and then becomes large, it "reserves" the remainder of the session chunk it was using for itself. After a session is reserved for a file, a new session segment will be allocated for any other medium files in that

directory.

When allocating subsequent pieces for a session, they are rotated around to other stripe groups that can hold user data unless **InodeStripeWidth** is set to 0. When **InodeStripeWidth** is set, session chunks are rotated in a similar fashion to **InodeStripeWidth**. The direction of rotation is determined by a combination of the session key and the index of the client in the client table. The session key is based on the inode number so odd inodes will rotate in a different direction from even inodes. Directory session keys are based on the inode number of the parent directory.

If this capability is enabled, **StripeAlignSize** is forced to 0. In fact, all stripe alignment requests are disabled because they can cause clipping and can lead to severe free-space fragmentation.

The old **AllocSessionReservation** parameter is deprecated and replaced by **allocSessionReservationSize.**

If any of the following "special" allocation functions are detected, **allocSessionReservationSize** is turned off for that allocation: **PerfectFit**, **MustFit**, or **Gapped files**.

When this feature is enabled, **AllocationStrategy** must be set to **Round**. As of StorNext 6, this is enforced when creating and modifying file systems. If a file system was created using a prior version of StorNext and ASR was enabled but **AllocationStrategy** was not set to **Round**, the FSM will run. However, the **AllocationStrategy** will be treated as **Round** and a warning will be issued whenever the configuration file is parsed.

- XML: **bufferCacheSize** <value>

  Old: **BufferCacheSize** <value>

  This variable defines how much memory to use in the FSM program for general metadata information caching. The amount of memory consumed is up to 2 times the value specified but typically less.

  Increasing this value can improve performance of many metadata operations by performing a memory cache access to directory blocks, inode info and other metadata info. This is about 10 - 1000 times faster than performing I/O.

  There are two buffer caches: the L1 cache and the L2 cache. If bufferCacheSize is configured as 1G or smaller, only the L1 cache is used. If bufferCacheSize is configured greater than 1G, the first 512M is used by the L1 cache and the remainder is used by the L2 cache. Blocks may reside in both caches. Blocks in the L2 cache are compressed by about a factor of 2.4, allowing for better memory utilization. For example, if bufferCacheSize is set to a value of 8G, the FSM will actually be able to cache about 7.5 * 2.4 = 18 G of metadata. Depending on the amount of RAM in the MDC and the number of allocated metadata blocks, in some cases it may be possible to keep all used metadata in cache which can dramatically improve performance for file system scanning. Cvfsck also uses the buffer cache and specifying a large enough value of bufferCacheSize to cover all metadata will result in a large speed increase. The cvadmin "metadata" command can be used to determine the value of bufferCacheSize required to cache all metadata.

  Also see the useL2BufferCache configuration parameter.

- XML: **caseInsensitive** <true|false>

  Old:

  The **caseInsensitive** variable controls how the FSM reports case sensitivity to clients. Windows clients are always case insensitive, Mac clients default to case insensitive, but if the FSM is configured as case sensitive then they will operate in case sensitive mode. Linux clients will follow the configuration variable, but can operate in case insensitive mode on a case sensitive filesystem by using the caseinsensitive mount option. Linux clients must be at the 5.4 release or beyond to enable this behavior.

  Note: You must stop the file system and run **cvupdatefs** once the config file has been updated in order to enable or disable case insensitive. Clients must re-mount the file system to pick up the change.

  When enabling case insensitive, it is also strongly recommended that **cvfsck -A** be run to detect name case collisions. **Cvupdatefs** will not enable case insensitive when name case collisions are present in the file

system.

- XML: **cvRootDir** <path>

  Old: **CvRootDir** <path>

  NOTE: Not intended for general use. Only use when recommended by Quantum Support.

  The **CvRootDir** variable specifies the directory in the StorNext file system that will be mounted by clients. The specified path is an absolute pathname of a directory that will become the root of the mounted file system. The default value for the **CvRootDir** path is the root of the file system, "/". This feature is available only with Quantum StorNext Appliance products.

- XML: **storageManager** <true|false>

  Old: **DataMigration** <Yes|No>

  The **storageManager/DataMigration** statement indicates if the file system is linked to the **Stornext Storage Manager**, which provides hierarchical storage management capabilities to a Stornext Filesystem. Using the **Stornext Storage Manager** requires separately licensed software.

- XML: **debug** <debug_value>

  Old: **Debug** <debug_value>

  The **Debug** variable turns on debug functions for the FSM. The output is sent to */usr/cvfs/data/*<file_system_name>*/log/cvfs_log*. These data may be useful when a problem occurs. A Quantum Technical Support Analyst may ask for certain debug options to be activated when they are trying to analyze a file system or hardware problem. The following list shows which value turns on a specific debug trace. Multiple debugging options may be selected by calculating the bitwise OR of the options' values to use as debug_value. Output from the debugging options is accumulated into a single file.

```
0x00000001     General Information
0x00000002     Sockets
0x00000004     Messages
0x00000008     Connections
0x00000010     File (VFS) requests
0x00000020     File file (VOPS)
0x00000040     Allocations
0x00000080     Inodes
0x00000100     Tokens
0x00000200     Directories
0x00000400     Attributes
0x00000800     Bandwidth Management
0x00001000     Quotas
0x00002000     Administrative Management
0x00004000     I/O
0x00008000     Data Migration
0x00010000     B+Trees
0x00020000     Transactions and Journal
0x00040000     REST API calls and data
0x00080000     Memory Management
0x00100000     QOS IO
0x00200000     External API
0x00400000     Windows Security
0x00800000     Journal Activity
0x01000000     Dump Statistics (Once Only)
0x02000000     Extended Buffers
0x04000000     Extended Directories
0x08000000     Queues
```

```
0x10000000     Extended Inodes
0x20000000     Metadata Archive
0x40000000     Xattr manipulation
0x80000000     Development debug
```

NOTE: The performance of the file system is dramatically affected by turning on debugging traces.

- XML: **dirWarp** <true|false>

  Old: **DirWarp** <Yes|No>

  NOTE: This setting has been deprecated and is no longer supported. It will be ignored.

- XML: **enforceAcls** <true|false>

  Old: **EnforceACLs** <Yes|No>

  Enables Access Control List enforcement on XSan clients. On non-XSan MDCs, **windowsSecurity** should also be enabled for this feature to work with XSan clients.

  This variable is only applicable when **securityModel** is set to **legacy**. It is ignored for other **securityModel** values. See **securityModel** for details.

- XML: **enableSpotlight** <true|false>

  Old: **EnableSpotlight** <Yes|No>

  Enable Spotlight indexing.

- XML: **eventFiles** <true|false>

  Old: **EventFiles** <Yes|No>

  NOTE: Not intended for general use. Only use when recommended by Quantum Support.

  Enables event files processing for Data Migration

- XML: **eventFileDir** <path>

  Old: **EventFileDir** <path>

  NOTE: Not intended for general use. Only use when recommended by Quantum Support.

  Specifies the location to put Event Files

- XML: **extentCountThreshold** <value>

  Old: **ExtentCountThreshold** <value>

  When a file has this many extents, a RAS event is triggered to warn of fragmented files. The default value is 49152. A value of 0 or 1 disables the RAS event. This value must be between 0 and 33553408 (0x1FFFC00), inclusive.

- XML: **fileLocks** <true|false>

  Old: **FileLocks** <Yes|No>

  The variable enables or disables the tracking and enforcement of file-system-wide file locking. Enabling the **File locks** feature allows file locks to be tracked across all clients of the file system. The FileLocks feature supports both the POSIX file locking model and the Windows file locking model.

  If enabled, byte-range file locks are coordinated through the FSM, allowing a lock set by one client to block overlapping locks by other clients. If disabled, then byte-range locks are local to a client and do not prevent other clients from getting byte-range locks on a file, however they do prevent overlapping lock attempts on the same client.

- XML: **forcePerfectFit** <true|false>

  Old: **ForcePerfectFit** <Yes|No>

  NOTE: Not intended for general use. Only use when recommended by Quantum Support.

Enables a specialized allocation mode where all files are automatically aligned and rounded to **PerfectFitSize** blocks. If this is enabled, **allocSessionReservationSize** is ignored.

- XML: **fsBlockSize** <value>

  Old: **FsBlockSize** <value>

  The File System Block Size defines the granularity of the file system's allocation size. The block size is fixed at 4K. When an older file system is upgraded to StorNext 5, if the block size is other than 4k, the file system is converted to a 4K block size. For these file systems, the original block size value remains in the config file. If a file system is remade that had a file system block size other than 4K, the config file is rewritten, changing the file system block size parameter value to 4K.

- XML: **fsCapacityThreshold** <value>

  Old: **FsCapacityThreshold** <value>

  When a file system is over **fsCapacityThreshold** percent full, a RAS event is sent to warn of this condition. This value must be between 0 and 100, inclusive. The default value is 0, which disables the RAS event for all file systems except the HA shared file system which defaults to 85%. To disable this RAS event for the HA shared file system, set fsCapacityThreshold to 100.

- XML: **fsmMemLocked** <true|false>

  Old: **FSMMemlock** <Yes|No>

  The **FSM Memory lock** variable instructs the FSM to ask the kernel to lock it into memory on platforms that support this. This prevents the FSM from getting swapped or paged out and provides a more responsive file system. Running with this option when there is insufficient memory for the FSM to run entirely in core will result in the FSM terminating. The default value is **No**. This is only supported on POSIX conforming platforms.

- XML: **fsmRealTime** <true|false>

  Old: **FSMRealtime** <yes|no>

  The **FSM Realtime** variable instructs the FSM to run itself as a realtime process on platforms that support this. This allows the FSM to run at a higher priority than other applications on the node to provide a more responsive file system. The default value is **No**. This is only supported on POSIX conforming platforms.

- XML: **globalShareMode** <true|false>

  Old: **GlobalShareMode** <Yes|No>

  The **GlobalShareMode** variable enables or disables the enforcement of Windows Share Modes across StorNext clients. This feature is limited to StorNext clients running on Microsoft Windows platforms. See the Windows CreateFile documentation for the details on the behavior of share modes. When enabled, sharing violations will be detected between processes on different StorNext clients accessing the same file. Otherwise sharing violations will only be detected between processes on the same system. The default of this variable is **false**. This value may be modified for existing file systems.

- XML: **globalSuperUser** <true|false>

  Old: **GlobalSuperUser** <Yes|No>

  The **Global Super User** variable allows the administrator to decide if any user with super-user privileges may use those privileges on the file system. When this variable is set to **true**, any super-user has global access rights on the file system. This may be equated to the **maproot=0** directive in NFS. When the **Global Super User** variable is set to **false**, a super-user may only modify files where it has access rights as a normal user. This value may be modified for existing file systems. If **storageManager** is enabled and this variable is set to **false**, the value will be overriden and set to **true** on storage manager nodes. A storage manager node is the MDC or a Distributed Data Mover client. Apple Xsan clients do not honor the setting of **globalSuperUser**.

- XML: **haFsType** <HaShared|HaManaged|HaUnmanaged|HaUnmonitored>

Old: **HaFsType** <HaShared|HaManaged|HaUnmanaged|HaUnmonitored>

The **HaFsType** configuration item turns on StorNext High Availability (HA) protection for a file system, which prevents split-brain scenario data corruption. HA detects conditions where split brain is possible and triggers a hardware reset of the server to remove the possibility of split brain scenario. This occurs when an activated FSM is not properly maintaining its brand of an arbitration block (ARB) on the metadata LUN. Timers on the activated and standby FSMs coordinate the usurpation of the ARB so that the activated server will relinquish control or perform a hardware reset before the standby FSM can take over. It is very important to configure all file systems correctly and consistently between the two servers in the HA cluster.

There are currently three types of HA monitoring that are indicated by the **HaShared**, **HaManaged, and HaUnmanaged** configuration parameters.

The **HaShared** dedicated file system holds shared data for the operation of the **StorNext File System** and **Stornext Storage Manager** (SNSM). There must be one and only one **HaShared** file system configured for these installations. The running of SNSM processes and the starting of managed file systems is triggered by activation of the **HaShared** file system. In addition to being monitored for ARB branding as described above, the exit of the **HaShared** FSM triggers a hardware reset to ensure that SNSM processes are stopped if the shared file system is not unmounted.

The **HaManaged** file systems are not started until the **HaShared** file system activates. This keeps all the managed file systems collocated with the SNSM processes. It also means that they cannot experience split-brain corruption because there is no redundant server to compete for control, so they are not monitored and cannot trigger a hardware reset.

The **HaUnmanaged** file systems are monitored. The minimum configuration necessary for an HA cluster is to: 1) place this type in all the FSMs, and 2) enter the peer server's IP address in the **ha_peer**(4) file. Unmanaged FSMs can activate on either server and fail over to the peer server without a hardware reset under normal operating conditions.

On non-HA setups, the special **HaUnmonitored** type is used to indicate no HA monitoring is done on the file systems. It is only to be used on non-HA setups. Note that setting HaFsType to HaUnmonitored disables the HA monitor timers used to guarantee against split brain. When two MDCs are configured to run as an HA pair but full HA protection is disabled in this way, it is possible in rare situations for file system metadata to become corrupt if there are lengthy delays or excessive loads in the LAN and SAN networks that prevent an active FSM from maintaining its branding of the ARB in a timely manner.

- XML: **inodeCacheSize** <value>

  Old: **nodeCacheSize** <value>

  This variable defines how many inodes can be cached in the FSM program. An in-core inode is approximately 800 - 1000 bytes per entry.

- XML: **inodeDeleteMax** <value>

  Old: **InodeDeleteMax** <value>

  NOTE: Not intended for general use. Only use when recommended by Quantum Support.

  Sets the trickle delete rate of inodes that fall under the **Perfect Fit** check (see the **Force Perfect Fit** option for more information. If **Inode Delete Max** is set to 0 or is excluded from the configuration file, it is set to an internally calculated value.

- XML: **inodeExpandMin** <file_system_blocks>

  Old: **InodeExpandMin** <file_system_blocks>

- XML: **inodeExpandInc** <file_system_blocks>

  Old: **InodeExpandInc** <file_system_blocks>

- XML: **inodeExpandMax** <file_system_blocks>

  Old: **InodeExpandMax** <file_system_blocks>

The **inodeExpandMin**, **inodeExpandInc** and **inodeExpandMax** variables configure the floor, increment and ceiling, respectively, for the block allocation size of a dynamically expanding file. The new format requires this value be specified in bytes and multipliers are not supported. In the old format, when the value is specified without a multiplier suffix, it is a number of file system blocks; when specified with a multiplier, it is bytes.

The first time a file requires space, **inodeExpandMin** blocks are allocated. When an allocation is exhausted, a new set of blocks is allocated equal to the size of the previous allocation to this file plus **inodeExpandInc** additional blocks. Each new allocation size will increase until the allocations reach **inodeExpandMax** blocks. Any expansion that occurs thereafter will always use **inodeExpandMax** blocks per expansion.

NOTE: when **inodeExpandInc** is not a factor of **inodeExpandMin**, all new allocation sizes will be rounded up to the next **inodeExpandMin** boundary. The allocation increment rules are still used, but the actual allocation size is always a multiple of **inodeExpandMin**.

NOTE: The explicit use of the configuration variables **inodeExpandMin**, **inodeExpandInc** and **inodeExpandMax** are being deprecated in favor of an internal table driven mechanism. Although they are still supported for backward compatibility, there may be warnings during the conversion of an old configuration file to an XML format.

- XML: **inodeStripeWidth** <value>

  Old: **InodeStripeWidth** <value>

  The **Inode Stripe Width** variable defines how a file is striped across the file system's data stripe groups. The default value is 4 GBs (4294967296). After the initial placement policy has selected a stripe group for the first extent of the file, for each **Inode Stripe Width** extent the allocation is changed to prefer the next stripe group allowed to contain file data. Next refers to the next numerical stripe group number going up or down. (The direction is determined using the inode number: odd inode numbers go up or increment, and even inode numbers go down or decrement). The rotation is modulo the number of stripe groups that can hold data.

  When **Inode Stripe Width** is not specified, file data allocations will typically attempt to use the same stripe group as the initial allocation to the file.

  When used with an **Allocation Strategy** setting of **Round**, files will be spread around the allocation groups both in terms of where their initial allocation is and in how the file contents are spread out.

  Inode Stripe Width is intended for large files. The typical value would be many times the maximum **Stripe Breadth** of the data stripe groups. The value cannot be less than the maximum **Stripe Breadth** of the data stripe groups. Note that when some stripe groups are full, this policy will start to prefer the stripe group logically following the full one. A typical value is 4 GB (4294967296) or 8 GBs (8589934592). The size is capped at 1099511627776 (1TB).

  If this value is configured too small, fragmentation can occur. Consider using a setting of 1MB with files as big as 100 GBs. Each 100 GB file would have 102,400 extents!

  The new format requires this value be specified in bytes, and multipliers are not supported. In the old format, when the value is specified without a multiplier suffix, it is a number of file system blocks; when specified with a multiplier, it is bytes.

  When **allocSessionReservationSize** is non-zero, this parameter is forced to be >= **allocSessionReservationSize**.

  If **Inode Stripe Width** is greater than **allocSessionReservationSize**, files larger than **allocSessionReservationSize** will use **Inode Stripe Width** as their **allocSessionReservationSize** for allocations with an offset beyond **allocSessionReservationSize**.

- XML: **ioTokens** <true|false>

  Old: **IoTokens** <Yes|No>

  The **I/O Tokens** variable allows the administrator to select which coherency model should be used when different clients open the same file, concurrently. With **ioTokens** set to false, the coherency model uses 3

states: exclusive, shared, and shared write. If a file is exclusive, only one client is using the file. Shared indicates that multiple clients have the file open but for read only mode. This allows clients to cache data in memory. Shared write indicates multiple clients have the file open and at least one client has the file open for write. With "Shared Write" mode, coherency is resolved by using DMA I/O and no caching of data.

A problem with DMA I/O is that small or unaligned I/Os need to do a read-modify-write. So, two racing clients can undo each other's writes since they could have data in memory. This occurs when a client has read into a buffer, modifies part of the buffer, and then write it using DMA (after the other client's write that occurred before this client read into the buffer being written). Different platforms have requirements on the granularity of DMA I/O, usually at least 512 bytes that must be written and also using a 512 or greater boundary for the start and end of the I/O.

If one sets **ioTokens** to true (the default setting), each I/O performed by a client must have a token. Clients cache and can do many I/Os while they have the token. When the token is revoked, all data and associated attributes are flushed.

Customers, who have multiple writers on a file, should set **ioTokens** to true, unless they know that the granularity and length of I/Os are safe for DMA. File locking does NOT prevent read-modify-write across lock boundaries.

The default for I/O Tokens is true.

For backward compatibility, if a client opens a file from a prior release that does not support **ioTokens**, the coherency model drops back to the "Shared Write" model using DMA I/O (**ioTokens** false) but on a file-by-file basis.

If **ioTokens** is changed and the MDC is restarted, files that were open at that time continue to operate in the model before the change. To switch these files to the new value of **ioTokens**, all applications must close the file and wait for a few seconds and then re-open it. Or, if the value was switched from true to false, a new client can open the file and all clients will transparently be switched to the old model on that file.

- XML: **journalSize** <value>

  Old: **JournalSize** <value>

  Controls the size of the file system journal. **cvupdatefs**(8) must be run after changing this value for it to take effect. The FSM will not activate if it detects that the journal size has been changed in the config file, but the metadata has not been updated.

- XML: **maintenanceMode** <true|false>

  Old: **MaintenanceMode** <Yes|No>

  The **maintenanceMode** parameter enables or disables maintenance mode for the file system. In maintenance mode, all client mount requests are rejected by the FSM except from the client running on the same node as the FSM.

  NOTE: Not intended for general use. Only use when recommended by Quantum Support.

- XML: **maxLogs** <value>

  Old: **MaxLogs** <value>

  The **maxLogs** variable defines the maximum number of logs a FSM can rotate through when they get to **MaxLogSize**. The current log file resides in */usr/cvfs/data/*<file_system_name>*/log/cvlog*.

- XML: **maxLogSize** <value>

  Old: **MaxLogSize** <value>

  The **maxLogSize** variable defines the maximum number of bytes a FSM log file should grow to. The log file resides in */usr/cvfs/data/*<file_system_name>*/log/cvlog*. When the log file grows to the specified size, it is moved to **cvlog_<number>** and a new **cvlog** is started. Therefore, the maximum space consumed will be **maxLogs** multiplied by **maxLogSize**.

- XML: **namedStreams** <true|false>

Old: **NamedStreams** <Yes|No>

The **namedStreams** parameter enables or disables support for Apple Named Streams. Named Streams can be used by macOS clients to efficiently store resource forks and extended attributes directly in file system metadata instead of using Apple Double files. If namedStreams is not enabled when the file system is initialized, **cvupdatefs**(8) must be run after enabling namedStreams for it to take effect. The FSM will not activate if it detects that namedStreams has been enabled in the config file, but the metadata has not been updated by **cvupdatefs**. Enabling namedStreams is meant to be a permanent operation. Once enabled, disabling namedStreams requires a special procedure only available through technical support that is not always feasible. Note that most "copy" programs on Windows and Linux do not preserve namedStreams. This includes Windows Explorer. Also note that this parameter applies to Apple Named Streams support in the file system only. The StorNext NAS SMB server has its own named streams option that must be activated separately.

- XML: **opHangLimitSecs** <value>

  Old: **OpHangLimitSecs** <value>

  This variable defines the time threshold used by the FSM program to discover hung operations. The default is 180. It can be disabled by specifying 0. When the FSM program detects an I/O hang, it will stop execution in order to initiate failover to backup system.

- XML: **perfectFitSize** <value>

  Old: **PerfectFitSize** <value>

  For files in perfect fit mode, all allocations will be rounded up to the number of file system blocks set by this variable. Perfect fit mode can be enabled on an individual file by an application using the SNFS extended API, or for an entire file system by setting **forcePerfectFit**.

  If **InodeStripeWidth** or **allocSessionReservationSize** are non-zero and Perfect fit is not being applied to an allocation, this rounding is skipped.

- XML: **quotas** <true|false>

  Old: **Quotas** <Yes|No>

  The **quotas** variable enables or disables the enforcement of the file system quotas. Enabling the quotas feature allows storage usage to be tracked for individual users and groups. Setting hard and soft quotas allows administrators to limit the amount of storage consumed by a particular user/group ID. See **snquota**(1) for information on quotas feature commands.

  NOTE: Quotas are calculated differently on Windows and Linux systems. It is not possible to migrate a meta data controller running quotas between these different types.

  NOTE: Quotas are not allowed when **securityModel** is set to legacy and **windowsSecurity** is set to false.

  NOTE: When using a Windows MDC, quotas are not allowed if **securityModel** is set to unixpermbits.

- XML: **quotaHistoryDays** <value>

  Old: **QuotaHistoryDays** <value>

  When the **quotas** variable (see above) is turned on, there will be nightly logging of the current quota limits and values. The logs will be placed in the */usr/cvfs/data/*<file_system_name>*/quota_history* directory. This variable specifies the number of days of logs to keep. Valid values are 0 (no logs are kept) to 3650 (10 years of nightly logs are kept). The default is 7.

- XML: **remoteNotification** <true|false>

  Old: **RemoteNotification** <Yes|No>

  The **remoteNotification** variable controls the Windows Remote Directory Notification feature. The default value is no which disables the feature. Note: this option is not intended for general use. Only use when recommended by Quantum Support.

- XML: **renameTracking** <true|false>

Old: **RenameTracking** <Yes|No>

The **renameTracking** variable controls the **Stornext Storage Manager** (SNSM) rename tracking feature. This replaces the (global) Storage Manager configuration variable MICRO_RENAME that was present in older versions of StorNext. It is by default set to 'false'. Note that this feature should ONLY be enabled at sites where Microsoft, or other similar applications, end up renaming operational files during their processing. See the fsrecover(1) man page for more information on use of renameTracking.

- XML: **reservedSpace** <true|false>

  Old: **ReservedSpace** <Yes|No>

  NOTE: Not intended for general use. Only use when recommended by Quantum Support.

  The **reservedSpace** parameter allows the administrator the ability to control the use of delayed allocations on clients. The default value is **Yes**. **reservedSpace** is a performance feature that allows clients to perform buffered writes on a file without first obtaining real allocations from the FSM. The allocations are later performed when the data are flushed to disk in the background by a daemon performing a periodic sync.

  When **reservedSpace** is **true**, the FSM reserves enough disk space so that clients are able to safely perform these delayed allocations. The meta-data server reserves a minimum of 4GB per stripe group and up to 280 megabytes per client per stripe group.

  Setting **reservedSpace** to **false** allows slightly more disk space to be used, but adversely affects buffer cache performance and may result in serious fragmentation.

  XML: metadataArchive <true|false>

  The **metadataArchive** statement is used to enable or disable the Metadata Archive created by the FSM. The Metadata Archive contains a copy of all file system metadata including past history of metadata changes if **metadataArchiveDays** is set to a value greater than zero. The Metadata Archive is used for disaster recovery, file system event notification, and file system auditing among other features.

  XML: metadataArchiveDir <path>

  The **metadataArchiveDir** statement is used to change the path in which the Metadata Archive is created. The default path is */usr/adic/database/mdarchives/* for all file systems except non-managed file systems not running in an HA environment where the path is then */usr/cvfs/data/*<file_system_name>/.

  XML: metadataArchiveSearch <true|false>

  The **metadataArchiveSearch** statement is used to enable or disable the Metadata Archive Search capability in Metadata Archive. If enabled, Metadata Archive supports advanced searching capabilities which are used by various other StorNext features. Metadata Archive Search is enabled by default and should only be turned off if performance issues are experienced.

  XML: metadataArchiveCache <bytes>

  The **metadataArchiveCache** statement is used to configure the size of the memory cache for the Metadata Archive. The minimum cache size is 1GB, the maximum is 500GB, and the default is 2GB.

  XML: metadataArchiveDays <value>

  The **metadataArchiveDays** statement is used to set the number of days of metadata history to keep available in the Metadata Archive. The default value is zero (no metadata history).

  XML: audit <true|false>

  The **audit** keyword controls if the filesystem maintains extra metadata for use with the snaudit command and for tracking client activity on files. The default value is false and this feature requires that **metadataArchive** be enabled.

  XML: restAccess <privileged|enabled|disabled>

  Controls the presentation of a rest API for various filesystem capabilities on Linux systems. An https service is presented by the FSM if this is enabled. Various utilities such as **sgmanage** and parts of the GUI make use of this. Some rest services also depend on **metadataArchive** being enabled. When the mode is

set to **privileged**, the access information for the service is only available to privileged users. When the mode is **enabled**, any user may view the service. The service may be disabled completely with by setting this to disabled. The default is **privileged**.

- XML: **restoreJournal** <true|false>

  Old: **RestoreJournal** <Yes|No>

  NOTE: The restoreJournal statement has been deprecated and replaced by metadataArchive. It is supported for backward compatibility only. A restoreJournal setting of true is equivalent to setting metadataArchive to true and setting metadataArchiveDays to zero (no metadata history).

  The **restoreJournal** statement is used to enable or disable the Metadata Archive created by the FSM process.

- XML: **restoreJournalDir** <path>

  Old: **RestoreJournalDir** <path>

  NOTE: The **restoreJournalDir** statement has been deprecated and replaced by metadataArchiveDir. It is supported for backward compatibility only and is ignored completely if metadataArchive is set to true.

  The **restoreJournalDir** statement is used to change the path in which the Metadata Archive is created. The default path is */usr/adic/database/mdarchives/* for all file systems except non-managed file systems not running in an HA environment where the path is then */usr/cvfs/data/* <file_system_name>/.

- XML: **restoreJournalMaxHours** <value>

  Old: **RestoreJournalMaxHours** <value>

- XML: **restoreJournalMaxMb** <value>

  Old: **RestoreJournalMaxMB** <value>

  The **restoreJournalMaxMB** and **restoreJournalMaxHours** statements are obsolete and are only included for backward compatibility. Setting these global variables does not affect the file system in any way.

- XML: **securityModel** <legacy|acl|unixpermbits>

  Old: **SecurityModel** <legacy|acl|unixpermbits>

  The **securityModel** variable determines the security model to use on SNFS clients. **legacy** is the default value.

  When set to **legacy**, the **windowsSecurity** variable is checked to determine whether or not Windows clients should make use of the Windows Security Reference Monitor (ACLs). The **windowsIdMapping** variable is ignored for this security model.

  When set to **acl**, all SNFS clients (Windows and Unix) will make use of the Windows Security Reference Monitor (ACLs). The **windowsSecurity**, **windowsIdMapping**, and **enforceAcls** variables are ignored for this security model.

  When set to **unixpermbits**, all SNFS clients (Unix and Windows) will use Unix permission bit settings when performing file access checks. When **unixpermbits** is specified, an additional variable, **windowsIdMapping**, is used to control the method used to perform the Windows User to Unix User/Group ID mappings. See the **windowsIdMapping** variable for additional information. The **windowsSecurity**, **useActiveDirectorySFU**, **enforceAcls**, and **unixIdFabricationOnWindows** variables are ignored for this security model.

  NOTE: The **unixpermbits** setting does not support the Windows NtCreateFile function FILE_OPEN_BY_FILE_ID option, which opens a file by inode number versus file name.

- XML: **spotlightSearchLevel** <FsSearch|ReadWrite>

  Old: **SpotlightSearchLevel** <FsSearch|ReadWrite>

  Set the SpotlightSearchLevel. This option only applies when Xsan MDCs are used and should not be used elsewhere as it can interfere with Spotlight Proxy functionality.

- XML: **spotlightUseProxy** <true|false>

  Old: **SpotlightUseProxy** <Yes|No>

  Enable properly configured Xsan clients to act as proxy servers for OS X Spotlight Search on SNFS.

- XML: **stripeAlignSize** <value>

  Old: **StripeAlignSize** <value>

  The **stripeAlignSize** statement causes the allocator to automatically attempt stripe alignment and rounding of allocations greater than or equal to this size. The new format requires this value be specified in bytes and multipliers are not supported. In the old format, when the value is specified without a multiplier suffix, it is a number of file system blocks; when specified with a multiplier, it is bytes. If set to default **value** (-1), it internally gets set to the size of largest **stripeBreadth** found for any **stripeGroup** that can hold user data. A value of 0 turns off automatic stripe alignment. Stripe-aligned allocations are rounded up so that allocations are one stripe breadth or larger.

  If an allocation fails with stripe alignment enabled, another attempt is made to allocate the space without stripe alignment.

  If **allocSessionReservationSize** is enabled, **stripeAlignSize** is set to 0 to reduce fragmentation within segments which occurs when clipping within segments.

- XML: **trimOnClose** <value>

  Old: **TrimOnClose** <value>

  NOTE: Not intended for general use. Only use when recommended by Quantum Support.

- XML: **useL2BufferCache** <true|false>

  Old: **UseL2BufferCache** <yes|no>

  The **useL2BufferCache** variable determines whether the FSM should use the compressed L2 metadata block cache when the bufferCacheSize is greater than 1GB. The default is true. Setting this variable to false may delay FSM startup when using a very large value for bufferCacheSize.

  NOTE: This variable may be removed in a future release.

  NOTE: Not intended for general use. Only use when recommended by Quantum Support.

- XML: **unixDirectoryCreationModeOnWindows** <value>

  Old: **UnixDirectoryCreationModeOnWindows** <value>

  The **unixDirectoryCreationModeOnWindows** variable instructs the FSM to pass this value back to Microsoft Windows clients. The Windows SNFS clients will then use this value as the permission mode when creating a directory. The default value is 0755. This value must be between 0 and 0777, inclusive.

- XML: **unixFileCreationModeOnWindows** <value>

  Old: **UnixFileCreationModeOnWindows** <value>

  The **unixFileCreationModeOnWindows** variable instructs the FSM to pass this value back to Microsoft Windows clients. The Windows SNFS clients will then use this value as the permission mode when creating a file. The default value is 0644. This value must be between 0 and 0777, inclusive.

- XML: **unixIdFabricationOnWindows** <true|false>

  Old: **UnixIdFabricationOnWindows** <yes|no>

  The **unixIdFabricationOnWindows** variable is simply passed back to a Microsoft Windows client. The client uses this information to turn on/off "fabrication" of uid/gids from a Microsoft Active Directory obtained GUID for a given Windows user. A value of yes will cause the client for this file system to fabricate the uid/gid and possibly override any specific uid/gid already in Microsoft Active Directory for the Windows user. This setting should only be enabled if it is necessary for compatibility with Apple MacOS clients. The default is false, unless the meta-data server is running on Apple MacOS, in which case it is

true.

This variable is only applicable when **securityModel** is set to **legacy** or **acl**. It is ignored for other **securityModel** values. See **securityModel** for details.

- XML: **unixIdMapping** <value>

  Old: **UnixIdMapping** <value>

  When **securityModel** is set to **acl**, the **unixIdMapping** variable determines the method Linux and Unix clients use to perform Unix User/Group ID to Windows User mappings used by ACLs. This setting has no effect on Windows or Xsan clients.

  The default value of this variable is **none** which is incompatible with setting **securityModel** to **acl**.

  A value of **winbind** should be used when the environment contains Linux clients that are all bound to Active Directory using Winbind and running the winbind service.

  A value of **mdc** should be used when the MDCs for a file system are bound to Active Directory using Winbind but one or more of the Linux clients in the environment are not running Winbind. For example, Linux clients may instead be bound to Active Directory using sssd. The use of **mdc** unixIdMapping allows such environments to be supported by having Linux clients forward ID mapping requests to the MDC for processing.

  When **unixIdMapping** is set to **algorithmic**, UIDs are mapped to SIDs using the following:
  RID(uid) = (2 * uid) + 1000
  The RID is then appended to the Domain SID. For the **algorithmic** unixIdMapping, the default value of the Domain SID is:
  S-5-21-3274805877-1740924817-4269325941
  For example, a user having a UID of 400, will have the SID:
  S-5-21-3274805877-1740924817-4269325941-1800
  GIDs are mapped to SIDs using the following:
  RID(gid) = (2 * gid) + 1001
  The RID is then appended to the Domain SID. For example, a group having a GID of 300 will have the SID:
  S-5-21-3274805877-1740924817-4269325941-1601
  Note: while commonly only required when using Open Directory, the Domain SID can be overridden using the StorNext domainsid (4) configuration file.

- XML: **unixNobodyGidOnWindows** <value>

  Old: **UnixNobodyGidOnWindows** <value>

  The **unixNobodyGidOnWindows** variable instructs the FSM to pass this value back to Microsoft Windows clients. The Windows SNFS clients will then use this value as the gid for a Windows user when no gid can be found using Microsoft Active Directory. The default value is 60001. This value must be between 0 and 2147483647, inclusive.

- XML: **unixNobodyUidOnWindows** <value>

  Old: **UnixNobodyUidOnWindows** <value>

  The **unixNobodyUidOnWindows** variable instructs the FSM to pass this value back to Microsoft Windows clients. The Windows SNFS clients will then use this value as the uid for a Windows user when no uid can be found using Microsoft Active Directory. The default value is 60001. This value must be between 0 and 2147483647, inclusive.

- XML: **useActiveDirectorySFU** <true|false>

  Old: **UseActiveDirectorySFU** <Yes|No>

  The **useActiveDirectorySFU** variable enables or disables the use of Microsoft's Active Directory Services for UNIX (SFU) on Windows based SNFS clients. (Note: Microsoft has changed the name "Services for UNIX" in recent releases of Windows. We are using the term SFU as a generic name for all similar Active

Directory Unix services.) This variable does not affect the behavior of Unix clients. Active Directory SFU allows Windows-based clients to obtain the Windows user's Unix security credentials. By default, SNFS clients running on Windows query Active Directory to translated Windows SIDs to Unix uid, gid and mode values and store those credentials with newly created files. This is needed to set the proper Unix uid, gid and permissions on files. If there is no Active Directory mapping of a Windows user's SID to a Unix user, a file created in Windows will have its uid and gid owned by **NOBODY** in the Unix view (See **unixNobodyUidOnWindows**.)

Always use Active Directory SFU in a mixed Windows/Unix environment, or if there is a possibility in the future of moving to a mixed environment. If **useActiveDirectorySFU** is set to **false**, files created on Windows based SNFS clients will always have their uid and gid set to **NOBODY** with default permissions.

However, if it is unlikely a Unix client will ever access the SNFS file system, then you may get a small performance increase by setting **useActiveDirectorySFU** to **false**. The performance increase will be substantial higher only if you have more than 100 users concurrently access the file system via a single Windows SNFS client.

This variable is only applicable when **securityModel** is set to **legacy** or **acl**. It is ignored for other **securityModel** values. See **securityModel** for details.

The default of this variable is **true**. This value may be modified for existing file systems.

- XML: **windowsIdMapping** <ldap|mdc|mdcall|none>

  Old: **WindowsIdMapping** <ldap|mdc|mdcall|none>

  The **windowsIdMapping** variable determines the method Windows clients should use to perform the Windows User to Unix User/Group ID mappings. **ldap** is the default value.

  This variable is only applicable when **securityModel** is set to **unixpermbits**. It is ignored for other **securityModel** values. See **securityModel** for details. Note that due to caching, the effect of changing the **windowsIdMapping** may not be seen on Windows clients until 10-15 minutes after the FSM is restarted unless StorNext is also subsequently restarted on Windows clients.

  When set to **ldap**, Microsoft Active Directory is queried to obtain uid/gid values for the Windows User, including support for up to 32 supplemental GIDs.

  When set to **mdc**, the SNFS MDC is queried to obtain uid/gid values for Windows users that are in the Active Directory domain that the system belongs to. This includes support for an unlimited number of supplemental GIDs. However, local users and groups are NOT mapped. The **mdc** setting is not valid on Windows MDCs.

  When set to **mdcall**, ID mapping on Windows works the same as described above for the **mdc** type except that locally created Windows accounts are also mapped. Note that with this setting, Windows systems that are not joined to any domain can still use MDC mapping. The **mdcall** setting is not valid on Windows MDCs.

  When set to **none**, then there is no specific Windows User to Unix User mapping (see the Windows control panel). In this case, files will be owned by NOBODY in the Unix view.

- XML: **windowsSecurity** <true|false>

  Old: **WindowsSecurity** <Yes|No>

  The **windowsSecurity** variable enables or disables the use of the Windows Security Reference Monitor (ACLs) on Windows clients. This does not affect the behavior of Unix clients. In a mixed client environment where there is no specific Windows User to Unix User mapping (see the Windows control panel), files under Windows security will be owned by **NOBODY** in the Unix view. The default of this variable is **false** for configuration files using the old format and **true** when using the new XML format. This value may be modified for existing file systems.

  This variable is only applicable when **securityModel** is set to **legacy**. It is ignored for other **securityModel** values. See **securityModel** for details.

  NOTE: Once windowsSecurity has been enabled, the file system will track Windows access control lists

(ACLs) for the life of the file system regardless of the **windowsSecurity** value.

AUTOAFFINITY DEFINITION

A **autoAffinity** defines a mapping of file extension(s) to an **Affinity**. A **noAffinity** defines a mapping of file extensions to an affinity of 0. The **Affinity** must exist in the stripe group section (see below). At file creation time, if the file has an extension in the list specified, it will be assigned the **Affinity** or 0. This is only done for regular files and not other types of files such as directories, devices, symbolic links, etc. An extension can only exist once for all **autoAffinity** and **noAffinity** mappings.

Extensions in a file name are defined by all the characters following the last "." in the file name. The **extension** tag in the configuration file is followed by the characters in the extension without the ".". There is one special extension that is defined by not specifying an extension. This is the "empty" extension and tells file creation to map all files not matching another extension to the **autoAffinity** or **noAffinity** mapping it is in.

For example, an administrator can map all files ending in .dpx to an affinity of **Movies**. Or, all remaining files could be mapped to an affinity of **Other**.

Customers can explicitly assign affinities to files and directories using the cvmkdir, cvmkfile, or cvaffinity commands. Or, files can be assigned affinities with library API calls from within applications. The automatic affinities defined in this section take precedence and override affinities set with cvmkdir/cvmkfile or via a library function. For example, if a directory exists with an affinity of **Audio** and a file is created in that directory with a dpx extension with the above autoAffinity mapping. The *.dpx files gets assigned the **Movies** affinity overriding **Audio**.

The cvaffinity command can be used to later change the affinity of a file to some other value.

Some applications create temporary files before renaming them to their final name. Mappings of extension to affinity take effect only on the create call. So for these applications, the temporary file name determines the file's affinity. If the temporary file name has a different extension or no extension, the temporary's extension is used for the mapping. If the file is renamed to a different extension, the mapping is not affected. A typical example of this is Microsoft Word.

DISKTYPE DEFINITION

A **diskType** defines the number of sectors for a category of disk devices, and optionally the number of bytes per disk device sector. Since multiple disks used in a file system may have the same type of disk, it is easier to consolidate that information into a disk type definition rather than including it for each disk definition.

For example, a 9.2GB Seagate Barracuda Fibre Channel ST19171FC disk has **1778311** total sectors. However, using most drivers, a portion of the disk device is used for the volume header. For example, when using a **Prisa** adapter and driver, the maximum number of sectors available to the file system is **11781064**.

When specified, the sector size must be 512 or 4096 bytes. The default sector size is 512 bytes.

DISK DEFINITION

Note: The XML format defines disks in the stripeGroup section. The old format defines disks in a separate section and then links to that definition with the **node** variable in the stripe group. The general description below applies to both.

Each **disk** defines a disk device that is in the Storage Area Network configuration. The name of each disk device must be entered into the disk device's volume header label using **cvlabel**(8). Disk devices that the client cannot see will not be accessible, and any stripe group containing an inaccessible disk device will not be available, so plan stripe groups accordingly. Entire disks must be specified here; partitions may not be used.

The disk definition's **name** must be unique, and is used by the file system administrator programs.

A disk's status may be up or down. When down, this device will not be accessible. Users may still be able to see directories, file names and other meta-data if the disk is in a stripe group that only contains userdata, but attempts to open a file affected by the downed disk device will receive an **Operation Not Permitted (EPERM)** failure. When a file system contains down data stripe groups, space reporting tools in the operating system will not count these stripe groups in computing the total file system size and available free

blocks. *NOTE*: when files are removed that only contain extents on **down** stripe groups, the amount of available free space displayed will not change.

Each disk definition has a type which must match one of the names from a previously defined **diskType**.

NOTE: In much older releases there was also a **DeviceName** option in the **Disk** section. The **DeviceName** was previously used to specify a operating system specific disk name, but this has been superseded by automatic volume recognition for some time and is no longer supported. This is now for internal use only.

STRIPEGROUP DEFINITION

The **stripeGroup** defines individual stripe groups. A stripe group is a collection of disk devices. A disk device may only be in one stripe group.

The **stripeGroup** has a name **name** that is used in subsequent system administration functions for the stripe group.

A stripe group can be set to have it's status up or down. If down, the stripe group is not used by the file system, and anything on that stripe group is inaccessible. This should normally be left up.

A stripe group can contain a combination of **metadata**, **journal**, or **userdata**. There can only be one stripe group that contains a **journal** per file system. Best performance is attained with a minimum of 2 stripe groups per file system with one stripe group used exclusively for metadata/journal and the other for user data. Metadata has an I/O pattern of small random I/O whereas user data is typically of much larger size. Splitting apart metadata and journal so there are 3 stripe groups is recommended particularly if latency for file creation, removal and allocation of space is important.

When a collection of disk devices is assembled under a stripe group, each disk device is logically striped into chunks of disk blocks as defined by the **stripeBreadth** variable. For example, with a 4k-byte block-size and a stripe breadth of 86 file system blocks, the first 352,256 bytes would be written or read from/to the first disk device in the stripe group, the second 352,256 bytes would be on the second disk device and so on. When the last disk device used its 352,256 bytes, the stripe would start again at drive zero. This allows for more than a single disk device's bandwidth to be realized by applications.

The allocator aligns an allocation that is greater than or equal to the largest **stripeBreadth** of any stripe group that can hold data. This is done if the allocation request is an extension of the file.

A stripe group can be marked up or down. When the stripe group is marked down, it is not available for data access. However, users may look at the directory and meta-data information. Attempts to open a file residing on a downed stripe group will receive a **Permission Denied** failure.

There is an option to turn off reads to a stripe group. *NOTE*: Not intended for general use. Only use when recommended by Quantum Support.

A stripe group can have write access denied. If writes are disabled, then any new allocations are disallowed as well. When a file system contains data stripe groups with writes disabled, space reporting tools in the operating system will show all blocks for the stripe group as **used**. Note that when files are removed that only contain extents on write-disabled stripe groups, the amount of available free space displayed will not change. This is typically only used during *Dynamic Resource Allocation* procedures (see the StorNext User Guide for more details).

Allocations can be disabled on a stripe group. This would typically be done as a step towards retiring a stripe group. Unlike disabling writes, turning off allocations allows writes to a file which do not require a new allocation. On Linux systems, the stripe group management utilities **sgmanage** and **sgoffload** can be used to change this field, while the file system remains up and on-line.

Affinities can be used to target allocations at specific stripe groups, and the stripe group can exclusively contain affinity targeted allocations or have affinity targeted allocations co-existing with other allocations. See **snfs.cfg**(5) and **snfs.cfgx**(5) for more details.

Each stripe group can define a multipath method, which controls the algorithm used to allocate disk I/Os on paths to the storage when the file system has multiple paths available to it. See **sgmanage**(8) for details.

Various realtime I/O parameters can be specified on a per stripe group basis as well. These define the maxi-

mum number of I/O operations per second available to real-time applications for the stripe group using the **Quality of Service (QoS)** API. There is also the ability to specify I/Os that should be reserved for applications not using the QoS API. Realtime I/O functionality is off by default.

A stripe group contains one or more disks on which to put the metadata/journal/userdata. The disk has an **index** that defines the ordinal position the disk has in the stripe group. This number must be in the range of zero to the number of disks in the stripe group minus one, and be unique within the stripe group. There must be one disk entry per disk and the number of disk entries defines the stripe depth. For more information about disks, see the DISK DEFINITION section above.

NOTE: The **StripeClusters** variable has been **deprecated**. It was used to limit I/O submitted by a single process, but was removed when asynchronous I/O was added to the file system.

NOTE: The **Type** variable for Stripe Groups has been **deprecated**. Several versions ago, the **Type** parameter was used as a very course-grained affinity-like control of how data was laid out between stripe groups. The only valid value of **Type** for several releases of SNFS has been **Regular**, and this is now deprecated as well for the XML configuration format. **Type** has been superseded by **Affinity**.

FILES

/usr/cvfs/config/.cfgx*
/usr/cvfs/config/.cfg*

SEE ALSO

snfs.cfgx(5), **snfs.cfg**(5), **sncfgedit**(8), **cnvt2ha.sh**(8), **cvfs**(8), **cvadmin**(8), **cvlabel**(8), **snldapd**(8), **cvmkdir**(1), **cvmkfile**(1), **acldomain**(4), **ha_peer**(4), **mount_cvfs**(8), **sgmanage**(8), **sgoffload**(8)

" "StorNext File System"

NAME

snfs_dlc_network.json − StorNext File System DLC Server Filter Control

SYNOPSIS

Linux only.

/usr/cvfs/config/snfs_dlc_network.json

DESCRIPTION

The *StorNext File System* DLC capability will, by default, inform all DLC clients of all DLC gateways available. The clients will then perform a connectivity test and establish connections with the gateways they have visibility to. In a large scale deployment, this can lead to too many different servers being available for efficient operation.

The **snfs_dlc_network.json** file on the MDC for the file system is used to limit the set of servers visible to each client. When a client connects, the set of DLC servers it is told about is a subset of the possible ones. This can be used to define a configuration where clients still have sufficient bandwidth and redundancy to perform I/O, while avoiding large meshes of connections which are excessive.

The file defines groups of clients and gateway addresses they are allowed to see. Clients and gateways are defined in terms of an IP address and optional netmask. When a client requests a set of gateways, it's metdata network address is used to find a matching record in the configuration, the current set of gateways available which match the gateway addresses in the record are passed back to the client.

If no existing gateways match, the client will have no data path, if there is no record for the client then the client will also have no data path.

An example file looks like

```
{
    "filter": [
        {
            "clients": [
                {
                    "addr": "10.65.100.50"
                },
                {
                    "addr": "10.65.100.52"
                }
            ],
            "gateways": [
                {
                    "addr": "192.168.100.1",
                    "netmask": 24
                }
            ]
        },
        {
            "clients": [
                {
                    "addr": "10.65.100.1",
                    "netmask": 20
                }
            ],
            "gateways": [
                {
```

```
                        "addr": "10.65.100.1",
                        "netmask": 20
                    }
                ]
            }
        ]
    }
```

The complete set of filters for assigning gateways to clients is the defined by the **filter** set. Each record defines a group of clients and the associated gateways. The clients are specified in terms of their metadata network addresses, the gateway addresses are in terms of the networks used for DLC data traffic. The first set which matches a client's address is used.

In the example, two specific clients are referenced, there is no netmask, so an exact address comparison is used. These will be given access to all available gateways on the 192.168.100.0/24 network.

All clients on the 10.65.100.1/20 network will be given access to all the gateways on this network.

Note that the first record is providing special behavior for these two clients which overrides the behavior for the second set.

A special **addr** value of *any* can be used for a client or gateway address. This will match any client or gateway and can be used to set a catch all rule. It would typically be used as the last filter item when the client address is any as it will always match and no further rules will be applied.

The **snfs_dlc_network.json** is read by fsm processes at startup and at regular intervals thereafter if it has changed. A configuration change will be applied live without restarting services or remounting clients. On an HA configuration the file is propogated from the primary to the secondary by the system.

FILES

/usr/cvfs/config/snfs_dlc_network.json

NAME

snfs_metadata_network_filter.json − StorNext Metadata network filter configuration file

SYNOPSIS

/usr/cvfs/config/snfs_metadata_network_filter.json

/usr/cvfs/config/snfs_metadata_network_filter.d/filter_files

DESCRIPTION

The *StorNext File System* (*SNFS*) **snfs_metadata_network_filter.json** file provides a way to limit metadata traffic to a subset of the interfaces or IP addresses configured on the MDC.

SYNTAX

The *snfs_metadata_network_filter.json* file follows the JSON schema. Each of the attribute-value pairs occupies one line and is terminated by a comma, save for the final attribute-value pair. Currently, there is a single section for filters, which is a json array and is thus enclosed by a pair of square brackets. The filter elements are either an IP address and network mask length or an interface name substring. No comments of any kind are allowed. Refer to this man page's **EXAMPLE** section for a visual representation of the described format.

The directory *snfs_metadata_network_filter.d* can be created and used to add a set of extra configuration files with same syntax, all the files are combined to create a composite filter.

The configuration file will be loaded by the **fsmpm**(8) process when it receives a **SIGHUP** signal or when StorNext services are re-started. The result of the parse of *snfs_metadata_network_filter.json* will be logged to the */usr/cvfs/debug/nssdbg.out* log file. This includes the results from a successful parse or syntax or symantic errors.

"**filter_list**"
: This is the array containing filter elements, that is, addresses or interface names that are not to be used for metadata traffic.

"**ip_address**"
: This is a string containing the IPv4 of IPv6 address to be filtered.

"**netmask_length**"
: This is an integer which applies to the corresponding ip_address. All addresses matching the IP address bits up to the network mask length will be filtered. If no **netmask_length** is specified for a given address, only the exact address will be filtered, essentially a netmask of 32 for IPv4 addresses and 128 for IPv6.

"**interface_name**"
: This is a string containing the first characters of the interface names that are to be filtered. For example "em1" will match all interfaces starting with that string. The string "eth0:" will match virtual sub-interfaces such as "eth0:0" but will not filter interface "eth0".

NOTE: This configuration file only needs to exist where the **fsm**(8) daemon will run, typically the metadata controllers. The **snfs_metadata.conf** file is only supported on Linux systems.

EXAMPLE

The following is an example of the **snfs_metatdata_network_filter.json** file.

```
{
   "filter_list": [
      {
         "ip_address": "12::",
         "netmask_length": 64
      },
      {
         "ip_address": "192.168.200.0",
         "netmask_length": 24
      },
```

```
        {
          "ip_address": "192.168.201.1"
        },
        {
          "ip_address": "192.168.202.0",
          "netmask_length": 24
        },
        {
          "interface_name": "em1"
        },
        {
          "interface_name": "eth0:"
        }
      ]
    }
```

FILES

/usr/cvfs/config/snfs_metadata_network_filter.json

SEE ALSO

fsnameservers(4), **fsm**(8), **fsmpm**(8), **gethostname**(2)

NAME

snfs_ras − StorNext File System RAS Events

DESCRIPTION

The StorNext File System supports logging and delivery of specific Reliability/Availability/Serviceability (RAS) events. When a RAS event occurs, an entry is added to the event log (*/usr/cvfs/ras/raslog*) on the Name Server coordinators (see **fsnameservers**(4)). It is also possible to configure the coordinators to automatically send e-mail for RAS events or pass the RAS event to any script or executable as described in the NOTES section below. In an environment that includes the StorNext Management Suite (SNMS), RAS events also generate Service Request RAS tickets that can be viewed through the SNMS GUI by selecting "System Status" from the Service menu.

EVENTS

The following list contains the currently supported events.

Event:

SL_EVT_NO_RESPONSE (Not responding)

Occurs:

When a reply from the FSM is delayed.

Example detail:

client2.foo.com (kernel): Timeout while attempting to force data flush for file ´file1´ (inode "895468127") for file system ´snfs1´ on host client1. Allowing host client2 to open file. This may cause data coherency issues. The data path for host client1 should be inspected to confirm that I/O is working correctly.

Suggested Action(s):

Confirm that the data path is working properly on the system specified in the event detail.

Event:

SL_EVT_INVALID_LABEL (Label validation failure)

Occurs:

When client-side label verification fails.

Example detail:

wedge.foo.com (kernel): fs snfs1: disk label verification for ’CvfsDisk0’ failed on rawdev /dev/rhdisk0 blkdev /dev/hdisk0 (HBA:2 LUN:0)

Suggested Action(s):

Check for corrupt, incorrect, or missing labels using the cvlabel command. Also inspect system logs for I/O errors and check SAN integrity.

Event:

SL_EVT_DISK_ALLOC_FAIL (Failed to allocate disk space)

Occurs:

When a disk allocation fails due to lack of space.

Example detail:

fsm[PID=1234]: fs snfs1: Disk Allocation failed

Suggested Action(s):

Free up disk space by removing unnecessary disk copies of files, or add disk capacity.

Event:

SL_EVT_COMM_LUN_FAIL (LUN communication failure)

Occurs:

When an I/O error occurs in a multi-path environment, causing a path to become disabled.

Example detail:
wedge.foo.com (kernel): Disk Path ’/dev/rdisk0’ (HBA:2 LUN:0) used by file system snfs1 temporarily disabled due to I/O error

Suggested Action(s):
Check system and RAID logs for SAN integrity.

Event:
SL_EVT_IO_ERR (I/O Error)

Occurs:
When an I/O error is detected by the FSM or a client.

Example detail:
wedge.foo.com (kernel): fs snfs1: I/O error on cookie 0x12345678 cvfs error ’I/O error’ (0x3)

Suggested Action(s):
Check LUN and disk path health, as well as overall SAN integrity. Also inspect the system logs for driver-level I/O errors.

Event:
SL_EVT_SHUTDOWN_ERR (Error shutting down)

Occurs:
When a problem occurs when unmounting or shutting down a file system.

Example detail:
wedge.foo.com cvadmin[PID=1234]: fs snfs1: could not unmount all cvfs file systems

Suggested Action(s):
Inspect the file system and system logs to determine the root cause.

Event:
SL_EVT_INITIALIZATION_FAIL (Initialization failure)

Occurs:
When the FSM or fsmpm process fails to start up or a mount fails.

Example detail:
wedge.foo.com fsm[PID=1880]: fs snfs1: FSM Initialization failed with status 0x14 (missing disk(s))

Suggested Action(s):
Correct the system configuration as suggested by the event detail, or examine the system logs to determine the root cause. If the detail text suggests a problem with starting the fsmpm process, run "cvlabel -l" to verify that disk scanning is working properly.

Event:
SL_EVT_LICENSE_FAIL (License failed)

Occurs:
When a StorNext license expires.

Example detail:
wedge.foo.com fsm[PID=2588]: fs snfs1: StorNext Client lady.foo.com (1372A4B126) license has expired. All further client operations not permitted.

Suggested Action(s):
Contact the Quantum Technical Assistance Center to obtain a valid license.

Event:
SL_EVT_LICENCE_REQUIRED (License Required)

Occurs:
When a StorNext license will expire within 48 hours.
When a StorNext trial period has expired, or will expire within 7 days.

Example detail:

wedge.foo.com fsm[PID=3325]: fs snfs1: Please update license file! StorNext File System license will expire on Tue Dec 12 11:28:26 CST 2006
luke.foo.com fsm[PID=4567]: fs snfs2: Trial of StorNext software ends in 3 days on Monday May 2 12:13:45 CDT 2022. Please obtain a valid product key.
luke.foo.com fsm[PID=4567]: fs snfs2: Trial of StorNext software ended on Monday May 2 12:13:45 CDT 2022. Please obtain a valid product key.

Suggested Action(s):

Contact the Quantum Technical Assistance Center to obtain a valid license or product key.

Event:

SL_EVT_FAIL_OVER (Fail-over has occurred)

Occurs:

During during FSM fail-over.

Example detail:

wedge.foo.com fsm[PID=3325]: fs snfs1: Fail-over has occurred: Previous MDC ’172.16.82.71’ Current MDC ’172.16.82.78’

Suggested Action(s):

Inspect the system log and the FSM cvlog to determine the root cause.

Event:

SL_EVT_META_ERR (Metadata error)

Occurs:

When the FSM detects a metadata inconsistency.

Example detail:

wedge.foo.com fsm[PID=3325]: Invalid inode lookup: 0x345283 markers 0x3838/0x3838 gen 0x2 next iel 0x337373

Suggested Action(s):

Check SAN integrity and inspect the system logs for I/O errors. If the SAN is healthy, run cvfsck on the affected file system at the earliest convenient opportunity.

Event:

SL_EVT_BADCFG_NOT_SUP (Configuration not supported)

Occurs:

When an FSM configuration file is invalid or missing. Also occurs when the total number of FSMs running on metadata controllers under a fsnameservers domain exceeds the capacity limit of the heartbeat protocol.

Example detail:

wedge.foo.com fsm[PID=3832]: fs snfs1: Problem encountered parsing configuration file ’snfs1.cfgx’: There were no disk types defined

Example detail:

wedge.foo.com fsm[PID=3832]: fs snfs1: Problem encountered parsing configuration file ’snfs1.cfgx’: There were no disk types defined

Suggested Action(s):

Verify that a valid file system configuration file exists for the specified file system. Also check the system logs for additional configuration file error details. The capacity of the heartbeat protocol is a function of the number of FSMs and the length of the file-system names. The maximum number of FSMs can be configured by limiting file-system names to seven characters or fewer, and by ensuring that all clients are upgraded to use the expanded heartbeat-protocol packet size.

Event:

SL_EVT_TASK_DIED (Process/Task died, not restarted)

Occurs:
When the FSM or fsmpm unexpectedly exists.

Example detail:
wedge.foo.com fsm[PID=5543]: fs snfs1: PANIC: Alloc_init THREAD_MUTEX_INIT alloc_space lock

Suggested Action(s):
Check the FSM logs and system logs to determine the root cause. If possible, take corrective action. If you suspect a software bug, contact the Quantum Technical Assistance Center.

Event:
SL_EVT_SYS_RES_FAIL (System resource failure)

Occurs:
When a memory allocation fails.

Example detail:
wedge.foo.com fsm[PID=9939]: Memory allocation of 65536 bytes failed: file 'disks.c' line 256

Suggested Action(s):
Determine the cause of memory depletion and correct the condition by adding memory or paging space to your system. If SNFS is using excessive amounts of memory adjusting the configuration parameters might resolve the problem. For information about adjusting parameters, refer to the Release Notes, the **snfs_config**(5) and **mount_cvfs**(8) man pages, and the SNFS Tuning Guide.

Event:
SL_EVT_SYS_RES_CRIT (System resource critical)

Occurs:
When the filesystem is running out of space.

Example detail:
wedge.foo.com fsm[PID=3947]: fs 'snfs1': System over 10% full

Suggested Action(s):
Add additional storage or reduce file system usage. If the message indicates metadata stripe groups are full, add additional metadata storage.

Event:
SL_EVT_SYS_RES_WARN (System resource warning)

Occurs:
When a condition such as a fragmented file is detected.

Example detail:
wedge.foo.com fsm[PID=5213]: fs 'snfs1': Excessive fragmentation detected in file 'foo'

Suggested Action(s):
If fragmentation has been detected, consult the **snfsdefrag**(1) man-page for instructions on performing fragmentation analysis and defragmenting files. Free space defragmentation for the file system as a whole may also be performed using the **sgdefrag**(8) utility.

Event:
SL_EVT_CONNECTION_FAIL (Connection rejected)

Occurs:
When the FSM rejects a client connection attempt (other than for a licensing issue).

Example detail:
wedge.foo.com fsm[PID=9939]: The client node [Node 13] does not support cluster-wide central control feature, please upgrade the client to a newer version

Suggested Action(s):
Check the system logs to determine the root cause.

Event:
SL_EVT_LUN_CHANGE (LUN mapping changed)

Occurs:
When an fsmpm disk scan detects a change in an existing path.

Example detail:
wedge.foo.com fsmpm[4872]: Disk 'CvfsDisk0' is no longer accessible as '/dev/hdisk0' -- path may have been assigned to another device.

Suggested Action(s):
If the LUN mapping change is unexpected, run the cvadmin "disks" and "paths" commands to confirm that all LUN paths are present. Also check SAN integrity and inspect the system logs to determine the root cause.

Event:
SL_EVT_JOURNAL_ERR (Journaling error)

Occurs:
When journal recovery fails.

Example detail:
wedge.foo.com fsm[5555]: fs snfs1: Journal error: Journal_recover: Journal_truncate failed

Suggested Action(s):
Contact the Quantum technical assistance center and open a service request.

NOTES

To reduce overhead, some types of RAS events are throttled so that only one event is generated per hour per system.

To quickly set up RAS event e-mail notification, use the following steps on each of the Name Server coordinators:

1. copy */usr/cvfs/examples/rasexec.example* to */usr/cvfs/config/rasexec*

2. edit */usr/cvfs/config/rasexec* and modify RAS_EMAIL as appropriate.

After setting up notification, you may prefer to exclude certain events to reduce the number of e-mail messages you receive. This can be accomplished by adding events that you wish to skip to the RAS_EXCLUDE variable in */usr/cvfs/config/rasexec*.

To call an arbitrary executable for each RAS event, simply invoke a command from */usr/cvfs/config/rasexec*. The first argument passed to rasexec is the event (e.g. SL_EVT_IO_ERR) and the second is the detail string and these can be passed into your program. Be careful when choosing the command to run so that it does not hang or cause other ill effects.

FILES

/usr/cvfs/ras/raslog
/usr/cvfs/ras/OLDraslog
/usr/cvfs/config/rasexec
/usr/cvfs/examples/rasexec.example

SEE ALSO

fsnameservers(4), **fsm**(8), **cvfs_failover**(8), **snfsdefrag**(1), **sgdefrag**(8)

NAME

snfs_rest_config.json − StorNext File System REST Configuration file

SYNOPSIS

/usr/cvfs/config/snfs_rest_config.json

DESCRIPTION

The *StorNext File System* (SNFS) *snfs_rest_config.json* file is a JSON formatted configuration file used to control the behavior of REST services on REST capable versions of StorNext and to manage the behavior of REST services in SNFS processes.

The REST interface is for internal StorNext use only, thus there is no support or documentation of the REST interface itself.

A version of the file is installed with the software and does not need to be changed under typical operating conditions.

The *minport* and *maxport* may be useful in a fire walled environment. The data timeout may be useful for long running requests.

Process specific definitions are applied to specific commands and daemons. The process name is the json object identifier which encapsulates the command or process specific options.

Process names currently supported are:
fsm
fsmpm
qbmanage
sgmanage
snhistory
snrecover
snquota
SM_metadb
fsrestd
sntierd

The process name sgmanage includes sgdefrag and sgoffload commands. All processes support the keywords data_timeout and conn_timeout. The process fsmpm supports the enable_server and rest_job_keep_days keywords. The process fsm supports the enable_rest keyword. The processes fsm and fsmpm support enable_proxy.

SYNTAX

The *snfs_rest_config.json* file follows the JSON schema. Each of the attribute-value pairs occupies one line and is terminated by a comma, save for the final attribute-value pair. Items used for processes appear in a process specific section identified by the process name. No comments of any kind are allowed. Refer to this man page's **EXAMPLE** section for a visual representation of the described format.

General keywords are:

"**conn_timeout**": *time*
: Specifies the number of milliseconds before a connect request will timeout. *conn_timeout* defaults to 5000 (5 seconds).

"**data_timeout**": *time*
: Specifies the number of milliseconds before a connection will timeout. *data_timeout* defaults to 10000 (10 seconds), except for the fsm process, which has a default of 900000 (900 seconds).

"**enable_proxy**": *true|false*
: Currently unsupported. Enables use of a proxy interface. *enable_proxy* defaults to false.

"**enable_rest**": *true|false*
: Enables use of the REST interface. *enable_rest* must be included as true for use of any services. Default value is false, but is specified as true in the installed file.

"**enable_server**": *true|false*
: Currently unsupported. Enable rest control over some file system admin features. *enable_server* defaults to false, but is specified as true in the installed file.

"**maxport**": *port_value*
: Specifies the top number for the range of allowed ports. Use in conjunction with *minport*. *maxport* defaults to 0 (no range).

"**minport**": *port_value*
: Specifies the bottom number for the range of allowed ports. Use in conjunction with *maxport*. *minport* defaults to 0 (no range).

"**port**": *port*
: Controls the port used for http communication. *port* is process specific only. The default value is unspecified.

"**proxy_admin**": "*proxy_admin*"
: Deprecated, see the snrest(8) command. If a proxy web server is used, URL of server to which to connect. However, this value is ignored if an api gateway is configured with the snrest(8) command. *proxy_admin* defaults to null.

"**proxy_gw**": "*proxy_gw*"
: Deprecated, see the snrest(8) command. If a proxy web server is used, URL for incoming requests to that server. However, this value is ignored if an api gateway is configured with the snrest(8) command. *proxy_gw* defaults to null.

"**registration_name**": "*registration_name*"
: Overrides the use of hostname for a url published by the system. This is expected to be a name which can be mapped to the location of the SNFS host from external locations. For use in environments where the host name does not resolve to a routable address. The *registration_name* defaults to null.

"**ssl**": "*ssl*"
: Controls the use of secure socket negotiation parameters. In particular the minimum TLS version which will be allowed and the set of ciphers allowed. *ssl* has default values, but values are specified in the installed file for use.

"**ssl_port**": *ssl_port*
: Controls the port used for https communication. *ssl_port* is process specific only and defaults to unspecified.

"**rest_job_keep_days**": *days*
: Specifies how long a background job's log and state are to be retained. A background job can be submitted for certain potentially long-running tasks such as file system check.

"**proxy_retry_seconds**": *seconds*
: Specifies how long to wait between retries when registration with the proxy web server fails. The default value is 300 (5 minutes).

EXAMPLE

The following is the installed **snfs_rest_config.json** file. This file should not need modification.

```
{
    "enable_rest"   : true,
    "enable_server" : true,
    "ssl" : {
        "ssl_enabled"  : true,
        "min_tls"      : "1.0",
        "ciphers"      : "HIGH:!aNULL:!MD5:!RC4:!3DES",
        "pemfile"      : "/usr/cvfs/config/certs/server.crt",
        "keyfile"      : "/usr/cvfs/config/certs/server.key",
```

```
            "ssl_timeout"  : 124800,
            "verify_peer"  : false,
            "verify_depth" : 100,
            "cafile"       : "",
            "capath"       : ""
        },
        "rest_job_keep_days" : 120,
        "proxy_retry_seconds" : 300,
        "fsm" : {
            "data_timeout" : 900000,
            "conn_timeout" : 5000
        },
        "fsmpm" : {
            "data_timeout" : 10000,
            "conn_timeout" : 5000
        },
        "qbmanage" : {
            "data_timeout" : 10000,
            "conn_timeout" : 5000
        },
        "SM_metadb" : {
            "data_timeout" : 300000,
            "conn_timeout" : 5000
        },
        "fsrestd" : {
            "data_timeout" : 300000,
            "conn_timeout" : 5000
        },
        "sgmanage" : {
            "data_timeout" : 60000,
            "conn_timeout" : 5000
        },
        "snhistory" : {
            "data_timeout" : 300000,
            "conn_timeout" : 5000
        },
        "sntierd" : {
            "data_timeout" : 300000,
            "conn_timeout" : 5000
        }
    }
```

LIMITATIONS

Only the **Linux** and **Unix** platforms are supported with snrest commands and the associated snfs_rest_config.json configuration file

FILES

/usr/cvfs/config/snfs_rest_config.json

SEE ALSO

fsm(8), **fsmpm**(8), **qbmanage**(8), **sgdefrag**(8), **sgmanage**(8), **sgoffload**(8), **snhistory**(8)

NAME

snhamgr − StorNext High Availability Manager

SYNOPSIS

snhamgr [**-m**|**-v**] **--primary**

snhamgr [**-m**|**-v**] **status**

snhamgr [**-m**|**-v**] **start**

snhamgr [**-m**|**-v**] **stop**

snhamgr [**-m**|**-v**] **config**

snhamgr [**-m**|**-v**] **peerdown**

snhamgr [**-m**|**-v**] **peerup**

snhamgr [**-m**|**-v**] **force smith**

snhamgr [**-m**|**-v**] **clear**

snhamgr [**-m**|**-v**] **mode=**{**default**|**single**|**config**|**locked**}

DESCRIPTION

The StorNext High Availability System (HA) protects against data corruption from so-called *Split Brain Scenario*, which is when redundant servers are writing to common storage in an uncontrolled way. In normal operation, the StorNext product will not allow uncontrolled writes, but some types of hardware or software failures can make the system vulnerable. HA provides protection against *split-brain scenario* by: 1) monitoring for situations where the Primary server must relinquish control according to strict timing rules and, 2) resetting the primary server before the secondary server could complete usurpation of control. Adding an HA option to the configuration of each SNFS file system activates the HA protections. Creating the */usr/cvfs/config/ha_peer* file with the numerical IP address of the peer metadata servers (MDC) allows communication between fsmpm processes on the HA MDCs that reduces the chance of unnecessary HA Reset incidents. These configuration changes are sufficient to establish HA protected redundant-server configurations that: 1) do not include the Storage Manager, and 2) do not use the StorNext GUI. Configurations that use those features have more complex management requirements that are met by the HA Manager Subsystem.

The HA Manager Subsystem automates operation of the HA system for configurations that include the HaShared file system. It manages the HA operating *mode* and *status* on each of the two servers. These two pairs of values describe the cluster state, which is essential for controlling the HA system so that StorNext can be started, operated, and stopped for administrative activities while continuously maintaining protection against *split-brain scenario*. Rules built into the HA Manager prevent combinations of modes that could put the cluster at risk. Rules built into StorNext components' control scripts allow their components to start only when it is safe.

Warning: Modifications to StorNext software that compromise the HA operating rules could result in data corruption from *split-brain scenario*.

Modes are set on the local server, and can be queried by either server. They are stored in a file to allow the modes to continue across reboots. The following are the modes and capabilities they allow:

default HA reset capability is enabled. When the conditions for a potential *split-brain scenario* are detected, system-kernel software performs a software-driven hardware reset, so-called *Shoot Myself In The Head* (SMITH). It has this name because of the way that resets are invoked autonomously. When a server cannot communicate with its peer, the peer is assumed to be running in *default* mode. StorNext configurations without the HaShared file system operate without the HA Manager in *default* mode at all times.

CAUTION: When transitioning the primary server from *single* mode to *default* mode, verify that the HA shared file system is activated and that it has access to all of its disks. To do this, run the cvadmin *fsmlist* command and verify that the state is not BLOCKED for the HA shared file system FSM. If it is, then attempt to refresh the network paths to the meta-data disks using the cvadmin

disks refresh command. See the cvadmin man page for more detailed information on cvadmin commands.

locked HA reset capability is disabled. StorNext components are prevented from running. This mode can be entered and exited automatically by scripts, so the server must continue running and communicating while it is in this mode. It allows the peer server to operate StorNext with HA reset capability disabled in *single* or *config* mode to allow administrative tasks without the risk of *split-brain scenario*. When requested, the *locked* server must communicate its status, so *locked* mode is not effective for servers that are rebooting or powered down.

CAUTION: If a secondary MDC in *locked* mode is rebooted or powered down while the primary MDC is in *config* or *single* mode, snhamgr may detect that the HA cluster is in an invalid state. If it does, it will attempt to safeguard the HA cluster by stopping StorNext on the primary MDC and putting it into *default* mode. To return the HA cluster back to the config state, check to ensure that the secondary MDC is either powered off and peerdown is set on the primary MDC or that the secondary MDC is running and its snhamgr mode is set to *locked*. Once this is verified, restart StorNext on the primary MDC and then set its snhamgr mode back to *config*.

peerdown

The peer server is out of service. This mode is equivalent to *locked*, but is certified by an administrator rather than reported by a server to its peer. The attribute is stored locally. This mode is used when the peer server is powered down.

In the event that the peer returns to service and begins to communicate, the assertion that the peer is down becomes false. Immediate action may be taken by the local server to transition itself to a safe operating mode, which could trigger an HA reset. The best practice is to power off the server or deinstall StorNext before setting *peerdown* mode, and to unset the mode before powering on the server.

Note: Issuing the *peerdown* command should only be done interactively, not by scripts, which might lead to an operational mistake that produces an avoidable HA reset.

single HA reset capability is disabled. The peer server must be *peerdown* (recommended) or *locked*. The *single* mode is useful when an HA reset would be unhelpful because there is no peer server to take over the cluster. The peer server must be *peerdown* (recommended) or *locked*. An HA cluster can be operated non-redundantly for an indefinite period in the *single* mode. A server can transition from default to single and from single to default without restarting StorNext.

config HA reset capability is disabled. The peer server must be *peerdown* or *locked*. The *config* mode is meant for use when: 1) making changes to configuration files, and 2) stopping and starting StorNext processes to apply those changes. When transitioning out of *config* mode into *default* mode, StorNext must be stopped, which ensures that all processes are started in the correct sequence.

OPTIONS

-m Output status in a format that may be easier to use in scripts.

-v Increase the verbosity.

COMMANDS

--primary

Set the primary status for the local server. This is used by the */usr/cvfs/lib/snactivated* script after activating the *HaShared* file system. **Note: This command is exclusively for internal use by HA system software.** Use of this command outside of that context could undermining HA protection and allow corruption of the SNSM database. The command cannot be run from a terminal as a precaution.

status Return cluster modes and operating statuses. All commands return status; this one does nothing else.

start Stop each server when there is a need and transition both servers to default mode, then bring up the local server first followed by the peer server so that the local server becomes *primary* and the

peer server becomes *secondary*.

stop Safely stop both servers in the cluster without incurring a HA reset. The secondary server is placed in *locked* mode, which stops StorNext on that server, then the *primary* server is placed in *config* mode and stopped, and then both servers are put in *default* mode with StorNext stopped.

config First, check that the peer server is in *locked* or *peerdown* mode. Then, place the local server in *config* mode. The command must be run on the *primary* server or either server when CVFS is stopped on both.

peerdown
Certify that the peer server is powered off. **Note: This command should never be run from a script.** Misuse could lead to an unnecessary HA reset or data corruption.

peerup Undo the *peerdown* mode. The command will fail if the local mode is *config* or *single*. Run this command before powering on the peer server. The local server will assume the peer is in *default* mode until the peer starts *snhamgr_daemon* communications.

force smith
Trigger an immediate HA reset if the local server is in *default* mode. This command is meant for use in health-monitoring scripts or in testing the capability of a system to issue the HA reset. The command is two words to make accidental firing less likely.

Note: It is not recommended to use this command to administratively failover a system in a production environment. The preferred method to gracefully failover the primary system to its peer node is to simply stop CVFS and restart it after the peer has become primary. For example, on the node that is primary run:

```
# service cvfs stop
```

Wait for the peer to become primary, then run:

```
# service cvfs start
```

clear Remove the file referenced by the HA_IDLE_FAILED_STARTUP environment variable. Run this after correcting any conditions that caused the previous failure of StorNext startup scripts.

mode=<*modeval*>
Set the mode of the local server to *modeval*. The mode is stored on the local server so that it persists across reboots.

config Set the mode to *config*. The peer server must be in *locked* or *peerdown* mode.

default Set the mode to *default*. The peer server can be in any mode.

locked Set the mode to *locked*. The peer server can be in any mode.

single Set the mode to *single*. The peer server must be in *locked* (recommended) or *peerdown* mode.

USAGE

The *snhamgr* command can be used in scripts or interactively. It is multi-threaded to allow simultaneous invocations. A simple status command takes about one second to complete, and involves running a StorNext script on each server to collect operational status. The stop, start, and config cluster commands can take several minutes to complete, and involve running several StorNext scripts on each server. Interrupting the *snhamgr* command during the execution of one of these commands does not stop the command, which may continue to completion in the *snhamgr_daemon*.

Care should be taken to avoid simultaneous contradictory actions that could leave one or both servers in need of manual intervention to restore proper StorNext production operation.

After completing a command, *snhamgr* prints out cluster status in one of two formats. The default format is easier for humans to read. The *-m* option produces a one-line output that may be easier to use in scripts.

In the unlikely event that the *snhamgr_daemon* is not running, the status output displays *error* for all of its

fields. The best action in that event is to determine the reason for the stoppage of the *snhamgr_daemon*, to correct the problem, and to restart the daemon with: **service snhamgr start**.

SEE ALSO

cvfs(8), **ha_peer**(4), **snhamgr_daemon**(8)
Quantum StorNext User's Guide

Code start macro

NAME

snhamgr_daemon − StorNext HA Manager daemon

SYNOPSIS

service snhamgr {**start**|**stop**|**restart**|**status**}

DESCRIPTION

The *snhamgr_daemon*, together with its **snhamgr**(8) command-line interface (CLI), automates some operations of high-availability (HA) redundant StorNext HA servers. It enforces rules of the HA Reset system for preserving the HA protections against *Split Brain Scenario* while allowing HA resets to be disabled during administrative activities.

The *snhamgr_daemon* process runs continuously on servers after they have been converted to HA with the **cnvt2ha.sh**(8) script. These daemons provide a distributed multi-tasking system that measures the state of the cluster and communicates with the *snhamgr* CLI to report and optionally change the state of the cluster. StorNext component scripts make calls to the *snhamgr* CLI at control points so that HA operating rules are upheld.

The *snhamgr_daemon* is started by the **init.d**(7) mechanism before any other StorNext software. Likewise, it is stopped after all other StorNext software on shutdown of the server operating system. This allows it to communicate status to its peer and be called by StorNext scripts whether StorNext is running or stopped. The daemon has a companion *watcher* process that attempts to restart the daemon if it stops.

At startup, the *snhamgr_daemon* reads the */usr/cvfs/install/.ha_mgr* file to determine the operational mode of the local server and to determine if the peer server has been certified as *peerdown* by an administrator. The file is updated when commands are issued to change the server's modes. This allows the HA modes to persist across reboots and StorNext restarts.

The daemon runs StorNext component scripts to measure or change the operational status of the server at startup, when a request arrives from the *snhamgr* CLI, or when a request arrives from the peer server's *snhamgr_daemon*. The *snhamgr_daemon* can take actions to at these junctures to enforce the HA operating rules. Otherwise, the *snhamgr_daemon* is idle.

Output traces for commands run by StorNext are captured in the */usr/cvfs/debug/hamgr_cmds_trace* file. Information in the file can be difficult to read because scripts are often run in parallel and the output of several commands can be interlaced. When the file is two megabytes at the start of a command, the file is moved to */usr/cvfs/debug/hamgr_cmds_trace.old* and a new file is created.

ENVIRONMENT

The following environment variables are defined in */usr/adic/.profile* and regulate the operation of *snhamgr_daemon*.

SNSM_HA_CONFIGURED

The *cnvt2ha.sh* script creates the file referenced by the *SNSM_HA_CONFIGURED* variable. When the file does not exist, the *snhamgr_daemon* exits, which means that the daemon only operates on systems having an *HaShared* file system (see the **snfs_config**(5) and **cnvt2ha.sh**(8) man pages for details). The *snhamgr* command-line interface is still used by component scripts in HA configurations without an **HaShared** file system, but it: 1) does not communicate between servers, 2) reports both servers as being in *default* mode, and 3) always reports the statuses of the local server as *unconfigured* and the peer server as *unknown*.

The value of SNSM_HA_CONFIGURED is */usr/adic/install/.snsm_ha_configured*. **The file should not be modified except by Quantum support personnel.**

HA_IDLE_FAILED_STARTUP

When an algorithm in the startup scripts detects a failure to complete the previous startup, it creates the file referenced by this variable to prevent the looping of failed startups. When this file exists, StorNext will not start and the *snhamgr_daemon* reports the server mode as *failed_startup* and the status as *unknown* for both local and peer servers. The value of HA_IDLE_FAILED_STARTUP is */usr/cvfs/install/ha_idle_failed_startup*.

HAMGR_MAX_LOGSIZE

Controls the maximum size that the **ha_mgr.out** file can reach before a new log file will be started. The value is a number followed by an optional suffix (**K** for Kilobytes, **M** for Megabytes, **G** for Gigabytes), or the string **unlimited**, indicating that the file can grow without bound. The default is 1M and the minimum is 64K.

HAMGR_MAX_LOGFILES

Controls the number of old **ha_mgr.out** log files that will be saved by **fsmpm** when the current log file reaches its maximum size. There must be at least one, and the default is 4.

FILES

/usr/cvfs/install/.ha_mgr

Machine-written, human-readable value and timestamp for the local server mode and the remote server's potential *peerdown* mode. The timestamps tell when the modes were last changed.

/var/lock/subsys/snhamgr

This file prevents the running of more than one daemon, and provides an open-file link for stopping the daemon and checking its status.

/usr/cvfs/debug/ha_mgr.out

Log file for the *snhamgr_daemon*.

/usr/cvfs/debug/hamgr_cmds_trace

Trace file for commands and scripts invoked by the *snhamgr_daemon*.

SEE ALSO

cvfs(8), **snhamgr**(8), **cnvt2ha.sh**(8), **snfs_config**(5), **chkconfig**(8), **init**(8)

NAME

snhistory − Display StorNext File System history

SYNOPSIS

snhistory [**-F** *FsName*] [**-d** *pathname*] [*options*]

DESCRIPTION

The **snhistory** program provides the capability to query the *metadata archive* for the history of file system activity that has transpired between two given points of time. It requires that file system configuration parameter metadataArchive is set to "true", that metadataArchiveDays is set to a non-zero value, and that metadataArchiveSearch be set to "true". **snhistory** can return the history for the entire file system, for any directory and all of its files and sub-directories, or for a single file. The output format is standard JSON by default but can be changed to compact JSON, plain text, or compact text. In the compact text format, the file name is listed without any event association.

The **snhistory** program can be run in one of two modes: *event-history* mode or *changed-file* mode. By default, **snhistory** runs in event-history mode. To run in changed-file mode, the user must use either the -l or -L option.

In event-history mode, **snhistory** generates a list of file system events for all files and directories in ascending chronological order. The type of events generated are CREATE, REMOVE, RENAME, MODIFY, LINK, LINKCOUNT, CHMOD, CHOWN, and CHGRP.

In changed-file mode, **snhistory** generates a list of pathnames for all files in the file system or specified subdirectory that have been either created, deleted, modified, or renamed. The generated list is not in any specific order. For all output formats except compact text, each file listed will be displayed with the most "significant" event for the given time period along with the current size of the file. The most significant events are creates, removes, and renames. So, if a file was modified several times and then deleted during the given time period, the output from **snhistory** will show that the file was removed.

OPTIONS

−F *FsName*

This is the name of the StorNext file system. Using this option will cause **snhistory** to retrieve the activity history from the file system's active metadata archive.

−D *debug_level*

This option is for internal use only.

-C Specifies that the output format should be compact JSON or compact text if the -T option is used. The compact text format only prints the file path names and these path names are relative to the directory specified by the -p option, if present.

-T Specifies that the output format should be plain text instead of JSON.

-L Run **snhistory** in *changed-file* mode. **snhistory** will generate a pathname for all files that have been created, removed, renamed, or modified within the specified start and end times.

-l Same as the -L option except that pathnames will be as of the end time as specified by the -e option or the end TID as specified by the -t option. For example, if a file foo was created in directory foodir and then moved to directory bardir then using this option will cause the corresponding file CREATE event to show a pathname of bardir/foo. In essence, this says that during the specified time period file bardir/foo was created.

−d *pathname*

This instructs **snhistory** to query for activity history directly from the metadata archive that exists in the directory specified by pathname rather than sending a request to an activated FSM to query the metadata archive. This option overrides the -F option, so the user should use either -F or -d but not both. This option can not be used against an active fiesystem.

-h This option causes **snhistory** to display the command-line usage message.

-H This option causes **snhistory** to display a more detailed description of the command-line usage message.

−f *filename*

This instructs **snhistory** to return the activity history for the file with the specified name that exists in the directory given by the -p option. If the -p option is not used, then snhistory looks for the given file in the file system root directory.

−m *event_mask*

This option allows the user to specify the file system activity of interest and is valid only when **snhistory** is executed in *event-history* mode. This option takes a character string as an argument. The character string is made from one or more of the following characters:

a - Attribute changes (i.e. linkcount, chmod, chgrp, chown)
c - Creates, renames, hard links
r - Removes
m - Modifications
f - Files
d - Directories

For example, to request the history for all creates, removes, and modifications for files only, use "-m crmf". To request remove activity for files and directories, use "-m r".

−n *max_results*

This option is used to limit the number of events returned from **snhistory** when executed in event-history mode or the number of files returned when executed in changed-file mode. The argument max_results is specified in units of one thousand. That is, "-n 20" limits the output to 20,000 events (in event-history mode) or files (in changed-file mode) in the output. By default, **snhistory** returns 50,000 events or files requested within the specified time period. This option can be used to limit the output to a manageable size. If **snhistory** hits the specified limit before processing all events, it will return a "next" transaction ID (TID) and an "end" TID in the snhistory_summary record that can be used to continue the history query where the previous call left off. Setting max_results to large values can cause snhistory to fail when the FSM exceeds its response time limit or to consume excessive amounts of memory when the -d option is used.

−o *output*

This specifies the name of the output file. If this option is not used, **snhistory** writes its output to stdout.

−q *filename*

Write qustat counters to the named file on completion.

−p *pathname*

snhistory returns events for all files and directories starting at the StorNext root directory by default. Use this option to limit the query to a particular directory and all of its sub-directories. For example, "-p /foodir" limits the query to directory foodir whose parent is the StorNext file system root directory (i.e. /stornext/snfs1/foodir). The leading "/" character is required. It can also be specified using the full StorNext path, "-p /stornext/snfs1/foodir". If the full StorNext path is specified then the -F option is unnecessary.

−s *date:time*

Use this option to specify the starting date and time for the history query. The date:time argument has the format yyyy-mm-dd:hh:mm:ss. For example, "-s 2016-05-12:10:00:00".

−e *date:time*

Use this option to specify the ending date and time for the history query. The date:time argument has the format yyyy-mm-dd:hh:mm:ss. For example, "-e 2016-05-12:10:59:59".

−t *sTID[:eTID]*

Use this option to specify the starting and ending transaction ID for the history query. This option cannot be used if either the -s or -e options were used to specify the start and end times. This option also has a special case where if only the colon character is specified (i.e. snhistory -F snfs1 -t:), then the last transaction ID that currently exists in the metadata archive will be returned:

```
root => snhistory -F snfs1 -t:
# LAST_TID: 34568
root =>
```

−w *waittime*

Use this option to specify the number of minutes that snhistory should wait for a response from the FSM before timing out. This value can also be specified in *snfs_rest_config.json* for all instances of this command as part of the process entry named *snhistory*. This file is installed by default and has a value of 5 minutes.

JSON OUTPUT FORMAT

To demonstrate the format of the JSON output, four directories and one file were created in a new file system. Running **snhistory** in *changed-file* mode produces the following output:

```
root => snhistory -F snfs1 -L
{
   "snhistory_summary": {
      "command": "snhistory -F snfs1 -L",
      "time": 1464041388177080,
      "num_events": 1
      "next_tid": 136
   },
   "event_list": [
      {
         "type": "CREATE",
         "time": 1464032746853778,
         "pathname": "/a/b/d/e/newfile",
         "size": 29,
         "uid": 0,
         "gid": 0
      }
   ]
}
root =>
```

The first record is always a snhistory_summary record. At a minimum it contains the command-line that produced the given JSON output, the time when the command was executed, the total number of events returned, and the tid to use as the starting tid for the subsequent snhistory command to retrieve the next set of consecutive events. In this example, one file was created named "newfile".

The following is an example of using the -n option to limit the number of events returned to the caller to ten thousand:

```
root => snhistory -F snfs1 -s 2016-05-12:10:00:00 \
                    -e 2016-05-12:10:59:59 -n 10
{
   "snhistory_summary": {
      "command": "snhistory -F snfs1 -s 2016-05-12:10:00:00 -e
2016-05-12:10:59:59 -n 10",
      "time": 1464041388177080,
      "num_events": 10006
      "next_tid": 7723321
      "end_tid": 9332588
   },
   "event_list": [
   . . .
```

```
    ]
}
root =>
```

A little more than ten thousand events were returned because *snhistory* queries data at the level of file system transactions and many events can take place within a single transaction. Since the event limit was hit, the snhistory_summary record contains the fields for next_tid and end_tid. To continue with the query, the next call to *snhistory* would look like:

```
root => snhistory -F snfs1 -t 7723321:9332588 -n 10
```

The field end_tid will NOT be included in the snhistory_summary record when the value for next_tid has exceeded end_tid, signifying that all events within the given time period have been returned to the user.

EXAMPLE snhistory COMMANDS

To direct snhistory to search for events or changed files in a metadata archive that is associated with an active file system, the file system must be identified using either the -F or -p options. For example, the following two commands are equivalent:

```
root => snhistory -F snfs1 -p /foodir/bardir
root => snhistory -p /stornext/snfs1/foodir/bardir
```

To direct snhistory to search a metadata archive that is not associated with an active file system, use the -d option rather than the -F or -p options:

```
root => snhistory -d /tmp/snfs1-mdarchive -p /foodir/bardir
```

To query for the metadata history for file app001.dat in directory /foodir/bardir of file system snfs2:

```
root => snhistory -p /stornext/snfs2/foodir/bardir -f app001.dat
```

Here is an example of using snhistory in *event-history* mode. Query for all events within the given two hour time period using the text output format:

```
root => snhistory -F rhSource -T -s 2017-02-13:14:30:00 \
                        -e 2017-02-13:16:30:00
[02-15-2017 12:36:55][snhistory -F rhSource -T -s 2017-02-13:14:30:00 -e 2017-02-13:16:30:00]
num_events:9 next_tid:193
[02-13-2017 14:42:30][159] REMOVE /dir1/file12
[02-13-2017 14:42:30][159] REMOVE /dir1
[02-13-2017 14:42:30][159] REMOVE /dir1/file11
[02-13-2017 15:35:26][162] CREATE /foo
[02-13-2017 15:35:28][164] MODIFY /foo 4
[02-13-2017 15:44:44][170] CREATE /bar
[02-13-2017 15:44:46][172] MODIFY /bar 4
[02-13-2017 15:46:12][179] CREATE /soothe.txt
[02-13-2017 15:46:14][181] MODIFY /soothe.txt 418
root =>
```

The same example but using snhistory in *changed-file* mode. Query for all files that have changed during the same 2 hour time period above using the compact-text output format:

```
root => snhistory -F rhSource -LCT -s 2017-02-13:14:30:00 \
                          -e 2017-02-13:16:30:00
```

```
/dir1/file12
/dir1/file11
/foo
/bar
/soothe.txt
# NEXT_TID: 193
root =>
```

To query for all files created or renamed in the file system within the given one hour period using the compact JSON output format:

```
root => snhistory -F snfs2 -C -m fc -s 2017-02-13:14:00:00 \
                                -e 2017-02-13:14:59:59
```

EXIT STATUS

snhistory will return zero on success and non-zero on failure.

FILES

/usr/cvfs/data/FsName/FsName-mdarchive
/usr/cvfs/config/FsName.cfgx

SEE ALSO

snfs_rest_config.json(4)

NAME

WebService access to Storage Manager file operations

SYNOPSIS

snretrieve [-h] [-v] [-a] file [file ...]

snstore [-h] [-v] [-a] file [file ...]

snrmdiskcopy [-h] [-v] [-a] file [file ...]

snfileinfo [-h] [-a] file [file ...]

snjobinfo [-h] -m MOUNT job [job ...]

DESCRIPTION

Commands to allow remote client access to Storage Manager file management commands. For managed files allows control of store, retrieve and truncate operations. For file systems where remote web service access has been configured, these commands provide similar functionality to the storage manager commands fsretrieve, fsrmdiskcopy, fsstore and fsfileinfo. Commands run synchronously by default, the command waits for actions to complete and optionally reports on the results using the **-v** option. The **-a** option can be used to make the command asynchronous. In this case one or more job numbers are reported and snjobinfo can be used to later wait for and report the results.

The commands can take files or directories as arguments. If a directory is specified, then its contents are processed recursively. If a mix of files and directories are specified then more than one job will be used to run the commands.

If a file is specified with @ in front of its name, then this file is interpreted as a list of the files to be processed one entry per line. This allows a preselected set of content to be input to the command.

The snjobinfo command requires that a pathname in a filesystem be provided, this is so that the location of the web server can be determined. The snjobinfo command can be used repeatedly to look at the results of past commands.

These commands require python 2.7.9 or later version.

EXIT VALUES

Commands will exit with zero on successful runs and non-zero otherwise.

NAME

snlatency − Measure StorNext File System network latency

SYNOPSIS

snlatency *pathname* [*repeat*]

DESCRIPTION

The **snlatency** program allows measuring the round trip latency from the StorNext client to the FSM process on the MDC host. The file system to measure is selected via the *pathname* parameter.

snlatency uses a special RPC to the MDC in a loop to measure the time to send the message, the time for the FSM process to service the request, and the time to send the response back up to the system call level. The average of *repeat* calls is averaged and the latencies printed out in micro seconds.

If the MDC is not local, then the NTP clock algorithm is used to adjust for the difference between the two host's clocks before calculating the results, it is not used when the MDC the same host. The reason for this is that the NTP algorithm assumes the latency in both directions is the same and in effect averages them out. Comparing the latencies seen between the client local to the MDC and remote clients will show how much of the latency is caused by network infrastructure.

EXIT VALUES

snlatency will return 0 on success and non-zero on failure.

NAME

snlicense − Report StorNext license status

SYNOPSIS

snlicense [**-v**|**-m**|**-n**|**-h**|**-p**] [**-f** *license-file***]** [**-i** *file***]** [**-r** json|ascii**]** [**-s** *system-id***]** *license_type* [**nocap**]

DESCRIPTION

snlicense imports new licenses and reports on the current license status.

Reports include information such as license status, capacities and expirations. The currently supported license types can be displayed by typing **snlicense -h**.

The optional **nocap** argument is for internal use only to prevent a hang when certain components are not fully initialized.

This information should be submitted to Quantum Technical Support when contacting them about license issues.

OPTIONS

-f *license-file*
: Will use an alternative license file. The default license file is */usr/cvfs/config/license.dat*.

-h Display help

-i *file* Import licenses from the specified file

-m Will cause the output to be printed in a machine readable format

-n No RAS messages for license warnings and errors

-p Shows whether a product key is needed and if so, whether it exists and is valid

-P *cvfsid*
: Validates the product key for both MDCs of an HA pair

-s *system-id*
: Check the license entry for the specified system-identifier

-r json|ascii
: Report on license service usage.

-v Will cause more license info to be reported

USAGE

Use this command with the -i option to import licenses from the command line or without the -i option to check the current status of a particular license. To check the status, execute the program with the desired license type. See examples below for more info.

IMPORTING LICENSES

The **-i file_to_import** option can be used to import licenses from the downloaded license file into the local license file.

NOTE: The licenses are only imported to the local computer. For High Availability (HA) systems, the license file must be imported on both computers.

LICENSE EXPIRATION

Some licenses are cross-checked for expiration such that if one license expires, all cross-checked licenses are considered expired.

The **-v** option can be used to determine the expiration date of the feature and the expiration date of the feature with cross-checking.

- The "Actual expiration" is when the feature itself expires.
- "License expiration" is when the feature will become unusable due to one or more cross-checked licenses expiring.

Note: In some cases it is invalid to mix expiring licenses with non-expiring licenses. One notable exception is the **maintenance** license (which typically has an expiration).

The **-v** option will display a warning if there is mixed-expiration error on the specified license.

LICENSE SERVICE REPORT

The -r option reports on the license service usage. The license service is only used with 'san_client' and 'file_system' licenses.

EXAMPLES FOR CHECKING LICENSE STATUS

When the command is run, the license info for the indicated StorNext feature is reported. Note that if license status cannot be determined for a feature it is treated as NOT licensed. Also note some features have an associated capacity and some do not.

Examples:

Properly licensed with capacity available:

```
% snlicense proxy
- The proxy license status is: Good

% snlicense -v proxy
- Found existing license: proxy
- License expiration:     None
- Actual expiration:      None
- Licensed capacity:      65535 connections
- Current used capacity:  NA
- The proxy license status is: Good
```

Properly licensed (non-capacity license):

```
% snlicense maintenance
- The maintenance license status is: Good

% snlicense -v maintenance
- Found existing license: maintenance
- License expiration:     None
- Actual expiration:      None
- The maintenance license status is: Good
```

Unlicensed feature:

```
% snlicense failover
- The failover license status is: License Missing

% snlicense -v failover
- The failover license status is: License Missing
```

A valid temporary license:

```
% snlicense -v failover
```

- Found existing license: failover
- License expiration: Thu Nov 1 23:59:59 2012
- Actual expiration: Thu Nov 1 23:59:59 2012
- The failover license status is: Good

Expired license:

% snlicense failover
- The failover license status is: Expired

% snlicense -v failover
- Found existing license: failover
- License expiration: Thu Jan 1 23:59:59 2009
- Actual expiration: Thu Jan 1 23:59:59 2009
- The failover license status is: Expired

Mixed permanent and expiring licenses:

% snlicense server
- Having both permanent and temporary licenses is not allowed
(except for 'maintenance' and DLAN 'proxy' licenses).

% snlicense -v server
- Found existing license: server
- License expiration: Jan 1 23:59:59 2009
- Actual expiration: None
- Licensed capacity: 10 connections
- Current used capacity: NA
- Having both permanent and temporary licenses is not allowed
(except for 'maintenance' and DLAN 'proxy' licenses).

% snlicense failover
- Having both permanent and temporary licenses is not allowed
(except for 'maintenance' and DLAN 'proxy' licenses).

% snlicense -v failover
- Found existing license: failover
- License expiration: Jan 1 23:59:59 2009
- Actual expiration: Jan 1 23:59:59 2009
- Having both permanent and temporary licenses is not allowed
(except for 'maintenance' and DLAN 'proxy' licenses).

EXIT VALUES

snlicense will return one of the following upon exit.

```
0 - No error, feature licensed and usable
1 - Feature not licensed but not in use
2 - Current feature capacity undetermined but feature not in use
3 - Feature has no license string in the license file
4 - Feature has a license but the expiration date has been reached
5 - Feature has a license but the capacity has been reached
6 - Feature has a license but the current capacity could not be determined
7 - Feature license status could not be determined
8 - Invalid mixture of temporary and permanent licenses detected
```

LIMITATIONS

Only the **Linux** platform is supported with snlicense commands.

For Windows license information, follow the instructions in "License Dialog (cvntlicense.exe)".

SEE ALSO

- **www.quantum.com Support, Documentation Center**
- **StorNext Licensing Guide**
- **StorNext File System Release Notes**

NAME

snmetadb − Backup StorNext filesystem content

SYNOPSIS

Linux only.

[**-B** *fsname*]

[**-f** *fsname*]

[**-U** *fsname*]

[**-P** *fsname*]

[**-D** *fsname*]

[**-R** *fsname*]

[**-H** *fsname*]

DESCRIPTION

snmetadb provides capabilities for protecting the contents of a file system which has the *metadata archive* enabled. File system metadata and data can be stored to, and retrieved from an object store using S3 or to another file system.

snmetadb copies the contents of a file system to another, potentially remote location, over the S3 protocol. Or, copies it to another local file system. The copy process can include the file system data, or it can rely on Storage Manager to protect the data. The metadata copy made by **snmetadb** can be mounted as a read only fuse file system, or, used loaded on to a host and used to recover the file system after a disaster. Since the contents can include file system data, this recovery process can be used on non-managed StorNext file systems.

OPTIONS

The main command line options control the mode of operation for the command. The behavior of these modes is also influenced by a number of filesystem configuration options.

-B Make a full or incremental copy of the named filesystem metadata, and potentially its data, depending on the current state of the backup copy.

-f Make a full copy of the named file system metadata, and potentially data.

-U Update an existing copy of the file system metadata, and potentially data, with changes since the last time the command was run.

-P Prune old contents of the metadata copy and data so that only the current live content is saved.

-D Delete the copy of the file system from object storage.

-R Restore a remote copy of the file system metadata in preparation for doing a file system restore using cvmkfs -R or cvmkfs -D.

-H Run as a helper process of the file system fsm process during data restore.

FILE SYSTEM CONFIGURATION REQUIREMENTS

snmetadb will only operate on file systems which have the *drJournal* and *drCopy* features enabled. In addition, a set of other configuration options control the behavior of the process.

drCopyProtocol

defines the protocol used to communicate with the target location. The allowed protocols are
lattus for a Lattus object store using S3.
aws for Amazon Web Services using S3.
snmetadb for files on a locally accessible file system.

drCopyTarget

defines the URI of the location to store the content to. For S3 based storage, this has the format here depends on is this AWS storage, or an appliance such as Lattus servicing the S3 protocol.

http://hostname/bucket

https://hostname/bucket

Note that it is recommended to use a host name here and not an IP address. This configuration information ends up being part of the metadata archive contents and is expected to resolve to the service location independent of where the access is being made from.

drCopyOptions

controls how the data is represented in the copy of the file system.
compress will attempt to compress all data into large container objects using the LZ4 algorithm. This option should not be used when the majority of the data is incompressible.
adaptivecompress will attempt to compress data, but after a few samples will stop attempting to compress file types which have not generated good compression ratios.
storagemanager indicates that **snmetadb** should not copy file content, instead the Storage Manager component of StorNext is expected to be storing data to the object store.

OBJECT STORE ACCESS CONFIGURATION

snmetadb uses the s3cmd cli for some of its internal operations. This command needs configuration to setup access to the object store being used. Rather than use the default configuration file location, the internal use cases expect a config file at /usr/cvfs/config/.s3cfg This can be setup using an interactive process by running

s3cmd --configure --config=/usr/cvfs/config/.s3cfg

In addition, some data transfers are run more directly and require a separate configuration file to be setup at

/usr/cvfs/config/snmetadb_access.json

This is a json file containing a record per bucket used, each record contains the name of the bucket or namespace, the access key and the access secret. A template file is installed and can be modified.

METADATA COPY OPERATION

Running **snmetadb** with the -f or -U options accesses the live metadata archive managed by the fsm process and creates another copy on local storage in /usr/cvfs/backups. This copy will generally be smaller than the live metadata archive. Once this copy is complete, it is moved to the S3 location. In the case of an update operation the content moved will reflect only the changed content since the last operation and not the full metadata state of the file system.

When the target location actually is a file system and not an object store, the contents are placed there directly. If Storage Manager is responsible for moving file data, then the references it places into the file system metadata are used to access the content when using the fuse mount, there is no use of the Storage Manager database to access the content. In configurations where **snmetadb** itself is copying data, content is assembled into container files, uploaded to S3, and then removed locally.

If content is being stored to another file system instead of an object store then containers are still used, but they all remain in place. This configuration is intended for use with a relation point in a managed file system where the content could be archived to tape and truncated from the primary filesystem.

Note snmetadb itself does not truncate file content, nor will it backup the contents of previously truncated files if instructed to copy data itself, but running on managed content.

OBJECT STORE FORMAT

If an S3 bucket is specified as the target for content, then all content is stored at

s3://bucket/mdarchive/fsname/UUID/

Where UUID is associated with the original file system metadata archive and uniquely identifies it. The data containers are stored as objects at this prefix.

INVENTORY LOCATION

An inventory of the copy location of the metadata is stored as a *JSON* file in /usr/cvfs/backups/inventory. This file is all that is needed to retrieve the contents, or mount the fuse file system version. When storing to S3, the *JSON* is also stored to

s3://bucket/inventory/fsname.json

RESTORE PROCEDURE

When the file system copy contains file data, it can be used to perform a DR procedure on the file system on any StorNext host which has access to the content and sufficient disk space.

First the json file must be obtained from the object store, s3cmd can be used for this:

s3cmd get s3://bucket/inventory/fsname.json /usr/cvfs/backups/inventory/fsname.json

Then metadata archive itself can be retrieved using

snmetadb -R fsname

This will only function if the filesystem is not running locally. This first fetches the file system configuration file from the metadata archive. It will move any existing metadata archive out of the way and place the metadata archive and the configuration file in the correct places on the local system.

Once all LUNs are setup for the file system, it can be made with

cvmkfs -D fsname

Now the file system fsm can be started, and the file system mounted locally. Note that just activating the fsm is not the correct procedure here, the restore process places data in the filesystem and requires client access. The file system is accessible shortly after the fsm starts work on it, metadata is restored first, and once it has all been restored, data will be retrieved into the file system. This is performed by a copy of snmetadb started as a helper by the fsm. Should a user access a file while this process is ongoing, it will trigger an immediate retrieve of the requested file. The file system can be modified during the restore process, and updates to the backup copy can continue to be made after this.

LOG FILE

snmetadb logs activity to the file /usr/cvfs/debug/snmetadb.log

EXIT STATUS

snmetadb will return 0 on success and 1 on failure.

SEE ALSO

mount_snmetadb(8)

NAME

snprobe − Collect StorNext cluster state information

SYNOPSIS

Linux only.

snprobe [**-abcCdlLqrsv**] [**-D** *label*] [**-h** *host*] [**-m** *pattern*] [**-n** *nameserver*] [**-o** *output*] [**-p** *altpmap*] [**-t** *timeout*]

DESCRIPTION

Snprobe will use the StorNext administrative API to locate filesystems, hosts and storage associated with a StorNext cluster and output this as a json formatted description of the discovered resources. The output is designed for use by other programs, and not for an administrator to read directly.

OPTIONS

The command line options are used to control the level of detail provided.

-a Include all details.

-b Enable debug mode.

-c Requests that filesystem client hosts are queried. The output will include which clients are connected to which filesystems and some details about the software revision running on the client.

-C Option indicates that the json output should use a compact form which minimizes its size. This is for machine readable copy and is not recommended for human use.

-d Requests that the StorNext disks visible to each host are reported and information about their capacity and SCSI query information.

-D *label*

This option tests the label search and match capability of the snprobe api. If the specified label is found on any host in the cluster, information about the disk is printed to the output file. This information is in json format, and the normal json output is suppressed. Specifying this option automatically sets the client and disk options. The **-D** option may be specified multiple times to search for multiple labels in one call.

-h *host* Defines which host is queried for information first. This host must be running StorNext services. It will be queried for the list of filesystems it sees from the name service and where these filesystems are located. Each filesystem is then queried in turn for information about its size, usage, layout, and if requested, quota and client information. Finally each host identified is queried for information about the software level it is running, the platform it is running on and the licenses installed. If requested, the list of disks visible from the host is also reported. The **-h** option may be specified multiple times.

-l Requests that license information be reported for each visible host.

-L Do links. This option is not yet implemented.

-m *pattern*

Use this regular expression to identify disks that are to be assembled into the same class. This option may repeated. The disk inquiry string is used to match disks that fall into the same class. For many disk arrays, a filter is not needed. However, in some cases, such as when iscsi disks are used, the disk inquiry string is set by the administrator and may contain different characters for each disk. Without using a regular expression, each disk would be its own class.

-n *nameserver*

Indicates that the host is a nameserver for the cluster. Code is present in StorNext and snprobe to return the list of nameservers in the cluster and set this value automatically. To support hosts in clusters which have not yet been upgraded with this functionality, nameservers can be identified on the command line. The **-n** option may be specified multiple times for the case that multiple nameservers are present.

-o *output*
Will cause output to be sent to the specified file, stdout is the default.

-p *altpmap*
Tells snprobe to use this port number to contact the alternate portmapper. See the fsports man page for more information on changing the default alternate portmapper port for a cluster.

-q Requests that disk quota usage is included for filesystems which have quotas enabled.

-r Bypass the IP address to hostname translation. This can be useful when there are environmental issues which slow down this translation.

-s List the services running on the host as part of the filesystem. Storage Manager information is not reported as it is not externally visible. For each service the last time it was started is reported, its current status, how many times it has run, and if it has core dumped, how many times this has happened. If no host is specified, the localhost is queried.

-t *timeout*
Use this value as the timeout for making connections to remote hosts. The units are milliseconds.

-v Print snprobe version information and exit. No json output.

EXIT STATUS

Snprobe will return 0 on success and 1 on failure to connect to any services.

SEE ALSO

StorNext File System Release Notes

NAME

snqbmd − StorNext QBM Client Daemon

SYNOPSIS

snqbmd

DESCRIPTION

The **snqbmd** is the server daemon that collects I/O bandwidth statistics and forwards the information to the MDC. The daemon monitors stornext mount points. When a mount point is determined to have QoS bandwidth management active on the MDC, this daemon will send updates to the fsm via the <mountpt>/.__snfsqos directory.

snqbmd logs to */usr/cvfs/debug/snqbmd.log* for debugging purposes.

The file */usr/cvfs/debug/snqbmd.verbose* can be used to increase logging for snqbmd. Place a numerical value between 0 and 4 in the file to change the debug verbosity. The default value is 1. The snqbmd daemon must be restarted to make the change effective. This can be done using the **cvadmin** restartd command.

OPTIONS

This process runs in the background and is started at boot time as part of the stornext startup process.

FILES

*/usr/cvfs/config/*_qbm.conf*
/usr/cvfs/debug/snqbmd.log
/usr/cvfs/debug/snqbmd.verbose

SEE ALSO

snfs_config(5)

NAME

snquota − StorNext Quota Configuration Utility

SYNOPSIS

snquota {**-F** *file_system_name*|**-P** *path*} *action* [*options*]

DESCRIPTION

The **snquota** command manipulates the quota system in the *StorNext file system*.

The quota system provides a means for limiting the amount of disk storage consumed on a per user or per group basis across an entire file system or within a designated directory hierarchy. Quota limits apply to the space consumed by disk-block allocations for a user or group, which is not equal to the sum of their file sizes. Disk-block allocations can be less than the file size if the file is sparse, or more if the file system has allocated extra sequential blocks for the efficiency of anticipated future writes.

There are three types of quotas: user quotas, group quotas, and directory quotas. User and group quotas limit the number of file system blocks that can be allocated by the user or group on which the limit is placed. When quotas are on, the total allocated file system space of all users and groups that own files in the file system are automatically kept.

Directory quotas are a little different. The system does not automatically keep track of the usage for each directory. The **snquota** command allows directories to be turned into the root of a **Directory Quota Name Space (DQNS)**. Then, the number and size of all files in the directory and all its subdirectories are tracked and (optionally) limited.

For all quota types, limits and usage values only apply to regular files, not directories, symlinks, or special device files

Each quota entity has two limits associated with it. These are the hard limit and the soft limit.

The hard limit is the absolute limit which file system space usage should not exceed. Any time the total allocated space is at or over the hard limit, all further allocations or write requests by the offending user or group will be denied.

The soft limit is a lesser limit. When the user exceeds this limit (but not the hard limit), allocations are still permitted for a while. On UNIX platforms, a warning message will be written to the TTY of the user. When the soft limit has been overrun for longer than the grace period, the soft limit becomes a hard limit and any further allocations or write requests are denied. When the usage again falls below the soft limit, allocation requests will again be serviced.

Previous versions of **snquota** required all three of the limit values (hard, soft, and grace) to be set or all three to be unset. This version relaxes that somewhat. It is now possible to specify just a hard limit without a soft and grace limit. It's also possible to set a soft limit and grace period without a hard limit. It's also possible to specify a soft limit without any grace period (such a limit warns periodically, but never prevents allocation). It's still not possible to specify a grace period without a soft limit. Although all three values (**-h**, **-s**, **-t**) still need to be on the command line, you can set the limits that should not be enforced to zero.

For performance reasons related to the distributed nature of StorNext, quota overruns are not only possible but likely. The overrun size depends upon a number of factors including the size of the allocation request(s) at the time of the quota overrun.

Some versions of **snquota** support setting default limits for the various types of quotas. For example, you can set a default hard limit, soft limit, and grace period for all users. After doing that, any new user created will inherit those limits. Note that setting default limits will not change the limits of existing quota entities. Use the **-0** option and either "**-u default**", "**-g default**", "**-d default**", or "**-fd default**". See the examples section.

When working with Directory Quotas, the specified file system must be mounted on the node running **snquota**.

Limits are not enforced against super-user accounts.

DIRECTORY QUOTA NAME SPACES

DQNSs are created by either of two actions, **-C** or **-M**. They have different performance trade-offs and which one to use depends on the situation at hand.

The **-C** action creates a DQNS whose usage values (the amount of disk space and the number of files) are already initialized to the correct value. In order to initialize them, **snquota** must walk the directory tree under the root of the DQNS and tally up how much disk space is used. For big directory trees, this process can take a long time. Any modifications to the files and directories in the DQNS will be stalled until this walk is complete.

The **-M** action quickly creates a DQNS whose usage values are zero. As files are created in the DQNS, the usage value will increase, but will never count the files that were present in the directory when it was created. In order to initialize the DQNS so the values are correct, the quota database must be rebuild using the **-R** action. A rebuild runs much faster than a file-tree walk on a per-inode basis, but it must look at **all** of the inodes in the file system. When the rebuild is running, modifications to the file system will be stalled until the rebuild is complete.

So, when creating a DQNS that is believed to contain only a small percentage of the inodes in the file system, use **-C**. When creating a DQNS (or many DQNSs) that use a large percentage of the files, use **-M**.

A typical situation where **-M** would be useful is converting an existing file system to use directory quotas. First every directory which needs to be a DQNS root is marked with a call to "snquota **-M**". Then, all of the DQNSs are initialized with one call to "snquota **-R**".

When in doubt, use **-C**.

Nesting of DQNSs is not allowed. This means that a DQNS may not be a subdirectory of another DQNS.

Directories can not be renamed across DQNS boundaries. Also, all hard links to an inode must be within the same DQNS. Attempts to rename directories/files or create hard links that would violate this rule will result in a EXDEV being returned.

If a directory tree contains inodes with hard links outside of the tree, an attempt to convert the tree into a DQNS via the **-C** action on the tree will result in an error. An attempt to convert the tree into a DQNS via the **-M** and **-R** actions will result in an error during the **-R** action.

QUOTAS IN MIXED OS ENVIRONMENTS

The user and groups names specified in **-u** and **-g** represent underlying identifiers that are determined by the OS type of the MDC.

On a Linux MDC, those identifiers are the classic UNIX User IDentifier (UID) and Group IDentifier (GID). When a UNIX client (Linux, MacOS, Solaris, etc) creates a file, it passes the user's UID and GID to the MDC. Those IDs are attached to the file and are used by the quota subsystem. When a Windows client creates a file, it passes a UID and GID it gets from one of three places:

1. If the Active Directory entry for the user has UNIX IDs associated with it, those are used. The behavior at this point is just like UNIX client. The administrator can set the IDs for a user via the AD configuration tool under the "UNIX Attributes" tab. This tab is part of the "Identity Management for UNIX" subsystem.
2. If the user doesn't have UNIX attributes, then the user and group "nobody" IDs from the file system configuration file are used.
3. If the process's SID is a special "Root SID", the UID/GID passed will be 0/0 (i.e. root). The "root SID" is S-1-5-18.

The Windows client can associate a NTSD with a file, but it's ignored by the quota subsystem. (It's only used for access control by the client at that point.)

On a Windows MDC, the favored identifiers are user and group SIDs derived from the NTSD which owns the file. If there is no NTSD associated with the file, the UID/GID values associated with the inode are used. So, when a Windows client creates a file, it passes in a NTSD. That NTSD broken into SIDs and used as file's owner identifiers as far as the quota subsystem is concerned. When a UNIX client creates a

file, it passes the usual UID/GID pair, not a NTSD. This is used by the quota system. If that file is accessed from a on Windows client, it gets assigned an NTSD. At that point the quota will be wrong. Subsequent allocations of that file will be charged to the SD and not the UID/GID.

So, the preferred method of running quotas with a mixture of UNIX and Windows clients is to run with a Linux MDC with UNIX user/group mappings for Active Directory users. That way, a user who logs into clients of either OS will have a single quota (which will be based on the UID).

Another option is to just use Directory Quotas. They are much more straight-forward to share between OS types.

UNITS

Usage and Limits are printed in a human-readable form, suffixed with "K", "M", "G", "T","P", or "E" for kibibytes, mebibyte, gibibyte, tebibyte, pebibytes, or exbibytes, respectively. These are base-2 values (i.e. 1K = 1024). A value without a suffix is in bytes.

File count values are also printed with these suffixes, but they are base-10 values (i.e. 1K = 1000).

Time values are printed with the suffixes "m", "h", "d", "w", "M" and "y" for minutes, hours, days, weeks, months, and years, respectively.

If the **-e** option is used, the suffixes are disabled and exact values are printed. Time units are in minutes.

These suffixes can also be used when specifying limits with the **-h**, **-s**, and **-t** options. Decimal values may be used (e.g -h 1.5g).

For the most part, the case of the suffix doesn't matter. The only exception to that is the time suffixes "m" and "M", meaning "minutes" and "months", respectively.

FILE SYSTEM SPECIFICATION

-F *FileSystemName*

Specify FileSystemName as the file system to manipulate. FileSystemName may be qualified with cluster and administrative domain, if other than the default cluster in which the command is being run. The syntax is FileSystemName@<cluster>[/<addom>].

-P *Path* Specify the file system containing *Path* as the file system to manipulate.

ACTIONS

-C This action creates an initialized DQNS on the directory specified by the **-d** argument. After this command is run, disk space usage and file counts will be tracked in the directory and all its subdirectories. Later, limits may be set on the DQNS using the **-S** action. Note that since this operation creates *and initializes* the DQNS, the directory tree contained by the new DQNS will be walked to total up the current usage values. This may take some time. Modifications to the files and directories in the DQNS will be stalled until this walk is complete.

-D This action destroys the DQNS specified by the **-d** argument. Disk space and file count usage values will no longer be tracked. Limits will no longer be enforced. Note that this does not modify or destroy the files and directories in the DQNS in any way.

-G This action returns the quota limits and values for the user, group, or directory specified by the **-u**, **-g**, or **-d** option, respectively.

-L This action lists the current quota limits and values for all user, group, and directory quotas.

-M This action creates (marks) an uninitialized DQNS on the directory specified by the **-d** argument. After this command is run, disk space usage and file counts will be tracked in the directory and all its subdirectories. Later, limits may be set on the DQNS using the **-S** option. Note that since this operation creates *(but does not initialize)* the DQNS, the usage values for the DQNS will start out at zero. The user should later use the **-R** action to initialize the usage values. See the **DIRECTORY QUOTA NAME SPACES** section above for a discussion on when to use **-M** and when to use **-C**. When in doubt, don't use this action. Use **-C** instead.

-R This action rebuilds the quota database. It is most useful when used after **snquota** has been used a number of times with the **-M** action. See the **DIRECTORY QUOTA NAME SPACES** section

above. Note that this action can take a long time. The file system will be unresponsive during this time. The action cannot be canceled after it is started. A prompt will be displayed confirming the intent to run the action unless the **-Y** option is specified. A rebuild preserves limits and DQNSs.

-S This action sets the quota limits for the user, group, or directory specified by the **-u**, **-g**, or **-d** option, respectively. The limits must be specified by the **-h**, **-s**, and **-t**, options. All three must be present, but some/all may be zero.

-X This action generates quota reports for all users and groups. There are three files placed in */usr/cvfs/data/*<file_system_name>:

1. quota_report.txt - a "pretty" text file report.
2. quota_report.csv - a comma delimited report suitable for Excel spreadsheets. Fields that can have commas, newlines, and other strange characters are escaped with double quotes. This is the case with DQNSs path fields.
3. quota_regen.in - a list of snquota commands that can be used to set up an identical quota database on another *StorNext file system*. Passing this file to the shell executes it as a script.

-Z This action completely destroys all quota configuration and then rebuilds the quota database as in the **-R** option above. Unlike the **-R** action, **-Z** clears the limits and directory quota name spaces. It is recommended that the quota_regen.in file be generated using the **-X** option to save the quota configuration prior to using **-Z**.

OPTIONS

-0 List, read, and set default limits. When specified with the **-L** action, the default limits are included in the list (if they are non-zero). When specified with the **-G** or **-S** actions, the default limits for the specified type of quota are read or written. When using **-0**, the argument for **-u**, **-g**, **-d**, or **-fd** should be "default".

-a When this option is used, directory quota paths printed by the **-L** and **-G** options will be absolute paths. Paths supplied to the **-d** option are also absolute paths (or relative to the CWD). When this option is absent, all paths are relative to the root of the specified file system.

-d *Directory*

This option specifies a DQNS on a *StorNext file system* to be used with the **-C**, **-D**, **-G**, **-M**, or **-S** options. The directory supplied is the root directory of the DQNS. The directory path is relative to the root of the specified file system, unless the **-a** option is used.

-e When used with the **-G** or **-L** actions, numbers will be printed as *exact* values. Usage and Limits which represent disk space are printed in bytes. Times are printed in minutes. For example, with this option, a one megabyte hard limit will be printed as "1048576", not "1M". A one day grace period will be printed as "1440", not "1d".

-f The **-f** option is only useful with the **-G** and **-S** actions and the **-d** option. When the **-f** option is present, limits and values represent the number of regular files contained in the DQNS. If the **-f** option is not present, limits and values represent the disk space contained in the DQNS.

-g *GroupName*

This option specifies the name of a group to get or set with the **-G** or **-S** action. The group name may also be of the form "G:id", where "id" is a number that represents a group's GID.

-H *HostName*

Use a hostname in a StorNext cluster that is different from the cluster the command is being run on. This option in rarely needed.

-h *HardLimit*

This option specifies a hard limit to set when used with the **-S** action. See the **UNITS** section above. The maximum hard limit value is 16 exbibytes - 1 fsBlockSize (the file system configured block size.) If the fsBlockSize is 4096, the maximum hard limit value is

18,446,744,073,709,547,520 bytes. (Note: commas were added for readability; commas are not allowed in values.)

-h This option causes **snquota** to print a friendly help message and exit. It only works when used by itself. If there are other options present, it is assumed that a hard limit is being specified.

-o {**text**|**xml**|**json**}
Print output in **text**, **xml**, or **json**. The default is **text**.

-s *SoftLimit*
This option specifies a soft limit to set when used with the **-S** action. See the **UNITS** section above. The maximum soft limit value is 16 exbibytes - 1 fsBlockSize (the file system configured block size.) If the fsBlockSize is 4096, the maximum soft limit value is 18,446,744,073,709,547,520 bytes. (Note: commas were added for readability; commas are not allowed in values.)

-t *GracePeriod*
This option specifies a grace period to set when used with the **-S** action. See the **UNITS** section above.

-u *UserName*
This option specifies the name of a user to get or set with the **-G** or **-S** action. The user name may also be of the form "U:id", where "id" is a number that represents a user's UID.

-Y When used with the **-R** or **-Z** actions, this option prevents **snquota** from asking for confirmation.

-z This option is the same as specifying "-h 0 -s 0 -t 0". It's only useful with the **-S** action.

EXIT VALUES

snquota will return 0 on success and non-zero on failure.

EXAMPLES

List all the quota limits and values on a file system named "data".

```
snquota -F data -L
```

List all the quota limits and values on a file system named "data", but in cluster "cluster1".

```
snquota -F data@cluster1 -L
```

Specify a hard limit of ten gigabytes, a soft limit of nine gigabytes, and a grace period of one week on user "lisa" in a file system named "data".

```
snquota -F data -S -u lisa -h 10g -s 9g -t 1w
```

Turn off quota limits for user "lisa" in a file system named "data".

```
snquota -F data -S -u lisa -z
```

Get the quota values for a group named "simpsons" on a file system mounted on "/stornext/data".

```
snquota -P /stornext/data -G -g simpsons
```

Create a **DQNS** on the directory "/lisa/saxophone_music" in a file system mounted on "/stornext/data".

```
snquota -P /stornext/data -C -d /lisa/saxophone_music
```

Specify a hard limit of one gigabyte, a soft limit of nine hundred megabytes, and a grace period of one day on pre-existing DQNS "/lisa/saxophone_music" in a file system named "data".

```
snquota -F data -S -d /lisa/saxophone_music -h 1g -s 900m -t 1d
```

Create a number of DQNSs using **-M** and **-R**. This is faster than using **-C** if these directories take up most of the space in the file system.

```
snquota -F data -M -d /bart/comics
snquota -F data -M -d /bart/pranks
snquota -F data -M -d /bart/itchy_and_scratchy
snquota -F data -R
```

Create the same DQNSs using **-C**. This is faster than using **-M** and **-R** if the directories are small.

```
snquota -F data -C -d /bart/comics
snquota -F data -C -d /bart/pranks
snquota -F data -C -d /bart/itchy_and_scratchy
```

Specify a hard limit of one thousand files, a soft limit of nine hundred files, and a grace period of one week on pre-existing DQNS "/bart/pranks" in a file system named "data".

```
snquota -F data -S -d /bart/pranks -f -h 1k -s 900 -t 1w
```

Set default hard limits limits for all newly created DQNSs. The first line set defaults for disk space and the second line sets defaults for files.

```
snquota -F data -S -0d default -h 1t -s 0 -t 0
snquota -F data -S -0fd default -h 1m -s 0 -t 0
```

SEE ALSO

snfs_config(5), **snfs_rest_config.json**(4)

NAME

snrecover − recover deleted files on a StorNext File System

SYNOPSIS

snrecover [**-F** *FsName*] -p *pathname* {**-t** *datetime* | **-T** *tid*} [*options*]

DESCRIPTION

The **snrecover** program uses a file system's currently active *metadata archive* to recover recently deleted files and directories. It requires that the file system's FSM be running and activated and that the configuration parameter metadataArchive was set to "true" and metadataArchiveDays was set to a non-zero value at the time the files were deleted. This program was developed for non-managed file systems, but can be used for managed file systems as well if the files or directories being recovered did not reside within a Storage Manager relation point. **snrecover** can be used to recover a single specific file or a directory with all of its sub-directories and files. Recovered directories and files are placed in the lost+found directory located in the root directory of the StorNext File System.

The **snrecover** program sends a "start recover" request to the FSM and immediately returns to the user. The file system FSM then performs the recovery operation independent of snrecover. As the FSM progresses in the recovery process, it writes progress update messages in the cvlog and system log files. When it is finished recovering the specified metadata, it will write a "Recovery complete" message in the log files.

For each file in the recovery attempt, **snrecover** will recover all user file extents for data in disk blocks that have not been reallocated for use by other files. During file recovery, if it is detected that a disk block was re-allocated to another file, then the recovery immediately terminates for that file and the recovery procedure moves onto the next file. In this way, a file may be only partially recovered, starting with the beginning of the file and ending just prior to the first reallocated disk block. Since StorNext can be aggressive in its attempts to re-use disk blocks, the more time that passes between when a file was deleted and when the recovery was attempted, the less likely a full recovery of all user data will be successful.

OPTIONS

−F *FsName*

This is the name of the StorNext file system that contained the deleted files.

-h This option causes **snrecover** to display the command-line usage message.

−p *pathname*

This is the full or relative pathname for the previously deleted file or directory to recover. The pathname must begin with a leading '/' character. If the path is relative to the root of the file system, the file system name must also be provided. If a full path is given, the file system name will be determined from the path. The parent of the directory or file to be recovered must exist at recovery time.

Note that **snrecover** can only recover files and directories that have been previously deleted. It cannot be used to find and recover deleted content within an existing directory. To find deleted files to recover within an existing directory, use **snhistory** to get the list of files that were deleted from that directory and then use that list as input to successive calls to **snrecover**.

−t *[date:]time*

Use this option to specify the date and time just prior to when the files were deleted. The date:time argument has the format yyyy-mm-dd:hh:mm:ss. For example, "-t 2016-05-12:09:15:00". If a date is not specified, then today is assumed.

−T *tid* Use this option to specify a transaction ID that will mark a reference point in time for the recovery. This option is for internal use only.

NOTES

The **snrecover** program does not perform the recovery itself. Rather, it sends a "recover" request to the activated FSM and it is the FSM that performs the recovery using the metadata restore infrastructure and the currently active metadata archive to perform a partial, historical metadata restore into directory lost+found.

For managed file systems, the **snrecover** recovery process does not coordinate with Storage Manager at all. As such, the **fsrecover** utility should be used to recover content that is protected under a Storage Manager

relation point rather than **snrecover**.

EXAMPLE

Given that today's date is 4/8/2016, either of the following two commands can be used to attempt to recover the contents of directory dir3 as it existed earlier in the day at 12:30pm on 4/8/2016 in the root directory of file system snfs1.

=> snrecover -p /stornext/snfs1/dir3 -t 2016-04-08:12:30:00
=> snrecover -F snfs1 -p /dir3 -t 12:30:00

EXAMPLE

Some files were earlier deleted from an existing directory but the names of those files have been forgotten. snhistory is used to find the files that were deleted from directory /stornext/snfs1/p1 and then that list is used as input for successive calls to snrecover.

=> snhistory -TCmr -p /stornext/snfs1/p1 -s 2017-12-05:17:00:00 -e 2017-12-05:17:15:00
REMOVE file12
REMOVE file11
NEXT_TID: 11410148

=> snrecover -p /stornext/snfs1/p1/file12 -t 2017-12-05:17:00:00
=> snrecover -p /stornext/snfs1/p1/file11 -t 2017-12-05:17:00:00

EXIT STATUS

snrecover will return zero if validation of the input arguments succeed or non-zero otherwise. If the request for the FSM to start a recovery fails, the message describing the reason for the failure will be in the system log file and the file system's cvlog file.

FILES

*/usr/cvfs/data/**
/usr/cvfs/data/FsName/log/cvlog
/usr/cvfs/config/FsName.cfgx

SEE ALSO

snhistory(8)

NAME

snrest − Administer StorNext File System REST API access

SYNOPSIS

snrest gateway admin-url *url*

snrest gateway import *filename*

snrest gateway show

snrest gateway remove [*url*]

snrest gateway register [*url*]

snrest gateway unregister [*url*]

snrest gateway status [*−d*]

snrest gateway ping [*url*]

snrest gateway ping [*−f filename]*

snrest config set [*name=value...]*

snrest config clear [*name...]*

snrest config show

DESCRIPTION

The **snrest** program supports control and status display for REST API access to the StorNext File System.

The **snrest gateway** commands support configuration and control of StorNext interactions with a single or multiple Quantum API Gateways.

Commands with a <url> parameter may specify the *url* using only the hostname of the API Gateway. If the *protocol* is not specified it defaults to **https** and if the port is not specified it defaults to 8443. This allows the caller to only specify the *hostname*, as in the following example:

```
snrest gateway admin-url <hostname>
```

which is the same as explicitly specifying the following:

```
snrest gateway admin-url https://<hostname>:8443
```

The **snrest gateway import** command imports configuration information from a Quantum API Gateway. This command takes a single argument which is a filename containing contact and authentication information and must be obtained from a Quantum API Gateway.

Alternatively, the **snrest gateway admin-url** command is a simpler form for configuring access to the API Gateway. Use this command only if your site has not configured site-specific authentication for gateway registration (in which case you must use **snrest gateway import** instead).

With **snrest gateway admin-url** simply specify the URL of the API Gateway as a command line argument.

The **snrest gateway show** command displays the API Gateway configuration.

The **snrest gateway remove [<url>]** command unregisters with the API Gateway and removes the API Gateway from the configuration. If the *url* is specified, only the specified API Gateway is unregistered and removed.

The **snrest gateway register [<url>]** command registers all the configured API Gateways. API Gateways that are registered will continue to periodically retry until the registration succeeds. This needs to be called if you use **snrest gateway import** or **snrest gateway admin-url** commands to add an API Gateway configuration to the configuration.

An optional URL parameter may be specified to **snrest gateway register**. The following command:

```
snrest gateway register <url>
```

results in the same actions as if the following commands were issued:

```
snrest gateway admin-url <url>
snrest gateway register
```

This allows the caller to configure the *url* and register with the API Gateway in a single command. If the specified API Gateway has already been configured, the command performs only the registration.

The **snrest gateway unregister** command unregisters with the currently registered API Gateways. If the *url* is specified, only that API Gateway is unregistered.

The **snrest gateway status** command displays the current status of the StorNext registrations with a Quantum API Gateway. This typically reports that either all or none of the internal endpoints in StorNext are registered with a particular API Gateway.. But there is also the possibility, especially in transitory states or when connectivity to the API Gateway is problematic, that only a subset of the internal endpoints are registered. If any endpoints are not currently registered the system will continue to try to register those endpoints until successful.

There is an additional option to get extra detail on the registration status with the **-d** option. This provides a detailed display of individual internal endpoints that register with the API Gateway and thus is primarily intended for use by Quantum Support.

The **snrest gateway ping** command tests connectivity to the API Gateway. If the API Gateway cannot be contacted an error message is displayed and the command exits with a non-zero exit status. By default, the following will ping the currently configured API Gateway:

```
snrest gateway ping
```

Alternatively, you can ping a specific URL with:

```
snrest gateway ping <url>
```

or use a file with the gateway configuration with the **-f** option:

```
snrest gateway ping -f <filename>
```

In this latter case the file specified would be of the same form used with **snrest gateway import**.

The **snrest config** commands support setting values in the **/usr/cvfs/config/snfs_rest_config.json** file. These values should only be changed in consultation with Quantum support. The names are hierachical value separated by spaces. For example, to change the **ssl_enabled** value under the **ssl** json object to **true**:

```
snrest config set ssl.ssl_enabled=true
```

To remove the ssl_enabled value:

```
snrest config clear ssl.ssl_enabled
```

Changes to the **snrest config** values will take effect the next time that StorNext is restarted.

The **snrest config show** command displays the current values of the StorNext REST configuration.

EXIT VALUES

snrest will return 0 on success and non-zero on failure.

LIMITATIONS

Only the **Linux** platform is supported with snrest commands.

NAME

WebService access to Storage Manager file operations

SYNOPSIS

snretrieve [-h] [-v] [-a] file [file ...]

snstore [-h] [-v] [-a] file [file ...]

snrmdiskcopy [-h] [-v] [-a] file [file ...]

snfileinfo [-h] [-a] file [file ...]

snjobinfo [-h] -m MOUNT job [job ...]

DESCRIPTION

Commands to allow remote client access to Storage Manager file management commands. For managed files allows control of store, retrieve and truncate operations. For file systems where remote web service access has been configured, these commands provide similar functionality to the storage manager commands fsretrieve, fsrmdiskcopy, fsstore and fsfileinfo. Commands run synchronously by default, the command waits for actions to complete and optionally reports on the results using the **-v** option. The **-a** option can be used to make the command asynchronous. In this case one or more job numbers are reported and snjobinfo can be used to later wait for and report the results.

The commands can take files or directories as arguments. If a directory is specified, then its contents are processed recursively. If a mix of files and directories are specified then more than one job will be used to run the commands.

If a file is specified with @ in front of its name, then this file is interpreted as a list of the files to be processed one entry per line. This allows a preselected set of content to be input to the command.

The snjobinfo command requires that a pathname in a filesystem be provided, this is so that the location of the web server can be determined. The snjobinfo command can be used repeatedly to look at the results of past commands.

These commands require python 2.7.9 or later version.

EXIT VALUES

Commands will exit with zero on successful runs and non-zero otherwise.

NAME

WebService access to Storage Manager file operations

SYNOPSIS

snretrieve [-h] [-v] [-a] file [file ...]

snstore [-h] [-v] [-a] file [file ...]

snrmdiskcopy [-h] [-v] [-a] file [file ...]

snfileinfo [-h] [-a] file [file ...]

snjobinfo [-h] -m MOUNT job [job ...]

DESCRIPTION

Commands to allow remote client access to Storage Manager file management commands. For managed files allows control of store, retrieve and truncate operations. For file systems where remote web service access has been configured, these commands provide similar functionality to the storage manager commands fsretrieve, fsrmdiskcopy, fsstore and fsfileinfo. Commands run synchronously by default, the command waits for actions to complete and optionally reports on the results using the **-v** option. The **-a** option can be used to make the command asynchronous. In this case one or more job numbers are reported and snjobinfo can be used to later wait for and report the results.

The commands can take files or directories as arguments. If a directory is specified, then its contents are processed recursively. If a mix of files and directories are specified then more than one job will be used to run the commands.

If a file is specified with @ in front of its name, then this file is interpreted as a list of the files to be processed one entry per line. This allows a preselected set of content to be input to the command.

The snjobinfo command requires that a pathname in a filesystem be provided, this is so that the location of the web server can be determined. The snjobinfo command can be used repeatedly to look at the results of past commands.

These commands require python 2.7.9 or later version.

EXIT VALUES

Commands will exit with zero on successful runs and non-zero otherwise.

NAME

snseq − StorNext cvdb "perf" trace analyzer

SYNOPSIS

snseq {read,write,rdwr} {--persec,--details,--csv} [filer_options] *files(s)*

snseq devread *file(s)*

snseq devwrite *file(s)*

snseq -h, --help

DESCRIPTION

snseq is a tool for analyzing cvdb "perf" traces. It is useful for determining file access patterns of applications as well as measuring I/O latency and performance at both VFS and device levels.

Running **snseq** first requires capturing client traces using the **cvdb** command. **snseq** emits to "stdout" so capturing output to a file requires shell redirection.

By default, **snseq** displays a human-readable ASCII format. However, some **snseq** command lines allow the specification of *--csv* that will cause it to emit CSV format that can be used with spreadsheet applications. This is especially useful for producing graphs. Visualizing data in this manner can provide insight into how an application is performing I/O.

OPTIONS

Analysis mode:

read Process VFS reads.

write Process VFS writes.

rdwr Process VFS reads and writes.

devread
Process device reads.

devwrite
Process device writes.

filter_options:

--seq track largest sequential I/O stream

--start start time offset

--end end time offset

--proc filter by process name

--minsize
filter out files smaller than byte_count

help:

-h, --help
Just display help and exit

EXAMPLE

```
# mkdir /tmp/cvdb
# cd /tmp/cvdb
# cvdbset perf
# cvdb -g -C -F &
test is run here
# killall cvdb
```

```
# snseq read --details cvdbout* > seq.out
```

NOTES

snseq uses python. If for some reason the Python package available on the system is incompatible with **snseq**, installing an additional Python package may be required to run **snseq**. Alternatively, cvdb "perf" traces captured on one system can often be processed on another system. It is often possible to capture cvdb perf traces on a client and then copy them over to the MDC and run **snseq** there. This technique can also be used to process perf traces taken from a backrev client, subject to certain limitations. For example, older cvdb perf traces do not contain enough information to display file names.

On Windows systems, snseq is installed as "snseq.py" On other platforms, it is installed as "snseq"

SEE ALSO

cvdbset(8), **cvdb**(8)

NAME

snstatd − StorNext Statistics Daemon

SYNOPSIS

StorNext File System process

DESCRIPTION

snstatd is a daemon that is launched by the *StorNext File System* (*SNFS*) **fsmpm**(8) process. It is used to gather statistics about file system activity and related activity such as I/O and storage manager activity. This daemon also services requests from the qustat command and archives statistics to .csv files. The archive files reside in /usr/cvfs/qustats.

snstatd is configured using the **qustat.conf**(5) configuration file, which specifies the intervals for gathering and archiving statistics. Debugging levels can also be specified in this file.

snstatd may be started, stopped or restarted in **cvadmin**(8) using the **startd**, **stopd**, **restartd** commands respectively. This may be done to propagate changes made to the existing **qustat.conf**(5) file.

snstatd logs to */usr/cvfs/debug/snstatd.log* for debugging purposes.

ENVIRONMENT

snstatd is available on all StorNext installs.

FILES

/usr/cvfs/debug/snstatd.log
/usr/cvfs/config/qustat.conf

SEE ALSO

qustat.conf(5), **qustat**(8),

NAME

WebService access to Storage Manager file operations

SYNOPSIS

snretrieve [-h] [-v] [-a] file [file ...]

snstore [-h] [-v] [-a] file [file ...]

snrmdiskcopy [-h] [-v] [-a] file [file ...]

snfileinfo [-h] [-a] file [file ...]

snjobinfo [-h] -m MOUNT job [job ...]

DESCRIPTION

Commands to allow remote client access to Storage Manager file management commands. For managed files allows control of store, retrieve and truncate operations. For file systems where remote web service access has been configured, these commands provide similar functionality to the storage manager commands fsretrieve, fsrmdiskcopy, fsstore and fsfileinfo. Commands run synchronously by default, the command waits for actions to complete and optionally reports on the results using the **-v** option. The **-a** option can be used to make the command asynchronous. In this case one or more job numbers are reported and snjobinfo can be used to later wait for and report the results.

The commands can take files or directories as arguments. If a directory is specified, then its contents are processed recursively. If a mix of files and directories are specified then more than one job will be used to run the commands.

If a file is specified with @ in front of its name, then this file is interpreted as a list of the files to be processed one entry per line. This allows a preselected set of content to be input to the command.

The snjobinfo command requires that a pathname in a filesystem be provided, this is so that the location of the web server can be determined. The snjobinfo command can be used repeatedly to look at the results of past commands.

These commands require python 2.7.9 or later version.

EXIT VALUES

Commands will exit with zero on successful runs and non-zero otherwise.

NAME

sntier − user interface for StorNext primary tiering

SYNOPSIS

sntier

−h, −−help show this help message and exit
−−host host Tiering host to use, default localhost
−−user user User name for authentication, default is login name
−−passwd passwd Password for authentication, will prompt if needed
subcommand

DESCRIPTION

Sntier is a wrapper around all the different interfaces for configuring and using the StorNext Primary Tiering feature. Primary tiering is a service for controlling the initial placement of and movement of file system content between different tiers of storage.

There are several different subcommands:

jobs Monitor and control tiering job status.

template
Define templates for jobs and policies.

policy Create/monitor tier policies.

submit Submit tiering job. There are also shortcuts for most of these.

location
Report pool location of file content

pool Monitor pool usage and Manage pools

config View or set tier service configuration

configreset
Reset components of configuration to their defaults

service Enable or disable local tiering service

setup Set default user and host for commands

Standard options for all command modes except **service** are:

−−**host** hostname
Host running tiering service, defaults to localhost.

−−**user** user
Username for authentication, defaults to login user.

−−**passwd** password
Will be prompted for if not supplied.

AUTHENTICATION

With the exception of running root on the same host as the tiering service, all commands require authentication as a designated user or administrator. Once authenticated, credentials are stored for a reuse by subsequent commands until they expire.

Authenticated users are granted one of two roles, administrator or user. A user may submit and monitor jobs, control their own jobs and view the current fill level of storage pools. Setting up authentication and user roles is covered under CONFIGURATION

DEFINITIONS

The storage within a file system is grouped into named pools. Each pool is one or more stripe groups. By defining pools consisting of different types of physical storage it is possible to direct the movement of content to a pool with the needed level of service. Each

Files are moved either by **jobs** or **policies**. The major difference between the two is that a job runs once and

a policy is executed repeatedly on a schedule and may also be limited to only running under certain conditions. Work for a **policy** is executed by **jobs**. Because of this, jobs and policies support the same set of parameters.

A **template** is a partial definition of a job or policy which can be used as a predefined set of rules for a **job** or **policy**. A **template** allows an administrator to predefine a set of tasks which end users can apply to content. Templates do not allow specification of the content, that comes from the job or policy definition.

Jobs, policies and templates are persistent across machine or service restarts.

CONFIGURATION

Each host used to provide Tiering capabilities must be configured first. Any Linux client running a Redhat derived OS may be used, this includes the MDC hosts. Tiering can be very I/O intensive, this should be factored in to the selection.

Tiering is enabled via the **snfs_rest_config.json** file in the config directory, once enabled, all configuration may be performed via the **sntier** config command. With no options this command will report the current configuration. The **config** option requires authentication as an admin, or running as root on the same host as the service. Setting up authentication for users or remote administration must be done on the service host first. The source of authentication and group membership is the authentication service configured for login on the service host.

Configuration state is persistent.

sntier config

[−h] [−−users [username [username ...]]]
[−−groups [groupname [groupname ...]]]
[−−admins [username [username ...]]]
[−−admin_groups [groupname [groupname ...]]]
[−−expiry secs] [−−job_size size] [−−job_min size]
[−−job_files files] [−−workers [8−64]]
[−−iothreads [2−8]] [−−iosize I/O size]
[−−smtp_server address] [−−smtp_port port] [−−smtp_tls]
[−−smtp_user email account]
[−−smtp_passwd email password] [−−snsm_user user]
[−−snsm_passwd passwd] [−−snsm_batch count]
[−−history [1−30] days] [−−query_size results]
[−−email {text,json}] [−−logging {quiet,verbose}]
[−−json]

Optional arguments

−h, −−help
: show this help message and exit

Authorization settings

−−users [username [username ...]]
: accounts allowed access

−−groups [groupname [groupname ...]]
: groups allowed access

−−admins [username [username ...]]
: accounts allowed admin access

−−admin_groups [groupname [groupname ...]]
: groups allowed admin access

−−expiry secs
: expiry time for authentication sessions

Without setting up authorization, only the root user on the local host can configure the tiering service. This account must be used to define the sets of users and groups who can perform admin or end user tasks. It the

host used has authentication integrated with an external service such as Active Directory, tiering will use the same source for account data.

SMTP email settings:

−−smtp_server address
host address of mail service

−−smtp_port port
port number of mail service

−−smtp_tls
ese TLS for email communication

−−smtp_user email account
email account name to use

−−smtp_passwd email password
email account password

With no email configuration settings, the tiering service will look for an open relay to send email and if one is not found no email will be delivered. Internally the mail command is used to deliver messages. Many modern email servers require two factor authentication, in this case the password here will be an ´application password´, both microsoft exchange servers and google support this mechanism.

Job processing settings

−−job_size size
Maximum job split data size

−−job_min size
Minumum job split data size

−−job_files files
Maximum files in job split

−−workers [8−32]
Content scan threads

−−iothreads [2−8]
I/O worker threads per scan thread

−−iosize I/O size
I/O buffer size (4Mbytes default)

These parameters control how I/O is performed by tiering, some tuning may dramatically improve throughput. The data size fields can take a suffix of G, M, K or B to specify units of Gibytes, Mibytes, Kibytes or bytes. Note these are power of 2 units.

Changing the iothreads parameter, or decreasing worker thread count currently requires restarting sntierd using the cvadmin −e "restartd sntierd" command.

Storage Manager web service settings

−−snsm_user user
Web service user id to use in retrieve operations

−−snsm_passwd passwd
Web servicce password to use in retrieve operations

−−snsm_batch count
Number of files to submit to storage manager in one request

Settings required to trigger storage manager content retrieves. Storage Manager web services must be enabled first.

Tier configuration settings

−−history [1−30] days
Days of job history to keep

−−query_size results
Max search query results from file system

−−email {text,json}
email report format

−−logging {quiet,verbose}
logging level

Configuration reset

Using the configreset mode, different groups of configuration can be set to their default values.

−−email
reset email configuration

−−auth reset authorization configuration

−−snsm
reset storage manager configuration

−−all reset all configuration

JOB SUBMISSION

A Tiering **job** represents a set of work the tiering service is being asked to perform. A **job** is submitted to a specific host for execution. Multiple hosts can perform tiering operations on the same file system, however, there is no coordination between them. All tiering operations are performed via the file system client and controlled via REST services. A **job** description consists of upto five different sections, or which **content** and **action** are always required.

sntier submit [−h] [−−template template]
[−−paths paths | −−search search_path | −−pathfile file]
[−−action {move,promote,demote,remove,inventory,checksum,validate}]
[−−target pool] [−−checksum algorithm] [−−estimate]
[−−iosize I/O size] [−−pin] [−−sessions] [−−retrieve]
[−−smstore | −−no−smstore] [−−smtrunc | −−no−smtrunc]
[−−min size] [−−max size] [−−newer N] [−−older N]
[−−pool poolname] [−−match regex] [−−exclude regex]
[−−files | −−dirs | −−all] [−−email address]
[−−format {text,json}] [−−verbose] [−−runat time]
[−−wait | −−watch]

optional arguments

−h, −−help
show this help message and exit

−−template template
template job parameters

−−wait wait for job completion or suspension

−−watch
monitor submitted job full screen

A template refers to the name of a pre−defined template which contains a set of job parameters. Options on the command line can override ones in the template. For example, a template may define moving content to a specific **pool**, a job could specify the template and the set of content to process.

Content options

−−paths paths
job paths

−−search search_path
job search directory

−−pathfile file
file containing job paths

The **paths** option may be used more than once, a directory followed by a / is processed recursively, individual files may also be listed. Relative path names are converted to absolute ones relative to the current working directory. No checking is performed in the CLI however, and it is possible to submit a job referencing content which is not available on the local host.

The **search** option can only be used once. Relative path names are converted to absolute ones relative to the current working directory.

All content is expected to refer to a files in a single file system.

Task options

−−action {move,promote,demote,remove,inventory,checksum,validate}
job action

−−target pool
Target storage pool

−−checksum algorithm
checksum algorithm to use

−−estimate
estimate work first

−−iosize I/O size
buffer size for I/O

−−pin pin content to storage pool

−−sessions
use allocation sessions during movement

−−retrieve
retrieve storage manager truncated content

−−smstore
mark files as store candidates on managed systems

−−no−smstore
mark files as not to be stored by storage manager

−−smtrunc
mark files as truncate candidates for storage manager

−−no−smtrunc
mark files as not to be truncated by storage manager

Content **pinning** allows moving content to a storage **pool** in a manner which will not be moved back by a policy.

The checksum implementation uses openssl and the supported algorithms are based on the ones supported by openssl. md5 and sha256 are the main ones, but also sha, sha1, sha224, sha384, sha512, whirlpool. Also available is xxhash which is a much lighter weight user of CPU.

When a checksum is applied during a data movement operation there is a significant overhead in terms of cpu and an additional data read.

Selection criteria

−−min size
minimum file size to process

−−max size
maximum file size to process

−−newer N
process files changed or accessed in the last N seconds

−−older N
process files not changed or not accessed in the last N seconds

−−pool poolname
process only files in named pool

−−match regex
process files matching regex

−−exclude regex
skip files matching regex

−−files only process files

−−dirs only process directories

−−all process both files and directories

Regular expression matching is currently not supported with the search form of content specification. The min and max size options can take a suffix of G, M, K or B to specify units of Gibytes, Mibytes, Kibytes of bytes. Note these are power of 2 units.

Reporting options

−−email address
email completion report

−−format {text,json}
email format

−−verbose
more verbose reporting

Scheduling options

−−runat time
time to run job

Action

There are several different actions which can be performed on selected content.

move Migrate file content to the target **pool** if not already there, assign directories to the **pool** so that new files will be placed in the pool.

promote
Make a new copy of the content in the target **pool** and leave the old one intact for a subsequent **demote** operation.

demote If the file has not been changed since a promote operation and the old copy is on the correct storage, make it the primary copy and remove the current primary copy. The combination of **promote** and **demote** consumes more disk resources, but for read only uses cases offers faster reclaimation of space in a pool. For files where the content has changed, a move is performed to replace the old copy with the new one.

remove Remove tiering labelling from files and directories. File content is left in its current location, any alternate copy is removed from the file system.

inventory
Scan the contents and build a report of the space being used in the different pools.

checksum
: Scan contents and generate a checksum using the specified algorithm store the checksum in an extended attribute.

validate
: Scan the contents, generate a checksum and validate it against a previously stored checksum extended attribute.

Criteria
: Criteria allows filtering down of the content to files and directories which match a set of rules.

Report
: An email report can be mailed to an email address at job completion.

Schedule
: A job can be scheduled to run at a specific time in the future. This allows running work during off hours for example.

JOB SUBMISSION SHORTCUTS

There is a short form for submitting all job types instead of using submit −−action= you can use the name of the action followed by a set of strings. For move, promote or demote, the first parameter is the target pool name, followed by a set of paths. For inventory, only paths are specified and for checksum and validate, the first following argument is the checksum algorithm. Other critera and task arguments are still supported.

For example:

```
sntier move pool /pathname −−older 300 −−wait
```

Is the equivalent of

```
sntier submit −−action=move −−target=pool −−paths=/pathname −−older 300 −−wait
```

MONITORING AND CONTROLLING JOBS

sntier jobs [−h]
[−−list | −−show job_id | −−delete job_id |
−−suspend job_id | −−resume job_id |
−−wait job_id | −−watch job_id]
[−−format {text,json}]

optional arguments:

−−list list jobs

−−show job_id
: job details

−−delete job_id
: delete job

−−suspend job_id
: suspend job

−−resume job_id
: resume job

−−wait job_id
wait for job done

−−watch job_id
refreshing display of job state

−−format {text,json}
report format

The **jobs** subcommand without other arguments brings up a refreshing display of jobs for a host. This includes all of jobs which have completed, jobs which are currently running and jobs which are queued to run. The following keys can be used in the short and long form displays:

up/down arrow
Select a job.

space Display job details.

d Delete a completed or queued job, or cancel a running job. Must be the original submitted of a job, or an administrator.

D Delete all completed or queued jobs for the current user, or all if performed as an admin. Does not impact a running job.

s Suspend execution of a running Job.

r Resume a suspended job.

q Exit the display

POLICY AND TEMPLATE DEFINITIONS

Templates and policies have exactly the same parameters available as jobs with a couple of extra options. The criteria section allows specifying a **−−fill percent** in combination with **−−pool name**. This allows a policy to skip processing work if the named pool is less than X percent full. In addition, the schedule section may contain −−interval secs to allow for policies which repeat at a regular interval.

If a template is created which contains these, it will not be legal to use it as a job template.

Templates do not specify the content of a job, this must be provided by the job or policy.

Managing Templates

sntier template [−h] [−−add name | −−list | −−delete pol−id]
[−−action {move,promote,demote,remove,inventory,checksum,validate}]
[−−target pool] [−−checksum algorithm]
[−−iosize I/O size] [−−pin] [−−sessions] [−−retrieve]
[−−smstore | −−no−smstore] [−−smtrunc | −−no−smtrunc]
[−−min size] [−−max size] [−−newer N] [−−older N]
[−−pool poolname] [−−fill percent] [−−maxmove space]
[−−match regex] [−−exclude regex] [−−files | −−dirs | −−all]
[−−email address] [−−format {text,json}] [−−verbose]
[−−interval secs] [−−runat time]

optional arguments:

−h, −−help
show this help message and exit

−−add name
add new policy

−−list list policy

−−delete pol−id
delete policy

Task options:

−−action {move,promote,demote,remove,inventory,checksum,validate}
policy action

−−target pool
Target storage pool

−−checksum algorithm
checksum algorithm to use

−−iosize I/O size
buffer size for I/O

−−pin pin content to storage pool

−−sessions
use allocation sessions during movement

−−retrieve
retrieve truncated storage manager content

−−smstore
enable storage manager copy creation

−−no−smstore
disable storage manager copy creation

−−smtrunc
enable storage manager file truncation

−−no−smtrunc
disable storage manager file truncation

Selection criteria:

−−min size
minimum file size to process

−−max size
maximum file size to process

−−newer N
process files changed or accessed in the last N seconds

−−older N
process files not changed or not accessed in the last N seconds

−−pool poolname
process content in named pool

−−fill percent
minimum fill percentage for pool

−−maxmove space
maximum content bytes to move

−−match regex
process files matching regex

−−exclude regex
skip files matching regex

−−files only process files

−−dirs only process directories

−−all process both files and directories

Reporting options:

−−email address
email completion report

−−format {text,json}
email format

−−verbose
more verbose reporting

Scheduling options:

−−interval secs
policy run interval

−−runat time
time to run policy

The **sntier template** command is used to add, list and delete templates. Templates are added using the **−−add name** option followed by the template definition. When used without **−−add** the template command displays a list of templates, these can be selected and deleted or viewed in more detail.

Managing Policies

sntier policy [−h] [−−add name | −−list | −−delete pol−id]
[−−template name]
[−−paths paths | −−search search_path | −−pathfile file]
[−−action {move,promote,demote,remove,inventory,checksum,validate}]
[−−target pool] [−−checksum algorithm]
[−−iosize I/O size] [−−pin] [−−sessions] [−−retrieve]
[−−smstore | −−no−smstore] [−−smtrunc | −−no−smtrunc]
[−−min size] [−−max size] [−−newer N] [−−older N]
[−−pool poolname] [−−fill percent] [−−maxmove space]
[−−match regex] [−−exclude regex] [−−files | −−dirs | −−all]
[−−email address] [−−format {text,json}] [−−verbose]
[−−interval secs] [−−runat time]

optional arguments:

−h, −−help
show this help message and exit

−−add name
add new policy

−−list list policy

−−delete pol−id
delete policy

−−template name
template for policy contents

Content options:

−−paths paths
policy paths

−−search search_path
policy search directory

−−pathfile file
file containing policy paths

Task options:

−−action {move,promote,demote,remove,inventory,checksum,validate}
policy action

−−target pool
Target storage pool

−−checksum algorithm
checksum algorithm to use

−−iosize I/O size
buffer size for I/O

−−pin pin content to storage pool

−−sessions
use allocation sessions during movement

−−retrieve
retrieve truncated storage manager content

−−smstore
enable storage manager copy creation

−−no−smstore
disable storage manager copy creation

−−smtrunc
enable storage manager file truncation

−−no−smtrunc
disable storage manager file truncation

Selection criteria:

−−min size
minimum file size to process

−−max size
maximum file size to process

−−newer N
process files changed or accessed in the last N seconds

−−older N
process files not changed or not accessed in the last N seconds

−−pool poolname
process content in named pool

−−fill percent
minimum fill percentage for pool

−−maxmove space
maximum content bytes to move

−−match regex
process files matching regex

−−exclude regex
skip files matching regex

−−files only process files

−−dirs only process directories

−−all process both files and directories

Reporting options:

- −−email address: email completion report
- −−format {text,json} email format:
- −−verbose: more verbose reporting

Scheduling options:

−−interval secs
: policy run interval

−−runat time
: time to run policy

The **sntier policy** command is used to add, list and delete policies. Use with **−−add name** to define a new policy, use with no arguments to list and delete policies interactively. The **−−list** option will report on the set of active policies.

Enabling or Disabling Tiering

The **sntier service** command run as root can enable or disable the local tiering service. Once enable for the first time, it needs basic configuration.

−−enable
: enable tiering

−−disable
: disable tiering

MANAGING POOLS AND MONITORING SPACE

The **sntier pool** command without options will list all the file systems present on a tiering service host and display either the defined storage pools and their space usage, or the set of stripe groups available for placing in tiers. The used space display refreshes, and will add and remove file systems as they mount or umount.

An administrator can create a pool configuration or delete the pool configuration either interactively, or using command line options:

−−add add pools. Note, this command will reset an existing configuration to the configuration specified. Specify all pools in a single command when using this option.

−−list list pools

−−delete
: delete pools. Note, this command will delete all pools. To delete only some pools use −−add to specify the desired configuration and unspecified pools will be removed.

−−mount mountpnt
: mount point

Pool specifications for add:

- −−sg name [sg ...]: Define non−exclusive stripe group
- −−sgexcl name [sg ...] Define exclusive stripe group

EXAMINING POOL LOCATION OF CONTENT

The **sntier location** command can be passed a list of files and it will report on the pool location of the content. This is done by examining the affinity of the file.

ENABLING OR DISABLING TIERING

The **sntier service** command run as root can enable or disable the local tiering service. Once enable for the first time, it needs basic configuration.

−−enable
enable tiering

−−disable
disable tiering

RETURN CODE

The sntier command exits with 0 on success and 1 on failure.

NAME

snvalidatecctl − Perform validation on a StorNext File System central control file

SYNOPSIS

snvalidatecctl [**-F** *pathname*] [**-h**]

DESCRIPTION

The **snvalidatecctl** program is used to validate a central control file for the portmapper of a StorNext file system.

To create a central control file: pipe/capture the output to a file from running **/usr/cvfs/bin/nss_cctl_template**

OPTIONS

-F *pathname*

Provide the path to the central control file. If not provided it will default to **/usr/cvfs/config/nss_cctl.xml**. If default is not present, a prompt to enter the file will appear, and will default to **/usr/cvfs/config/** prefixed if only a filename is provided. This does not take into account the current working directory for the path - It must be the full path if it is not in **/usr/cvfs/config/**.

-h Help - Display usage/help menu

SEE ALSO

nss_cctl(4), **nss_cctl_template**(8)

NAME

WebService setup for Storage Manager file operations

SYNOPSIS

snwebsetup [-h] -u USER -p PASSWORD -s SERVER_URI dirs [dirs ...]

DESCRIPTION

snwebsetup creates configuration files in Managed StorNext file systems which can be used by the snretrieve, snstore, sntruncate, snfileinfo and snjobinfo commands as well as other services. Each listed directory has a file called .StorNext_rest.json created in it. This command should be run on the MDC as it needs access to the local mount point information.

For StorNext client access, the file is expected at the root of a file system. For use with NAS access, the file is expected at the exported root of the share.

Prior to running this command, web services should be enabled and an account created which has access to web services. The user name and password for this account are used as input to the **snwebsetup** command.

SEE ALSO

nss_cctl_template(8)

NAME

sn_dmap − Disk map utility

SYNOPSIS

sn_dmap [*options*] *devname*

DESCRIPTION

sn_dmap is a utility that can be used to manage disk volumes that are thin-provisioned. Typically these volumes remap logical block addresses (LBAs) and allocate space as needed. Space is allocated when a block is first written and can only be freed by issuing a SCSI unmap command.

sn_dmap operates on one device at a time specified by the **devname** parameter. The **devname** parameter can either be a full path to the device like /dev/mapper/mpathai or the StorNext volume label. By default **sn_dmap** will print summary mapping information about the volume. More detailed information may be displayed using the -v option. Detailed output is also available in JSON format.

OPTIONS

-? Display usage and exit.

-c Clear. Unmap all the blocks on the volume except the StorNext label. This option should be used only to clean up a volume before a cvmkfs operation. This will effectively clear all the blocks on the volume except the StorNext label and cannot be undone.

-C Clear. Unmap all the blocks on the volume including the StorNext label. This option should be used only to clean up a volume before a cvlabel and a cvmkfs operation. This will effectively clear all the blocks on the volume including the StorNext label and cannot be undone.

-d[**dddd**]
Run in debug mode. The more "d's" specified, the more debug information is printed.

-f Force the clear or unmap operation without an warning message.

-h Help. Display usage and exit.

-l *LBA* The starting logical block address. Optional when the **-v** option is specified and required for an unmap operation. See **-u**.

-n *nblks*
Specify the number of blocks. This option is required for an unmap operation. See **-u**.

-o *nsegments*
Specify the number of segments to display. This option is valid with the **-v** option and limits the number of segments displayed to the specified value.

-j Write detailed information to standard output in JSON format.

-P Write detailed information to standard output in compact JSON format. This minimizes the size of the output and is intended to be machine readable only.

-q Used with -t. Only return status 0 if the device is in the list of Quantum-branded disks supporting UNMAP/TRIM.

-t Report whether or not the device is thin provisioned and exit with status 0 if thin and status 1 if not thin.

-u Unmap the range specified by the -l and -n options.

-v Verbose. Display information about all the segments. The **-l** option can be specified to display segments starting with the specified **LBA**. The **-o** may be specified to limit the output to the specified number of segments.

-x Display segment information in hexadecimal. This option is available only with the **-v** option.

EXAMPLES

Display general mapping information about the given volume:

```
per1-# sn_dmap  /dev/mapper/mpathal
/dev/mapper/mpathal {

              segments        blocks
    mapped           4       2277376    1.09 GiBytes
  unmapped           6   15818022912    7.37 TiBytes
     total          10   15820300288    7.37 TiBytes

} snfs_meta_qx3_L23
```

Display verbose information about each segment of the given volume. Note that we used the StorNext volume name as the device name in this example:

```
per1-# sn_dmap -v snfs_meta_qx3_L23
/dev/mapper/mpathal {

  Segment              LBA       NBlocks      Status
        0                0          8192      mapped    4.00 MiBytes
        1             8192       7806976    unmapped    3.72 GiBytes
        2          7815168       2088960      mapped   1020.00 MiBytes
        3          9904128       5718016    unmapped    2.73 GiBytes
        4         15622144        172032      mapped   84.00 MiBytes
        5         15794176    4294959104    unmapped    2.00 TiBytes
        6       4310753280    4294959104    unmapped    2.00 TiBytes
        7       8605712384    4294959104    unmapped    2.00 TiBytes
        8      12900671488    2919620608    unmapped    1.36 TiBytes
        9      15820292096          8192      mapped    4.00 MiBytes

} snfs_meta_qx3_L23
```

Unmap a segment with the force option (no warning):

```
per1-# sn_dmap -f -u -l 7815168 -n 2088960 snfs_meta_qx3_L23
```

Verify that the segment is now unmapped:

```
per1-# sn_dmap -v snfs_meta_qx3_L23
/dev/mapper/mpathal {

  Segment              LBA       NBlocks      Status
        0                0          8192      mapped    4.00 MiBytes
        1             8192      15613952    unmapped    7.45 GiBytes
        2         15622144        172032      mapped   84.00 MiBytes
        3         15794176    4294959104    unmapped    2.00 TiBytes
        4       4310753280    4294959104    unmapped    2.00 TiBytes
        5       8605712384    4294959104    unmapped    2.00 TiBytes
        6      12900671488    2919620608    unmapped    1.36 TiBytes
        7      15820292096          8192      mapped    4.00 MiBytes

} snfs_meta_qx3_L23
```

Clear all mapped segments except those containing the StorNext label:

```
per1-# sn_dmap -c snfs_meta_qx3_L23
```

```
sn_dmap: *WARNING WARNING WARNING*

    You are about to unmap all the blocks on the device /dev/mapper/mpathal.
    This will destroy all data on the StorNext volume snfs_meta_qx3_L23
    except the StorNext label. This operation can not be undone.

    Do you want to proceed? (y / n) -> y
```

Verify all is unmapped except the StorNext label (first and last segments):

```
per1-# sn_dmap -v snfs_meta_qx3_L23
/dev/mapper/mpathal {

  Segment                LBA       NBlocks      Status
      0                    0          8192      mapped    4.00 MiBytes
      1                 8192    4294959104    unmapped    2.00 TiBytes
      2           4294967296    4294959104    unmapped    2.00 TiBytes
      3           8589926400    4294959104    unmapped    2.00 TiBytes
      4          12884885504    2935406592    unmapped    1.37 TiBytes
      5          15820292096          8192      mapped    4.00 MiBytes

} snfs_meta_qx3_L23
```

Generate detailed output in JSON format:

```
[root@dev-snc-daiquiri-n1 ~]# sn_dmap -j snfs_data_dev-sncqx24ss-1_1_L7
{
   "raw device": "/dev/mapper/mpathab",
   "mappings": [
      {
         "lba": 0,
         "nblocks": 163840,
         "status": "mapped"
      },
      {
         "lba": 163840,
         "nblocks": 1998848,
         "status": "unmapped"
      },
(additional mappings...)
      {
         "lba": 1953112064,
         "nblocks": 8192,
         "status": "mapped"
      }
   ],
   "mapped segments": 597,
   "mapped blocks": 8077312,
   "mapped bytes": 4135583744,
   "unmapped segments": 596,
   "unmapped blocks": 1945042944,
   "unmapped bytes": 995861987328,
   "total segments": 1193,
   "total blocks": 1953120256,
   "total bytes": 999997571072
}
```

NOTES

The unmap option (-u) is most likely useful only for development purposes.

sn_dmap execs the cvlabel command (/usr/cvfs/bin/cvlabel) to match the StorNext volume name to the device.

sn_dmap is currently supported only on Linux.

FILES

/usr/cvfs/bin/sn_dmap

SEE ALSO

cvlabel(8)

NAME

sn_scsi_resize − Utility to resize a virtual volume

SYNOPSIS

sn_scsi_resize [*options*] *[devname1 devname3 devname3...]*

DESCRIPTION

sn_scsi_resize is a utility that can be used to tell the disk driver infrastructure in the Linux operating system to get the size of a volume. The size of a volume may have increased, particularly in the case of a thin provisioned volume. The Linux device driver stack does not automatically detect size changes.

If no devnames are specified, **sn_scsi_resize** will rescan all multipath devices and their slaves.

Devnames may be specified as absolute paths like /sys/block/dm-3, /sys/block/dm*, /sys/block/sdg, or shorthand like mpatha, mpathb representing /dev/mapper files.

sn_scsi_resize will execute a StorNext disk refresh after the SCSI devices have been resized.

OPTIONS

-? Display usage and exit.

-h Help. Display usage and exit.

-l List. This option will list the device sizes. No size rescan is done.

-v Verbose. Display the sizes of the devices. Similar to the list option but works during resize.

EXAMPLES

Rescan all multipath devices and their slaves:

```
sn_scsi_resize
```

Rescan the multipath device dm-3 and its slaves:

```
sn_scsi_resize /sys/block/dm-3
```

Rescan the multipath device /dev/mapper/mpatha and /dev/mapper/mpathb and their slaves:

```
sn_scsi_resize mpatha mpathb
```

List the sizes of all device and their slaves:

```
sn_scsi_resize -l
```

NOTES

sn_scsi_resize is currently supported only on Linux.

Devices representing partitions cannot be resized. Error messages are suppressed unless the -v option is specified.

FILES

/usr/cvfs/bin/sn_scsi_resize

NAME

takeover_ip − broadcast virtual IP information for StorNext

SYNOPSIS

/usr/cvfs/lib/takeover_ip -d *device_name* **-m** *mac_address* **-i** *ipv4_ip*

DESCRIPTION

takeover_ip provides a mechanism for sending a gratuitous arp reply when the secondary node in an HA pair takes over as primary and activates the configured virtual IPs (VIPs).

This is typically used by the startup/shutdown scripts, not by an administrator. Only use when recommended by Quantum Support.

FLAGS

-h Display help

-d *device_name*
The device name of the network interface the VIP is on

-m *mac_address*
Mac address of network device the VIP is on

-i *ipv4_ip*
IPv4 address for VIP

LIMITATIONS

Only the **Linux** platform is supported with takeover_ip since it requires an HA pair.

FILES

/usr/cvfs/config/ha_vip.txt

SEE ALSO

cvfs(8), **vip_control**(8), **cnvt2ha.sh**(8)
SNFS Installation Instructions

NAME

vidio, vidio2 − Video frame producer consumer performance test

SYNOPSIS

vidio [*options*] *dir_path* [*dir_path*...]

vidio2 [*options*] *dir_path* [*dir_path*...]

DESCRIPTION

vidio can emulate a producer or a consumer of video frames. When run as a producer (write mode), **vidio** generates video frames and writes them to files that are created in the specified directory. When **vidio** is run as a consumer (read mode), it reads frames from the files in the specified directory that were previously created and written. By default, **vidio** runs in producer mode and creates one file for each frame.

vidio will run in one of two modes, constrained or unconstrained. The default mode is unconstrained and **vidio** will produce or consume frames at an unconstrained rate; as fast as the I/O will allow. If the **-F** option is specified, **vidio** will produce or consume frames based upon the specified frame rate. Dropped frames are noted in the output.

Optionally, more than one directory can be named. In this case, **vidio** will start an identical I/O stream in each specified directory.

vidio will then write performance information to the standard output. The verbosity of the performance data can be controlled using the -v option. A realtime updating curses based display is optionally available via the **-c** option.

vidio2 is an experimental version of **vidio** with the same syntax but containing an updated I/O engine. Its behavior is subject to change with future releases.

OPTIONS

-? Display usage.

-B Use system buffered IO per StorNext rules. The default is direct I/O.

-c Display important statistics via a curses based continuously updating display.

-d[**dd**] Run in debug mode. The more "d's" specified, the more debug information is printed.

-f *framesize*

Specify the *framesize* or the frame type. Various type of video frame types may be specified. The default frame type is "hdtv". Currently this results in a frame size of 8,294,400 bytes. Use the -? option to get a list of currently supported frame types. The *framesize* can also be specified in bytes. Optionally, the suffixes k, m, g, K, M or G can be added to the numeric frame size value to represent kilobytes, megabytes, gigabytes, kilobinary, megabinary or gigabinary values, respectively. The lowercase letters represent base 10 units (e.g. 1k = 1000) and the uppercase letters represent base 2 units (e.g. 1K = 1024.)

-F *framerate*

Emulate a frame producing or consuming device by limiting the number of frames produced or consumed per second to the specified frame rate. If the file file system cannot keep up to the specified frame rate, the "Dropped frames" stat is incremented.

-l *filelist*

vidio can process frames from a file containing a list of frames/files. In this manner, vidio can be used to process a directory of arbitrary DPX files. Currently this option only works in read mode. The filelist is simply a file containing a list of frames to be processed in order. The list can have empty lines and/or comment lines that must begin with #. All other lines are assumed to be file names in the directory dir_path.

-n *nframes*

The number of frames to read or write. The default is currently 60 frames.

-N *nframes*
The number of frames per file. The default is one.

-O *offset*
File offset in bytes at which to start frames, default 0. This allows very basic emulation of mfx files with non-aligned essence data.

-u Use non-aligned memory buffer, simulate bad application buffer alignment.

-R Read files in reverse order, simulate scrubbing backwards during edit. A starting offset near the end of the file must be used.

-p *prefix*
Frame file names use the given *prefix* instead of the default of "vidio". Vidio then appends '_NNNNNNN' as the frame number to the *prefix*.

-q *qdepth*
Do asynchronous I/O by queuing requests *qdepth* deep. If a frame rate is specified, the *qdepth* will effectively equate to the number of frames that are buffered.

-r Consumer mode. Read frames of previously created using the -w option.

-T *milliseconds*
When in constrained mode, stop the test with an error if the actual IO time is greater than the specified number of milliseconds.

-v[**vv**] Print performance output in a more verbose fashion. The more "v's" specified, the more performance information is printed.

-V Show the vidio version and exit.

-w Producer mode. Create files and write frames. Create and write is the default test mode.

FILES

/usr/cvfs/bin/vidio
/usr/cvfs/bin/vidio2

SEE ALSO

cvfs(8)

NAME

vidiomap − Video frame allocation inspector and resequencer

SYNOPSIS

vidiomap [*options*] *target_dir* [*target_dir*...]

DESCRIPTION

The **vidiomap** utility can be used to determine how the **StorNext** file system has allocated the files within a given directory. Optionally, **vidiomap** can be used to "defragment" and "resequence" those files. Optionally, multiple target directories may be specified.

The **vidiomap** utility is intended to be used primarily on video frame files within a directory. It is not indented for use as a general purpose file system allocation analysis or "defragmentation" utility. See **snfsdefrag**(1).

Without options, **vidiomap** will print a summary analysis of the allocation of files in the target directory, including the number of regular files, the total number of extents, and space consumed. Also printed is information relating to the total number and average size of the gaps between the extents.

The **-v** option may be used to provide detailed information about file allocation on a per extent basis. The actual file system block numbers consumed are printed along with the gaps between extents. Notice that gaps can be negative, indicating the subsequent allocation was to a lesser file system block number than the current allocation.

To analyze the allocation of files within a directory, the desired file order first must be determined. By default, **vidiomap** will sort all the files in the target directory alpha-numerically by file name. Optionally, file names may be filtered by file prefix and file suffix. See the **-p** and **-s** options. Optionally, a list of file names may be provided. See the **-f** option.

The "resequencing" and "defragmentation" of files is a multi-step process that makes heavy use of the **StorNext** file System Application Programming Interface, **SNAPI**. The steps are a follows:

1. Determine the file order.
2. For each file, a "shadow file" is created and blocks are preallocated.
3. The data is copied from the original file, to the shadow file.
4. The newly allocated extents are swapped into the original file inode.
5. The "shadow files" are removed.

Shadow files are named *filename*_**shadow** and are created in the target directory.

As stated previously, **vidiomap** is not a general purpose file system "defragmentor". An older file system or a file system nearing capacity may have a fragmented free space pool. Using **vidiomap** to resequence files may not help in this case and could make fragmentation worse. Consider using **snfsdefrag** before resequencing files with **vidiomap**.

The resequencing option is intended to work with the StorNext **Allocation Session Reservation** feature. This feature is managed using the GUI or by modifying the **allocSessionReservationSize** parameter, see **snfs_config**(5).

Because resequencing copies the data to the newly allocated space, consider the performance impact of resequencing files on a production system. Resequencing a large number of files can take some time, depending on the size of the files, the performance of the underlying storage, and other file system activity.

OPTIONS

-? Display usage.

-d[**dd**] Run in debug mode. The more "d's" specified, the more debug information is printed.

-f *file_list*
Get the list of target files from the specified file instead of the the target directory. The files will be processed in the order listed. The format for this file is one file name per line.

-p *prefix*
Target only files with the specified *prefix*. If the prefix option is specified along with the suffix option, both must be true to target a given file.

-r Resequence and "defragment" the target files.

-s *suffix* Target only files with the specified *suffix*. If the prefix option is specified along with the suffix option, both must be true to target a given file.

-v Be verbose. Print each extent of each target file showing the file system blocks consumed and gaps between the extents.

FILES

/usr/cvfs/bin/vidiomap

SEE ALSO

cvfs(8), **snfsdefrag**(1), **snfs_config**(5), **vidio**(1)

NAME

vip_control − manipulate virtual IP information for StorNext

SYNOPSIS

/usr/cvfs/bin/vip_control *option*

DESCRIPTION

vip_control provides a mechanism for editing, listing, activating and deactivating virtual IPs (VIPs) for the StorNext system. Virtual IPs may be helpful for applications needing access to the primary MDC when running in an HA configuration. Changes made to VIPs are not automatically copied over to an MDC's HA peer. As such, any changes that are made to the VIP configuration need to be done on both HA MDCs. **vip_control** does not update firewall rules. Changes to VIP configurations may require additional changes to the firewall rules of the system.

OPTIONS

-h Display help

-a Activate all configured VIPs

-d Deactivate all configured VIPs

-i Show VIP status along with physical NIC status

-l Show configured VIPs in a compact format

-u *vip_str*

Update the VIP configuration with the *vip_str* string. This replaces the contents of the VIP configuration file. The format of *vip_str* is as follows:

```
MAC address, IPV4 VIP, netmask, IPV6 VIP, prefix length
```

Each field in a VIP entry is separated by a comma, and each VIP entry can be separated by either a newline or a semicolon. Typically, the semicolon is used.

The following is an example of setting up two VIPs:

```
# vip_control -u "0030482D38F6,10.0.0.2,255.255.255.0,,;0030482d38f7,10.1.0.2,255
```

NOTES

When changing the *ha_vip.txt* on Quantum Appliance you also need to make sure that the *iptables* rules are updated to reflect the new information. You can use the following command to reset the VIP

```
# /opt/DXi/scripts/netcfg.sh --reset_snvip
```

FILES

/usr/cvfs/config/ha_vip.txt /opt/DXi/scripts/netcfg.sh

LIMITATIONS

Only the **Linux** platform is supported with vip_control since it requires an HA pair.

SEE ALSO

cvfs(8), **snfs_config**(5), **cnvt2ha.sh**(8), **iptables**(8)
SNFS Installation Instructions

NAME

altstoreadd − Enables the Alternate Store Location Remote Copy feature on files and adds them to the alternate store candidate list. It can also be used to verify that a remote copy has been made for a file or list of files. Note: The Alternate Store Location Remote Copy feature is a licensed feature.

SYNOPSIS

altstoreadd [**-c** [**-e**] [**-m**] [**-i**]] [**-h**] *[file [file ...]]*

DESCRIPTION

This command should only be executed under the direction of Quantum technical support or professional services.

The **altstoreadd**(1) command takes as input a list of files either as command line arguments or from reading standard input and enables the Alternate Store feature by setting the REMOTE_COPY_ENABLED flag in the extended attributes of the files and adds them to the alternate store candidate list. The specified files must be full path names, regular files, reside in a StorNext file system that has an Alternate Store Node configured, and be members of a policy class that has the Alternate Store feature enabled.

If the -c option is specified, each file in the specified list is only checked for whether a remote copy has been successfully made. No candidates will be added in this case. This can be useful to determine how many files in the list are still waiting for a remote copy to be made.

OPTIONS

-c Checks each file in the specified list to determine if a remote copy has been made and returns a total count of the number of files that have had a remote copy made, a total count of the files in the list that have remote copy enabled, and a total count of the files in the list that were either invalid or do not have remote copy enabled.

-e The **-e** option is only valid when **-c** is also specified. This option will cause the path to be written to stdout for every file in the list that has a remote copy.

-m The **-m** option is only valid when **-c** is also specified. This option will cause the path to be written to stdout for every file in the list that has remote copy enabled but is missing a remote copy.

-i The **-i** option is only valid when **-c** is also specified. This option will cause the path to be written to stdout for every file in the list that either does not have remote copy enabled or is invalid.

-h Show the usage and exit.

EXIT STATUS

Exit codes for the **altstoreadd**(1) command are:

0 Success

2 Database Connection Error

3 Logging Initialization Error

4 Environment Setup Error

5 Option/Argument Usage Error

6 Policy Configuration Error

9 License Validation Error

EXAMPLES

```
$ altstoreadd /stornext/snfs1/sdisk_policy/sdiskfile1 /stornext/snfs1/sdisk_polic
/stornext/snfs1/sdisk_policy/sdiskfile1: new alternate store candidate added
/stornext/snfs1/sdisk_policy/sdiskfile2: new alternate store candidate added

$ find /stornext/snfs1/lto_policy -type f | altstoreadd
/stornext/snfs1/lto_policy/ltofile2: new alternate store candidate added
/stornext/snfs1/lto_policy/ltofile6: new alternate store candidate added
```

```
/stornext/snfs1/lto_policy/ltofile9: new alternate store candidate added
/stornext/snfs1/lto_policy/ltofile1: new alternate store candidate added

$ find /stornext/snfs1/lto_policy -maxdepth 1 -type f | altstoreadd -c
21 of 121 remote copies exist (0 invalid or not enabled)
```

SEE ALSO

altstoremod(1), **fsaltnode**(1) **fsmodclass**(1)

NAME

altstoremod − Display and/or manipulate the Alternate Store Location Remote Copy feature alternate store candidate list. Note: The Alternate Store Location Remote Copy feature is a licensed feature.

SYNOPSIS

altstoremod [**-e**[-i]**]** [**-m**[-i]**]** [**-n** *num_entries*] [**-l**] [**-p**] [**-v**] *[mnt_pt...*]

altstoremod -d [**-s p**|**q**|**i**|**r**|**c**|**e**] [**-c a**|**b**|**o**] [**-l**] [**-p**] [**-v**] *[mnt_pt...*]

altstoremod -u p|**q**|**i**|**r**|**c**|**e -s p**|**q**|**i**|**r**|**c**|**e** [**-c a**|**b**|**o**] [**-l**] [**-p**] [**-v**] *[mnt_pt...*]

altstoremod -r

DESCRIPTION

This command should only be executed under the direction of Quantum technical support or professional services.

The **altstoremod**(1) command can be used to perform a variety of tasks for displaying and manipulating the alternate store candidate list as follows:

Show files with remote copies and/or missing remote copies.
Delete candidates from the alternate store candidate list.
Update the status for candidates.
Reset all candidates with active status to pending status.

If no arguments are specified, the **altstoremod**(1) command will display statistics for all candidates scheduled for remote copies.

OPTIONS

-c a|**b**|**o** Show candidates scheduled for remote copies.
Using **-c a** displays all candidates.
Using **-c b** displays batch candidates.
Using **-c o** displays on-demand candidates.
The default behavior is to display all candidates when the **-c** option is not specified.

-d Delete candidates. Not valid with the **-u** option.

-e Show files with existing remote copies.

-i Show inactive files rather than active files. This option is only valid with the -e or -m option. The default behavior is to display information on active files.

-l List additional candidate information.

-m Show files with missing remote copies.

-n *num_entries*
Limit output to the specified number of candidate entries.

-p List full file path in the output.

-r Reset all candidates with active status to pending status.

-s p|**q**|**i**|**r**|**c**|**e**
Select candidates based on current status.
Valid with the **-d** or **-u** options.
Using **-s p** selects candidates with a status of pending.
Using **-s q** selects candidates with a status of queued.
Using **-s i** selects candidates with a status of in progress.
Using **-s r** selects candidates with a status of retrieve wait.
Using **-s c** selects candidates with a status of completed.
Using **-s e** selects candidates with a status of error.

-u p|q|i|r|c|e

Update the selected candidates to the specified status.
Not valid with the **-d** option.
Using **-u p** updates candidates to a status of pending.
Using **-u q** updates candidates to a status of queued.
Using **-u i** updates candidates to a status of in progress.
Using **-u r** updates candidates to a status of retrieve wait.
Using **-u c** updates candidates to a status of completed.
Using **-u e** updates candidates to a status of error.

-v Verify that candidates exist in the StorNext file system. Note that this command issues the stat(2) system call and can have a performance impact on the file system.

mnt_pt...

The StorNext managed file system mount point(s) used for displaying and manipulating the alternate store candidate lists. The default is all StorNext managed file system mount points.

EXIT STATUS

Exit codes for the **altstoremod**(1) command are:

0	Success
2	Database Connection Error
3	Logging Initialization Error
4	Environment Setup Error
5	Option/Argument Usage Error
6	Policy Configuration Error
7	Filesystem Configuration Error
8	TSM Database Request Error
9	License Validation Error
10	Altstore Candidate Table Database Request Error

SEE ALSO

altstoreadd(1), **fsaltnode**(1)

NAME

altstore_gather − Compile debugging information for StorNext Altstore feature

SYNOPSIS

altstore_gather

DESCRIPTION

The **altstore_gather**(1) command is called by pse_snapshot to collect Alternate Store Location feature information for issue analysis by Quantum staff.

EXIT STATUS

0 Successful completion.

NAME

archive_cmp − Compare Archive, Media Manager, and Tertiary Manager configuration information

SYNOPSIS

archive_cmp [**-a** *archive_name*] [**-u**] [**-d**]

archive_cmp -i [**-u**]

archive_cmp -r

archive_cmp -h

The following options are for internal usage only, not for general use.

archive_cmp -dryrun [**-a** *archive_name*] [**-u**] [**-d**]

archive_cmp -g

archive_cmp -f *file_name*

DESCRIPTION

The function of **archive_cmp**(l) utility is to verify that Media Manager tape drive configuration matches the Tertiary Manager view and also verify those tape drives still exist within the actual archive. The primary purpose of the utility is to provide an ability to update Storage Manager's configuration with an archive after the archive has had tape drive maintenance performed such as a tape drive replacement.

The utility first compares Media Manager's tape drive configuration of an archive to the tape drive's that are actually contained within the archive. If there are differences found, the utility will resync Media Manager's configuration with the archive if the **-u** option is specified; otherwise, it will just indicate the discrepancies it has found.

Secondly the utility compares the Tertiary Manager's tape drive configuration with Media Manager's configuration. If a discrepancy is discovered, it will log that Tertiary Manager has a tape drive configured that no longer exists within Media Manager and if the **-u** option is specified the utility will update any tape drive device paths that are found to be incorrect. If a tape drive that Tertiary Manager knows about no longer exists in the archive, it is considered an orphan drive and is no longer available for use by Storage Manager. One can remove the orphan tape drive by deleting it or use this utility to specify a replacement tape drive from a list of free tape drive know to Media Manager by use of the **-r** option. The benefit of doing a replacement versus a delete/add of a old/free drive is that the new replaced tape drive will keep the same Storage Manager drive ID, which affects drive pools, and if the original tape drive was configured in DDM, it makes the necessary updates so the new tape drive replaces the original tape drive within DDM.

If no options are specified, the utility will only report current state information of Media Manager tape drives, Tertiary Manager tape drives, the tape drives within the archive itself.

Any tape drives configured within a vault are ignored by the utility. These drives are manually loaded drives and their configuration cannot be validated.

The output of the utility is captured in the */usr/adic/MSM/logs/scripts/archive_cmp.log* file. The output of each run is concatenated to the log file and the log files are rolled when necessary. This log file is captured by a pse_snapshot.

OPTIONS

-a *archive_name*
: Only run the comparisons and updates against the specified archive.

-u
: Make any configuration updates that can be performed to resync the tape drive configurations between Tertiary Manager, Media Manager, and the archive(s) that do not require a user's decision.

-d
: Use this option if the utility is not being executed from the command line (daemon mode). With this option, an admin alert is issued if any issues are detected and need to be addressed. Output is only captured within the log file. This option is used when the utility is added as a scheduled feature through **fsschedule**(1).

-i The utility will run in a user interactive mode. If a decision needs to be made as to what archive to check or if an update needs to be done, the utility will prompt the user to make a selection.

-r Makes any configuration updates to resync tape drive configurations between the archives, Media Manager, and Tertiary Manager (*-u* updates) and indicates if any orphaned tape drives exist. If an orphan tape drive exists, it prompts the user to select to either skip, delete, or replace the orphan drive with an available free drive.

-debug Enable additional debug output, primarly used for debugging the utility.

-h Display help.

-dryrun
Used internally when this utility is added as a scheduled feature through **fsschedule**(1). It makes a dryrun of the options to be used by the utility through the scheduler to make sure they are valid.

-g Used internally by the GUI to obtain a list of orphaned Tertiary Manager drives and a list of free drives that could be used to replace the orphaned drive.

-f *file_name*
Used internally by the GUI to specify the file to use by the utility for orphan drive deletion/replacement.

EXAMPLES

Example 1.

Run utility in display mode. This example shows a drive in an archive not known to Media Manager and suggestions on how to resolve the mismatch.

```
[root@xxx ~]#archive_cmp.pl

Checking MSM’s configuration against Archive:
  Archive #     Archive Name     Archive Type     Serial #
  1             sl500            ACSLS            559000100372

  Archive sl500 is connected to ACSLS server acsls-srvr and is configured to ACS

  Comparing MSM’s configuration against acsls archive

    Drive HU10552H0T: MSM configuration matches Archive (Slot: 0,0,1,1)

WARN: Archive drive 1210022999 with slot location of 0,0,1,2 not found in MSM
xdicomp

    Update not enabled, run vsarchiveconfig to sync MSM and archive
    or run archive_cmp.pl -u

Checking TSM’s configuration against MSM’s:
    Drive HU10552H0T: TSM configuration matches MSM (DRVID: 1)
```

Example 2.

Run the utility in update mode. This example shows a mismatch between Media Manager and a archive where a a new drive was found in the archive that does not exist in Media Manager and what was run to resync them.

```
[root@xxx ~]#archive_cmp.pl -u

Checking MSM’s configuration against Archive:
```

```
  Archive #     Archive Name     Archive Type     Serial #
  1             sl500            ACSLS            559000100372

  Archive sl500 is connected to ACSLS server acsls-srvr and is configured to ACS

  Comparing MSM’s configuration against acsls archive

    Drive HU10552H0T: MSM configuration matches Archive (Slot: 0,0,1,1)

WARN: Archive drive 1210022999 with slot location of 0,0,1,2 not found in MSM
xdicomp

  Running "/usr/adic/MSM/cli/bin/vsarchiveconfig -u acsls -n sl500" to get MSM
  in-sync with archive

Checking TSM’s configuration against MSM’s:
    Drive HU10552H0T: TSM configuration matches MSM (DRVID: 1)
```

Example 3.

Run the utility in replacement mode. This example shows an orphan Tertiary Manager drive and the replacement of it with a free drive from the same archive.

```
[root@xxx ~]#archive_cmp -r

Checking MSM’s configuration against Archive:
  Archive #     Archive Name     Archive Type     Serial #
  1             i500a            SCSI             ADICA0C0238B02_LLA

  Comparing MSM’s configuration against scsi archive

    Drive 1210025795: MSM configuration matches Archive (Slot: d257l0)
    Drive HU18464262: MSM configuration matches Archive (Slot: d256l0)
    Drive 1310023452: MSM configuration matches Archive (Slot: d258l0)

Checking TSM’s configuration against MSM’s:
WARN:  Drive 1210022999: Orphaned within TSM, not found in MSM
       Drive ID: 3
    Drive HU18464262: TSM configuration matches MSM (DRVID: 1)
    Drive 1310023452: TSM configuration matches MSM (DRVID: 2)

Processing the orphan TSM drives
  Processing Drive 1210022999 (DRVID: 3):
    Enter "s" To skip processing this drive
    Enter "d" To delete this drive from the TSM configuration
    Enter one of the following serial numbers to replace the orphaned drive
        1210025795   Archive name: i500a, Archive type: scsi

1210025795

    Replacing drive 1210022999 with drive 1210025795

  The processing of orphan drives succeeded
```

EXIT STATUS

0	Success completion.
1	There was an error processing the arguments.
255	An internal fatal error prematurely terminated **archive_cmp**(l).

SEE ALSO

fsschedule(1), **fsschedlock**(1)

NAME

bucket_report − Lattus Per-Bucket Report

SYNOPSIS

bucket_report *User Password IP_Address*

DESCRIPTION

The **bucket_report** command issues a REST API command to a Lattus main controller node for collecting per-bucket information. The results of the REST API are in JSON format, which is piped into a JSON parser to glean the desired information. Output is a Comma Separated Values (CSV) list suitable for use as input to a spreadsheet program. The columns of the CSV output are:

```
Bucket Name
Sum of sizes of files in the bucket (bytes written)
Number of files in the bucket
```

The reported information is generated once per day by a job running on the Lattus. Because of this, the information can be 24 hours old, depending on the timing of the Lattus job versus the running of the **bucket_report** command. The information can be updated by running the following qshell command per bucket:

```
# login to Lattus control node
# or run this as a remote command with ssh
/opt/qbase3/qshell -c "q.dss.manage.monitorNameSpace('<bucket name>', realtime=Tr
```

The *User* and *Password* parameters are the credentials of a user on the Lattus Control Node that has *LIST* permission on */manage*. Refer to section 8.4 of the Lattus REST API Users Guide for information about users. The *admin* user that is described in that section can be used as the credentials for the **bucket_report** command.

The following qshell commands can be used on the Lattus Control Node to create a new user with only *LIST* permission if that is preferred.

```
# login to Lattus control node
# run qshell
q.dss.manage.addUser('bucketrpt','xyzzy')
q.dss.manage.setPermissions('/manage','bucketrpt', ['LIST'])
quit()
```

Lattus permissions can be checked by logging into the main controller node of the Lattus system, and running the following command:

```
/opt/qbase3/bin/dss --permission-settings-get /manage
```

The *IP_Address* parameter is the address of the Lattus main controller node.

This user-editable script assumes that the port is the default value 8080. The script can be modified if a different port number has been selected in the Lattus configuration.

The **bucket_report** and **bucket_csv** executable programs are placed in the */usr/adic/TSM/util* folder for use by StorNext administrators.

FILES

bucket_report -- a user modifiable shell script.
bucket_csv -- a filter that takes JSON input and produces CSV output

NAME

build_file − Create a test data file.

SYNOPSIS

build_file

build_file *FILETYPE SIZE* [*filename*]

build_file help

DESCRIPTION

The **build_file**(1) utility is used to create a test data file of a specified type and size. Each file has an alphabetic header which is 80 bytes long and contains the *FILETYPE* and is of size *SIZE*. The minimum file size is 80 bytes and the maximum is 2 terabyte (2,199,023,255,552 bytes.)

OPTIONS

help Will show long instructions.

FILETYPE

Specify the type of file to generate. This can be one of the following:

binary - Create a data file with binary data. The data is written as an incrementing 32-bit integer.

alpha - Create a data file with alpha data. The data consists of the printable characters, (ASCII 32 through ASCII 126) repeated until the file size specified is reached.

sparse - Create a sparse data file. The data is sparsely written by creating gaps or holes which occupies no physical space. 1024 bytes of binary data are written at the end of each 5 megabytes of file size, however this is only done up to 100 times. The file size chosen will be rounded up to the next multiple of 5,000,000 bytes.

trans - Create a data file with bytes selected to stress transitioning of the bits.

SIZE Specify the size of the file. It can be specified using a floating point number followed by a ‘G’, ‘M’, ‘K’, ‘B’ or nothing as a scaling factor. The scaling factors multiply the floating point number given by 1000000000, 1000000, 1000, or 1 respectively.

filename

Specify the name of the file. If not specified, the *SIZE* parameter will be used as the file name.

EXIT STATUS

0 No issues were found.

1 The command failed.

EXAMPLES

To create a binary file which is 12.5 gigabytes long and is named "big_file":

```
% build_file binary 12.5G big_file
```

To create an alpha file which is 80 bytes long and is named "small_file":

```
% build_file alpha 80 small_file
```

To create a sparse file which is 80 megabytes long and is named "med_file":

```
% build_file sparse 80m med_file
```

SEE ALSO

build_verify(1)

NAME

build_verify − Used to verify the data integrity of a file previously built (created) by the build_file (1) utility

SYNOPSIS

build_verify

build_verify [**-f**] *filename*

build_verify help

DESCRIPTION

The **build_verify**(1) utility used to verify the data integrity (correctness) of a file previously built by the **build_file**(1) utility. It does this by reading the header block in the file. This block is 80 bytes long and contains the filetype (**binary**, **alpha**, **sparse**, or **trans**) and size of the file. Given this information, **build_verify**(1) reads the file and determines if the data in the file is exactly what it should be. The **build_verify**(1) utility uses the same logic as that used in the **build_file**(1) utility. If an error in the file data is encountered, an error message will be displayed giving the address of the offending byte and the program will terminate.

OPTIONS

help Will show long instructions.

-f Selects fast verification. This options only affects sparse files. Fast verification option skips over the holes in a sparse file and only verifies the live data written between them. Fast verification will not detect any error where the holes are supposed to be.

filename

The name of the file to verify.

EXIT STATUS

0 No issues were found.

1 The command failed.

SEE ALSO

build_file(1)

NAME

checkArchiveAvailabilityTsm − Verify that all configured archives are online.

SYNOPSIS

checkArchiveAvailabilityTsm [**-ras**]

checkArchiveAvailabilityTsm -report

DESCRIPTION

The **checkArchiveAvailabilityTsm**(1) command will verify that all configured archives are online.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

EXIT STATUS

0 No issues were found.

1 An issue was found. If the **-ras** option is specified, a RAS Alert will also be generated.

NAME

checkDiskSpaceTsm − Verify that enough disk space exists for the StorNext database tables, logging, and other functions.

SYNOPSIS

checkDiskSpaceTsm [**-ras**] [**-failpct** *failValue*] [**-warnpct** *warnValue*]

checkDiskSpaceTsm -report

DESCRIPTION

The **checkDiskSpaceTsm**(1) command will find all file systems that are in use by StorNext that are running out of space. This does not include user SNFS file systems. It does include all file systems that are accessible from /usr/adic, including those reached by symbolic link.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

-failpct *failValue*
Used to indicate the file system percentage utilization at which a failure will be generated. The *failValue* must be an integer between 0 and 100. The default value is 95.

-warnpct *warnValue*
Used to indicate the file system percentage utilization at which a warning will be generated. The *warnValue* must be an integer between 0 and 100. The default value is 99.

EXIT STATUS

0 No issues were found.

1 One or more filesystem has exceeded the **-failpct** percentage full. If the **-ras** option is specified, a RAS Alert will also be generated.

2 One or more filesystem has exceeded the **-warnpct** percentage full.

EXAMPLES

This example shows a successful run:

```
% checkDiskSpaceTsm
- Scanning files and directories in /usr/adic...
- OK: Filesystem /dev/sda3 is at 37%  (/usr/adic/DSM)
- OK: Filesystem /dev/sdc is at 42%  (/usr/adic/mysql)
- OK: Filesystem /dev/sdb is at 75%  (/usr/adic/DSM/bin/fs_fmover)
- Exiting with status 0 (Success)
```

This example shows a run with warnings:

```
% checkDiskSpaceTsm -failpct 95 -warnpct 75
- Scanning files and directories in /usr/adic...
- OK: Filesystem /dev/sda3 is at 37%  (/usr/adic/DSM)
- OK: Filesystem /dev/sdc is at 42%  (/usr/adic/mysql)
- WARNING: A StorNext file system has exceeded the warning threshold for percent
- WARN: Filesystem /dev/sdb is at 75%  (/usr/adic/DSM/bin/fs_fmover)
- Exiting with status 2 (Warn)
```

This example shows a run with failure:

```
% checkDiskSpaceTsm -failpct 75
- Scanning files and directories in /usr/adic...
- OK: Filesystem /dev/sda3 is at 37%  (/usr/adic/DSM)
- OK: Filesystem /dev/sdc is at 42%  (/usr/adic/mysql)
- ERROR: A StorNext file system has exceeded the maximum allowable threshold for
```

```
- FAIL: Filesystem /dev/sdb is at 75%  (/usr/adic/DSM/bin/fs_fmover)
- Exiting with status 1 (Fail)
```

NAME

checkDriveAvailabilityTsm − Verify that all configured drives are online.

SYNOPSIS

checkDriveAvailabilityTsm [**-ras**]

checkDriveAvailabilityTsm -report

DESCRIPTION

The **checkDriveAvailabilityTsm**(1) command will verify that all configured drives are online.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

EXIT STATUS

0 No issues were found.

1 An issue was found. If the **-ras** option is specified, a RAS Alert will also be generated.

NAME

checkDriveSlotToDrivePathTsm − Verify that device path and device slot mappings are correct for tape drives.

SYNOPSIS

checkDriveSlotToDrivePathTsm [**-ras**] [**-timeout** *secs*] [**-retries** *num*] [**-help**]

checkDriveSlotToDrivePathTsm -report

DESCRIPTION

The **checkDriveSlotToDrivePathTsm**(1) utility will verify that device path and device slot mappings are correct for tape drives. The utility does this by mounting and dismounting tapes in the each of the tape drives.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

-timeout *secs*
Used to indicate the timeout in seconds that the **checkDriveSlotToDrivePathTsm**(1) will wait for a mount or dismount to occur. The default value is 60 seconds.

-retries *num*
Used to indicate the number of retries passed to the **vsmount**(1) and **vsdismount**(1) commands. The default value is 2 retries.

-help Used to show the usage for this command.

EXIT STATUS

0 No issues were found.

1 The command failed. If the **-ras** option is specified, a RAS Alert will also be generated.

WARNING

The **checkDriveSlotToDrivePathTsm**(1) utility will stop and restart Tertiary Manager during the verification process.

SEE ALSO

vsmount(1), **vsdismount**(1)

NAME

checkEventsTsm − Verify that Tertiary Manager is keeping up with the file system events.

SYNOPSIS

checkEventsTsm [**-ras**] [**-maxentries** *num*] [**-staletime** *num*]

checkEventsTsm -report

DESCRIPTION

The **checkEventsTsm**(1) will verify that Tertiary Manager is keeping up with the file system events by verifying that there are less than **-maxentries** candidates in the Tertiary Manager event file(s), and that the oldest unprocessed event file is less than **-staletime** seconds old.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

-maxentries *num*

Used to indicate the maximum number of entries at which **checkEventsTsm**(1) will fail. The default value is 50000.

-staletime *num*

Used to indicate the maximum age (in seconds) of outstanding event files beyond which **checkEventsTsm**(1) will fail. The default value is 2700 seconds (45 minutes).

EXIT STATUS

0 No issues were found.

1 The command failed. If the **-ras** option is specified, a RAS Alert will also be generated.

2 There are greater than **-maxentries** candidates in the Tertiary Manager event file(s), or a pending events file is unmodified for more than **-staletime** seconds.

NAME

checkMediaAnomalies − List tape media having reached the *suspect* threshold value or having been marked *error*.

SYNOPSIS

checkMediaAnomalies [**-inactives**] [**-outfile <filepath>**] [**-ras**] [**-suspect <count>**]

checkMediaAnomalies -report

DESCRIPTION

The **checkMediaAnomalies**(1) command lists files that have active segments on tapes that are marked *error* or have a *suspect* count at or above the configured threshold.

OPTIONS

-inactives

Include in the report files having inactive segments on the affected tapes. The default report does not include these.

-outfile <filepath>

Print the list of files to the specified path rather than the configured path. The file path is overwritten unless append is configured (see below). The file contains tab-separated columns for the tape label (MediaID) and the file pathname.

-suspect <count>

Override the configured threshold *suspect* count for determining affected tapes.

-ras Send a RAS Alert when copy-segment paths are found on suspected or errored media.

-report Describe the purpose of this command.

FS_SYSPARM PARAMETERS

SUSPECT_MEDIA_FILELIST_FILE

The pathname of the output file for the report. Output is produced only when there are media that meet *error* or *suspect* criteria. The default pathname is: */usr/adic/TSM/reports/MediaAnomalies*

SUSPECT_MEDIA_FILELIST_FILE_APPEND

Each run of the *checkMediaAnomalies* utility overwrites the output file unless this parameter is *true*, in which case the results are appended to the file. The default is *false*.

SUSPECT_MEDIA_FILELIST_INACTIVE

Set to *true* to include inactive copies in the output of the *checkMediaAnomalies* utility. The default is *false*.

SUSPECT_MEDIA_FILELIST_RAS

This indicates that a RAS message is to be sent when the *checkMediaAnomalies* utility produces a report file. The default is *false*.

SEE ALSO

health_check(l)

FILES

/usr/adic/TSM/config/filelist

This file contains the list of executables that are run by the health_check utility. Add the following line to cause the checkMediaAnomalies program to run at the scheduled health_check time.

```
health_check : 0 : Media      : checkMediaAnomalies          : 0
```

EXIT STATUS

0 No issues were found.

1 An issue was found. When the **-ras** option is specified, a RAS Alert will also be generated.

NAME

fsCheckMediaAvailabilityTsm − Verify that there are enough media available for all policies to store all file copies.

SYNOPSIS

checkMediaAvailabilityTsm [**-ras**]

checkMediaAvailabilityTsm -report

DESCRIPTION

The **checkMediaAvailabilityTsm**(1) command will verify that there are enough media available for all policies to store all file copies.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

EXIT STATUS

0 No issues were found.

1 An issue was found. If the **-ras** option is specified, a RAS Alert will also be generated.

NAME

checkPolicyClassStore − Verify that each policy class can either be stored automatically or by the scheduler.

SYNOPSIS

checkPolicyClassStore [**-ras**]

checkPolicyClassStore -report

DESCRIPTION

The **checkPolicyClassStore**(1) command will verify that each policy class has the Auto Store option turned on or is scheduled to be stored. If a policy class fails the tests, the command will generate a report detailing the issues preventing the stores.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

EXIT STATUS

0 No issues were found.

1 An issue was found. If the **-ras** option is specified, a RAS Alert will also be generated.

NAME

checkStoreCandidates − Verify that Tertiary Manager is keeping up with store candidate processing.

SYNOPSIS

checkStoreCandidates [**-ras**] [**-maxentries** *numentries*] [**-maxerrors** *numerrors*] [**-help**]

checkStoreCandidates -report

DESCRIPTION

The **checkStoreCandidates**(1) will verify that Tertiary Manager is keeping up with store candidate processing by checking the number of candidates in each store candidate table.

OPTIONS

-help Used to show the usage for this command.

-maxentries *numentries*
Used to indicate the maximum number of store candidates for a filesystem at which **checkStoreCandidates**(1) will fail. The default value is 10000.

-maxerrors *numerrors*
Used to indicate the maximum number of files in each table that have an error count before **checkStoreCandidates**(1) will fail. The default value is 0.

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

EXIT STATUS

0 No issues were found.

1 The command failed. If the **-ras** option is specified, a RAS Alert will also be generated.

2 There are greater than *numentries* store candidates in a filesystem or greater than *numerrors* files with errors.

NAME

checkTsmToMsmMediaSync − Verify the SNSM media are configured correctly.

SYNOPSIS

checkTsmToMsmMediaSync [**-ras**]

checkTsmToMsmMediaSync -report

DESCRIPTION

The **checkTsmToMsmMediaSync**(1) command will verify the SNSM media are configured correctly by verifying that every media in the mediadir table exists in the archivemedia table. The command will output the number of media checked and the number that failed.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

EXAMPLE

```
% checkTsmToMsmMediaSync
- Number of Media Checked: 20
- Number of Media Failed : 0
- Exiting with status code = 0 (Success)
```

EXIT STATUS

0 No issues was found.

1 An issue was found. If the **-ras** option is specified, a RAS Alert will also be generated.

NAME

dbdropfs − This utility will perform cleanup for file systems that have been removed

SYNOPSIS

dbdropfs [**-F**] *mount_point*

DESCRIPTION

This command will remove all information for a specific file system from the database. This command can only be run on file systems which have been remade via the **cvmkfs**(8) command or for those which no longer exist. The TSM software must be restarted to pick up the changes. If immediate cleanup is desired, a **fsclean -B** may be executed.

OPTIONS

-F This will bypass the prompt for verification.

mount_point
The file system to operate on.

SEE ALSO

fsclean(1)

NAME

dbdrvslot − Queries for drive slot information about an archive.

SYNOPSIS

dbdrvslot *archiveName*

DESCRIPTION

The **dbdrvslot**(l) command is issued from the command line in order to obtain drive slot information about an archive from the database. This information consists of the following colon separated values:

- drive identifier
- hardware location information
- slot
- drive serial number

The slot value is required by the **vsdrivecfg**(l) command when configuring a new drive. There is no drive slot information for vaults so this command will return nothing if a vault is specified.

OPTIONS

archiveName

Identifies the archive to obtain drive slot information about.

EXIT STATUS

0 The command completed successfully.

1 An error is detected by the Media Manager software.

EXAMPLES

Request port information for a scsi archive before any drives have been configured

dbdrvslot lib3

Output returned:

```
-:d256l0:0,0,12,256:1210024470
-:d257l0:0,0,12,257:1210027302
```

Request port information for an acsls archive before any drives have been configured

dbdrvslot acslslib

Output returned:

```
-:0,0,1,1:0,0,12,0:
-:0,0,1,2:0,0,12,1:
```

SEE ALSO

vsdrivecfg(l)

NAME

dirq_usage − Produce a per-share report of disk and tertiary storage usage

SYNOPSIS

dirq_usage

DESCRIPTION

The **dirq_usage** program calls the **dirq_usage_engine** program for each StorNext managed file-system device to create a report file of disk and tertiary storage usage per share for each file system. It automatically generates the input files per file system.

The Directory Quotas Name Space (DQNS) feature must be provisioned for each share directory. Setting a value of zero for the hard limit, soft limit, and grace period identifies the share directory and provides for tracking disk-space usage without limiting the amount of space used in the directory. See the *snquota* command for information. Any tertiary storage space that is not in a share directory is reported as a single value for *non-share files*.

This command can only be run on an MDC that is in *primary* mode, that is, the MDC that is active running the Storage Manager software.

Each file-system device has an output file in comma-separated-values format (CSV) with the following path name:

```
/var/shareUsageReports/<timestamp>/<mount name>/<mount name>_<timestamp>
```

For example:

```
/var/shareUsageReports/20151217111117/stornext_snfs1/stornext_snfs1_2015121711111
```

The output files from previous runs may be on either MDC since the path is not on a shared file system.

The CSV output file is suitable for importing into a spreadsheet where a pivot-table function can be used to sum the total space per share. The columns of the CSV output file are as follows:

```
Path Name, Disk-Space Size in Bytes, Tertiary-Space Size in Bytes
```

The digits of the timestamp are: year(4), month(2), day(2), hour(2), minute(2), second(2).

Note: Location information for a copy is recorded when the copy is made. When a file's location is changed to a different share, the **dirq_usage** report will continue to report the file's usage in the original share until the file is changed and a new copy is made. This discrepancy can be viewed with the **fsfileinfo** command in its *Stored Name* field.

SEE ALSO

dirq_usage_engine(1), **snquota**(1)

NAME

dirq_usage_engine − Process data for a disk-space and tertiary-space usage report

SYNOPSIS

dirq_usage_engine [**-dh**] **-f** *join_table_file* **-o** *output_file* **-p** *dirpath_file* **-s** *quota_report.csv*

DESCRIPTION

The **dirq_usage_engine** program generates a comma-separated-values (CSV) report of disk-space and tertiary-space usage per share for one file system. It is intended to be called by the **dirq_usage** script, which generates the input files per file system.

Share paths must be unique across share names and must not be nested. When duplicate or nested paths occur, a warning is printed and these paths are discarded in a consistent manner for the purposes of reporting usage.

OPTIONS

-d Debug mode. This generates information per share and per file, so the information can be very large on production systems.

−f *join_table_file*
CSV file with the results of joining the fileinfo and filecomp tables for one file-system device. The columns are: parent file key, file key, size in bytes for one segment. There is no header row.

-h Displays the usage of the command.

−o *output_file*
CSV report of disk and tertiary storage usage per share for one file system. The columns are: share pathname, quantity of disk space usage in bytes, quantity of tertiary space usage in bytes. There is no header row. The size is bytes of non-compressed data, so the value may need to be prorated for storage destinations that use compression. Alternatively, the charge-back rate could simply be scaled to reflect the cost of compressed storage. The non-share tertiary storage used for the file-system device is accumulated into a pseudo share with path name *non-share files*. The disk-space usage for non-share files is not available to this program, so it is always reported as zero.

−p *dirpath_file*
CSV rows of the dirpath table for one file-system device. The columns are: parent file key, path below the mount point. There is no header row.

−s *quota_report.csv*
Directory Quotas Name Space report output file for the file-system device.

EXIT VALUES

Returns zero on success and non-zero on failure.

SEE ALSO

dirq_usage(1)

NAME

diva_db_export − Query DIVA database for metadata input to the **fsimport_diva** command

SYNOPSIS

diva_db_export -d *DIVA_MySql_database_name* -o *output_file*

DESCRIPTION

diva_db_export(1) runs a query on an imported DIVA MySql database to collect all the metadata needed for input to the fsimport_diva command. The *output_file* target path must be writable by the *quantumdb* user. The *output_file* will be removed and replaced if it exists.

Refer to the **fsimport_diva**(1) man page for an example of the SQL statement.

EXAMPLES

diva_db_export -d diva_import -o /tmp/selected_diva_metadata

Selects metadata from the *diva_import* database, and writes it into the */tmp/selected_diva_metadata* output file.

SEE ALSO

diva_sm_dbload(1), **diva_db_export**(1), **fsimport_diva**(1), **sm_diva_media_cleanup**(1), **sm_diva_media_import**(1),

NAME

diva_sm_dbload − Load the CSV files generated by fsimport_diva into Storage Manager database

SYNOPSIS

diva_sm_dbload [**-h**] -i *CSV_files_directory* -p *file_system_mount_point*

DESCRIPTION

diva_sm_dbload(1) automates the loading of the CSV files that were generated by running the *fsimport_diva* command into the StorNext database. This program must be run as the *root* user.

OPTIONS

-h Prints the command help message.

-i *CSV_files_directory*
Names the directory where the *fsimport_diva* command wrote five CSV output files.

-p *file_system_mount_point*
Names the file-system mount-point directory where the DIVA files are being imported.

SEE ALSO

diva_db_export(1), **fsimport_diva**(1), **sm_diva_media_import**(1), **sm_diva_media_cleanup**(1)

NAME

dm_altstoretest − Test path names for REMOTE_COPY_ENABLED

SYNOPSIS

dm_altstoretest [**-s**|**-h**]

DESCRIPTION

Use this filter when searching for pre-existing files to add to the remote file copies made by the Alternate Store Location feature. It reads file path names from input, tests if they are enabled for Alternate Store Location copy to remote, ignores path names that are enabled or prints path names to stdout for those that are not enabled. Any irregularities, such as path names for non-existing files, are printed to stderr.

OPTIONS

-s do not print path names for non-existing files

-h print help information to stderr

EXAMPLE

Following is an example use for the filter.

```
# One-time script for finding pre-existing files for potential
# background copying by the Alternate Store Location (ASL) feature.
# Change the parameters of the 'find' command to be: 1) the full
# path of the ASL-enabled relation point, and 2) a date & time
# later than the activation of ASL on that relation point.

find /stornext/snfs1/rel_pt -type f ! -newermt "time string" ! -empty |
    dm_altstoretest > backgrd_candidates 2>altstoretest_log_file
if [ -s altstoretest_log_file ]
then
    echo Problems:
    cat altstoretest_log_file
    exit 1
fi
split -l 10000 backgrd_candidates backgrd_candidates_part_

#############################################################################

# Restartable script for spoon-feeding the background-copy
# candidates in quantities that limit the performance impact
# on database transactions.
function countrows () {
  rowcount=`altstoremod |
  sed -n -e '/=/ s/.*=\([0-9]*\).*=\([0-9]*\)/\1 \2/p' |
    awk '{a=a+$1+$2} END {print a}' `
  eval [ $rowcount -ge 10000 ]
  return
}

for background_slice in backgrd_candidates_part_*
do
    while countrows
    do
        sleep 30
    done
    dm_altstoretest < $background_slice 2>>altstoretest_log_file2 |
        altstoreadd >> added_files
```

done

SEE ALSO

altstoreadd(1)**, dm_util**(1)**, fsfileinfo**(1)

NAME

dm_foreign − Set extended attribute information for foreign files and directories

DESCRIPTION

This utility is used by the **fsimportnamespace**(1) script to create an inode with foreign flags and events set. Users should not run this command, unless directed to do so by Quantum technical assistance.

NAME

dm_info − Display extended attribute and inode information

SYNOPSIS

dm_info -F [**-H**] *file*...

dm_info -N [**-H**] *file*...

dm_info [**-A**|**-h**|**-H**|**-d**|**-i**|**-e**|**-s**|**-o**|**-a** *attr_arg*] *file*...

DESCRIPTION

This will display extended attribute and inode information used by the Tertiary Manager software. This includes an **extended attribute** and zero or more **external representations**.

The user must be root or the utility must be owned by root with the 's' bit set.

The **extended attribute** contains information about the file set and used by the Tertiary Manager software. The fields contained in the **extended attribute** are fixed, but vary somewhat between Quantum product releases. Each **external representation** contains information about one copy's stored segment(s) of the file, if any. The fields contained in each **external representation** segment depend on the medium the individual file segment is stored on (if any), and may also vary with the Tertiary Manager product release. These extend and replace now-deprecated **object ID**s.

OPTIONS

-F show file system hdl and event set of the file system

-N show extended attributes names

-A show affinity association

-h show file handle

-H interpret file argument(s) as file handle(s)

-d show device number and fsid

-i show inode and generation number

-e show event mask

-s show association

-o show object ids

-a *attr_arg*

show **extended attribute** field *attr_arg*, where *attr_arg* can be any of:

all	**class**	**cpymap**	**flags**
fsn*	**key**	**ldbn***	**medlist***
oneup	**reserved***	**slen***	**stublen**
stubsize	**totseg***	**totvers**	**vsn**

* deprecated - not present on newer files

Multiple **-a** flags and arguments may be used together.

file The names of the file(s) on disk to report on. The file path(s) must be in a managed file system. The entire path name need not be entered. If preceded by a slash (/) in the command definition, the full path name, starting from the root directory, is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent. If multiple files are entered, the files must be separated by spaces.

If the **-H** flag is set, each "file" should be a comma-separated tuple *fsid*,*inode-number*,*generation-number* specifying the file to be examined. The *inode-number* and *generation-number* must be entered in decimal, and the *fsid* in hexadecimal.

EXAMPLE

An example of the output is:

```
Filename:     myfile
handle (hex): 00054b300f55dd91000e00000000001000000000000041
   fsid: 0x00054b300f55dd91  dev: 23  rdev: 23  aff: n/a
   size: 1337    block size: 4096    number of blocks: 8
   ino: 65   gen: 1   type: S_IFREG   mode: 0100644   lnks: 1
   extended attributes:   attrname: SNEA   attr_ver: 7.40   key: 5
      class: 2  oneup: 5  vsn: 001  totvers: 1  cpymap: 0x1
      flags: 0x2   events: 0x160000   stub size,len: -1,0
      flags: ALL_COPIES_MADE
   objids: NONE
   eventlist:  0x00160000  WRITE  TRUNCATE  DESTROY
   region events: WRITE  TRUNCATE
   atime = 1490043921 -> Mon Mar 20 16:05:21 2017
   ctime = 1490043921 -> Mon Mar 20 16:05:21 2017
   mtime = 1490043921 -> Mon Mar 20 16:05:21 2017
  copy: 1
    offset: 0 length: 1337 bytes: 34
      medium: FC9A5D1A-9920-43EE-A190-A70502CF6FAB
      seg_time: 1490043921 [Mon Mar 20 16:05:21 2017]
      add_date: 1490044270 [Mon Mar 20 16:11:10 2017]
      oneup: 5
      version: 1
```

Older files may contain different versions of the extended attribute, and include additional fields. E.g., note below the **attr_ver** "5.82", indicating a version 5 SNEA attribute, 82 bytes long, and the additional fields present:

```
Filename:     myfile
handle (hex): 0003dc583e00125a000e0000000000000000000000000007
  fsid: 0x0003dc583e00125a  dev: 67895298  rdev: 67895298  aff: n/a
  size: 3000009    block size: 4096    number of blocks: 5888
  ino: 7   gen: 0   type: S_IFREG   mode: 0100664   lnks: 1
  extended attributes:   attrname: SNEA   attr_ver: 5.82   key: 6
    class,sub: 3,0  oneup: 6  vsn: 000  totvers: 2  totseg: 0
    cpymap: 0x1   flags: 0x0   events: 0x180000   stublen: 0
    flags:  NONE
  objids:                    Objid           Offset           Length
      04c0dff7aa2004fe0a3f3001001                0             1000
  eventlist: ATTRIBUTE  DESTROY
  region events:
  atime = 1086697327 -> Tue Jun  8 07:22:07 2004
  ctime = 1088686531 -> Thu Jul  1 07:55:31 2004
  mtime = 1088686531 -> Thu Jul  1 07:55:31 2004
```

Attempts to print **extended attribute** fields that do not exist in a file's **extended attribute** are ignored.

SEE ALSO

dm_util(1)

NAME

dm_trunc − Truncates a file.

SYNOPSIS

dm_trunc *startoffset count file...*

DESCRIPTION

This will truncate a file. It is highly recommended that the **fsrmcopy**(1) be used instead of this utility. There should be no need to use this utility unless directed to do so by Quantum technical assistance.

This utility must be run as root or be owned by root with the 's' bit set.

OPTIONS

startoffset

This is the offset within the file where the truncation should start to occur. This should be on a disk block boundary.

count The number of bytes to truncate. If set to 0, then all bytes from the specified offset will be truncated and the startoffset is automatically adjusted to be on a block boundary. Currently, only 0 is supported until managed regions is supported.

file The names of the file(s) on disk to truncate. The file path(s) must also be in a managed file system. The entire path name need not be entered. If preceded by a slash (/) in the command definition, the full path name, starting from the root directory, is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent. If multiple files are entered, the files must be separated by spaces.

SEE ALSO

fsrmcopy(1), **fsrmdiskcopy**(1)

NAME

dm_util − Set extended attribute information for a file

SYNOPSIS

dm_util -D *mnt_pt...*

dm_util -r *file...*

dm_util -x [**-v**] [**-k**] [**-c** *copynum* [**-c** *copynum* [...]]] *file...*

dm_util -X [**-v**] [**-c** *copynum* [**-c** *copynum* [...]]] *file...*

dm_util -T *tmp_type mnt_pt*

dm_util -A *affinity_file...*

dm_util -a|-d *eventset| flags file...*

dm_util -u *attr_arg attr_val... file...*

dm_util [**-v**] **-o** *objid_info_str file*

dm_util [**-F**] **-E** *eventlist_bits file*

WARNINGS

This utility should be used VERY carefully and only under the guidance of Quantum technical assistance.

DESCRIPTION

This utility can be used to modify flags and event settings for files managed by the Tertiary Manager software. It can also be used to disassociate (remove all) events from a file system or create special temporary directories in the file system.

The user must be root or the utility must be owned by root with the 's' bit set.

OPTIONS

-D *mnt_pt*

Disassociate (remove all) events from the indicated mount points.

-r Remove all events, flags, or keys for a file or directory.

-x Instantiate the file's external representations ("xreps") from the file's filecomp database records. The command defaults to all copies; if copies are specified with the **-c** option, only those copies will be instantiated; others will be unaffected. By default, any (now obsolete) objids associated with the file will be removed. If the **-k** option is specified, file objids will be kept rather than removed.

-X Delete the file's external representations ("xreps"). The default is to delete all copies' external representations; if copies are specified with the **-c** option, only those copies will be deleted; others will be unaffected. (Any filecomp records are unaffected.)

-T *tmp_type*

Create StorNext temp directory of type *tmp_type* on specified mount point. Valid values for *tmp_type*:

jnl : used for journaling
rel : used for file relocation feature
for : used for foreign migration feature
alt : used for alternate node retrieval feature

-A *affinity*

Set the specified *affinity* on files and directories. The affinity must be one that is currently configured in the file system. Specify *none* to clear the affinity setting on files and directories.

-a *eventset|flags*

Add the specified *eventset* or *flags* to a file.

-d *eventset|flags*
Delete the specified *eventset* or *flags* from a file.

eventset
Valid values are:

admin data attr inherit inherited migr
post read write tape_copy blocklet
all (delete only)

The **admin** event set is valid for directories only. All other event sets are for files.

flags Valid values:

all_copies no_reloc remote_copy_exists tier_override
foreign_filesystem no_store remote_copy_enabled trunc_exclude
in_process no_trunc stranger_media trunc_immed
invalid_path

-u class <*class_ndx*>|RECYCLE|TEMP|NONE
Updates the class to the specified class index. For special cases use keywords of **RECYCLE** for Recycle bin, **TEMP** for StorNext temp dir and **NONE** for no class.

-u media *segment media_index generation*
When setting media values, provide the segment number, the media index and media generation number. Example:

```
% dm_util -u media 2 347 1 filea
```

Where **2** is the segment number, **347** is the media index and **1** is the media generation number for that segment.

-u *attr_arg attr_val...*
Update the indicated attribute for a file or directory. Unless so indicated, attribute updates take one attr value.

attr_arg
Valid values are:

key class vsn oneup
stublen stubsize cpymap totvers
ldbn* slen* fsn* totseg*
media* objid reserved*

* deprecated - not present on newer files

The deprecated fields are not present in version 7 extended attributes, and may only be set on files with version 6 or earlier extended attributes.

When updating **cpymap** the *attr_val* must be provided in hex.

[**-v**] **-o** *objid_info_str*
Generate the object id for the indicated file using the object id information provided. Formatting information for the input string:

- Input format:
  fsid,ino,gen,ctime,mtime,pid,tid,cpy,vsn,seg,offset
- Field specifics:
  fsid - format of dm_info output (ex. 0004cb296df4ba27)
  ctime/mtime - date string format: YYYY:MM:DD:hh:mm:ss
  (ex. 2012:10:04:07:35:25) or Unix time
  (ex. 1349354125)
  If the last 3 items are not provided they default as follows:

vsn: 1, seg: 1, offset: 0
- If '-v' provided then report the input values for the object id

With the version 7 **extended attribute**, the **objid** attribute is deprecated. It may still be set using **dm_util**, but it is not used by Tertiary Manager software, and may be cleared by it as part of the normal upgrade procedure.

When updating the **objid** the *attr_val* provided will be a string that indicates: **objid**,offset,length

(That is the **objid** value and the offset and length for the data where the object id applies.)

Some samples of the *objid* attr value:

- Set objid for entire file
  . abcdefghijklmonp,0,0
- Set objid for offset/length: 0/100000
  . abcdefghijklmonp,0,100000
- Set objid for offset/length: 100000/100000
  . 1234567890123456,100000,100000
- Remove existing objid at offset 0
  . 0,0
- Remove existing objid at offset 100000
  . 0,100000

[**-F**] **-E** *eventlist_bits*

Set/Update events on the specified file or directory using the *eventlist_bits* specified. Valid options for the eventlist_bits:

0xnnnnn - set eventset to 0xnnnnn.

+0xnnnnn - add bits 0xnnnnn to eventset.

-0xnnnnn - clear bits 0xnnnnn from eventset.

+0xnnnnn:-0xmmmmm - add bits 0xnnnnn to eventset and clear bits 0xmmmmm from eventset.

- If '**-F**' provided then apply the changes to the filesystem.

mnt_pt file system mount point.

file The name(s) of the managed file(s) to update.

SEE ALSO

dm_info(1)

NAME

exclusions − file exclusions

SYNOPSIS

/usr/adic/TSM/config/excludes.store

/usr/adic/TSM/config/excludes.truncate

/usr/adic/TSM/config/excludes.postrestore

DESCRIPTION

These files allow criteria to be specified which will exclude files from store, truncation or post restore operations. The syntax of all the files is identical. Any updates to these files will not require TSM to be cycled for them to go into effect.

NOTE: with exclusion specifications being openly defined it is possible to potentially exclude files that may be critical for disaster recovery.

The format of this file is strict and each line must begin with one of the following specifiers: **#** , **EXACT:** , **CONTAINS:** , **BEGINS:** , **ENDS:** , **MATCH:** , and **MATCH_PATH:**

Each entry in the exclusion file must start in column one.

Store Exclusions

To facilitate identifying files to be excluded from store operations an exclusion file (*/usr/adic/TSM/config/excludes.store*) can be used. Filenames (node name, not full path names) are tested against the exclusions specified during preparation for stores and if the filename matches an exclusion the file is not included in the store operation. One caveat to this behavior occurs when files have been renamed after they have been stored. It will not matter if the new file meets or does not meet the exclusion criteria as the exclusions are applied against the original stored pathname. If it is unclear that this condition exists, then running the **fsfileinfo**(1) report on the file will display the stored path if it is indeed different the the current file path.

Truncation Exclusions

To facilitate identifying files to be excluded from truncation operations an exclusion file (*/usr/adic/TSM/config/excludes.truncate*) can be used. Full path names are tested against the specified exclusions as files are being stored. If the pathname matches an exclusion then the file will be marked so that it will not be truncated. This has the same effect as running **fschfiat -t** *e* on each file.

Disaster Recovery Exclusions

fspostrestore(1) is used when recovering a file system after a disaster. In addition to its primary restore responsibilities it can also create a file which contains a list of all the files in the file system that are excluded from truncations. This list includes files that either met the truncation exclusion criteria when it was stored or that have been explicitly marked with **fschfiat -t** *e*. The files are referred to as "no truncate" files. This list can be used in conjunction with the **fsretrieve -B** option to re-stage these files back to disk.

If some of these files are considered to be more critical than others, then there may be a need to separate them such that some are retrieved before others. In order to accomplish this, the */usr/adic/TSM/config/excludes.postrestore* exclusion file can be used to specify criteria which is used to filter the "no truncate" files.

Full path names are tested against the specified exclusions as files are being examined by **fspostrestore**(1). If the pathname matches the criteria then the file path will be written to a second list which results in two lists of files being generated instead of one. Then **fsretrieve -B** can be invoked for each one of the lists.

Exclusion File Tags:

This is the comment identifier. The remainder of the line is ignored.

EXACT:

A file name/path is matched if it is identical to this string.

CONTAINS:

A file name/path is matched if any part of this string appears in it.

BEGINS:
A file name/path is matched if it begins with this string.

ENDS: A file name/path is matched if it ends with this string.

MATCH:
A file name/path is matched if it matches the specified shell expression. This tag is different from the others as it will expand shell wildcards when evaluating a file. For example an asterisk (*) will match anything when used with this tag but only match an asterisk when used with the other tags.

Wildcards:
A string is a wildcard pattern if it contains one of the characters ?, * or [. These wildcard characters are expanded according to the following definitions:

A ? (not between brackets) matches any single character.

A * (not between brackets) matches any string, including the empty string.

Character classes:
An expression [...] where the first character after the leading [is not an ! matches a single character, namely any of the characters enclosed by the brackets. The string enclosed by the brackets cannot be empty; therefore] can be allowed between the brackets, provided that it is the first character. (Thus, [][!] matches the three characters [,] and !.)

Ranges:
There is one special convention: two characters separated by - denote a range. (Thus, [A-Fa-f0-9] is equivalent to [ABCDEFabcdef0123456789].) One may include - in its literal meaning by making it the first or last character between the brackets. (Thus, []-] matches just the two characters] and -, and [--0] matches the three characters -, ., 0, since / cannot be matched.)

Complementation:
An expression [!...] matches a single character, namely any character that is not matched by the expression obtained by removing the first ! from it. (Thus, [!]a-] matches any single character except], a and -.)

One can remove the special meaning of ?, * and [by preceding them by a backslash. Between brackets these characters stand for themselves. Thus, [[?*] matches the four characters [, ?, * and .

MATCH_PATH:
This is identical to MATCH except that it will only match a slash in a filename/path with a slash in then pattern and not by an asterisk (*) or a question mark (?) metacharacter, nor by a bracket expression ([]) containing a slash. So every slash in the file path must match a slash in the specified exclusion string.

EXAMPLES: store exclusions

A comment line

```
# Exclude definitions for temporary files
```

Will exclude "temp_work.tmp" and "backup.out", but no other files

```
EXACT:temp_work.tmp
EXACT:backup.out
```

Will exclude "exclusions.man", "testrun.temp", "temp.logs", and "temporary", but not "manifest" or "sherman"

```
CONTAINS:.man
CONTAINS:temp
```

Will exclude "tmp.file", "tmpstuff", and "temporary", but not "file.tmp", "file.tmp.1", or ".temporary"

```
BEGINS:tmp
BEGINS:temp
```

Will exclude "work.tmp" and "work.temp", but not "temp.work" and "tmp.work"

```
ENDS:.tmp
ENDS:.temp
```

Will exclude "tmp.file", "tmpstuff", and "temporary", but not "file.tmp", "file.tmp.1", or ".temporary"

```
MATCH:tmp*
MATCH:temp*
```

Will exclude any file starting with "a", "c", "e"

```
MATCH:[ace]*
```

Will exclude "temp.1", "tempXY", and "file.temp.1", but not "temporary", "tempABC" or "file.temp"

```
MATCH:*temp??
```

Will exclude "tmp.file", "tmpstuff", and "temporary", but not "file.tmp", "file.tmp.1", or ".temporary" Since store exclusions are based on file names and not paths, MATCH_PATH will yield the same results as MATCH

```
MATCH_PATH:tmp*
MATCH_PATH:temp*
```

EXAMPLES: truncate/postrestore exclusions

A comment line

```
# Exclude definitions for temporary files
```

Will exclude "/sn/fs1/temp_work.tmp" and "/sn/fs1/backup.out", but not "/sn/fs9/temp_work.tmp"

```
EXACT:/sn/fs1/temp_work.tmp
EXACT:/sn/fs1/backup.out
```

Will exclude "exclusions.man", "testrun.temp", "temp.logs", "temporary", anything in the "/sn/fs1/temp/" directory, but not "manifest", "sherman" or "/sn/fs1/xman/file"

```
CONTAINS:.man
CONTAINS:temp
```

Will exclude "/sn/fs1/dir1/tmp.file", and "/sn/fs1/dir1/tmpstuff", but not "/sn/fs1/dir1/file.tmp", or "/sn/fs1/dir2/tmp.file"

```
BEGINS:/sn/fs1/dir1/tmp
```

Will exclude "work.tmp" and "work.temp", but not "temp.work" and "tmp.work"

```
ENDS:.tmp
ENDS:.temp
```

Will exclude "/sn/fs1/dir1/tmpfile", "/sn/fs2/dir1/tmpfile", but not "/sn/fs1/dir2/tmpfile"

```
ENDS:/dir1/tmpfile
```

Will exclude "tmp.file", "tmpstuff", and anything under the "/sn/fs1/tmp/" directory, but not "file.tmp", or "file.tmp.1"

```
MATCH:*/tmp*
```

Will exclude any file in the "/sn/fs1/dir1/" directory and any subordinate directory such as /sn Will exclude "/sn/fs1/dir1/file1", "/sn/fs1/dir1/dir2/file2"

```
MATCH_PATH:/sn/fs1/dir1/*
```

Will exclude "tmp.file", "tmpstuff", and "/sn/fs1/dir/tmp.file", but not "file.tmp", or "/sn/fs1/dir2/tmp.file"

MATCH:*/dir1/tmp*

Will exclude "/sn/fs1/.profile", and "/sn/fsZ/.profile", but not "/sn/fs10/.profile" or "/SN/fs1/.profile"

MATCH:/sn/fs?/.profile

Will only exclude "/fs1/file", "/fs2/file", "/fs3/file" or "/fsZ/file"

MATCH:/fs[1-3Z]/file

Will exclude any file in the "/sn/fs1/dir1/" directory but not any files in subordinate directories such as "/sn/fs1/dir1/dir2/file"

MATCH_PATH:/sn/fs1/dir1/*

FILES

/usr/adic/TSM/config/excludes.store
/usr/adic/TSM/config/excludes.truncate
/usr/adic/TSM/config/excludes.postrestore

SEE ALSO

fschfiat(1), **fspostrestore**(1), **fsretrieve**(1)

NAME

fhpath − Generate file path from handle

SYNOPSIS

fhpath *phandle fhandle*

DESCRIPTION

The fhpath utility will convert the specified file handle and parent file handle into the associated path.

OPTIONS

phandle Parent handle in hex. (Use 0 if the parent handle is unknown.)

fhandle File handle of file or directory in hex.

SEE ALSO

dm_info(1)

NAME

filesystems

SYNOPSIS

/usr/adic/TSM/config/filesystems

DESCRIPTION

The file system configuration file contains the file systems that are being managed by the TSM software. Each file system is listed along with some configuration parameters for the file system. File system entries are automatically added to the file when the **fsaddrelation**(1) command is used to add the first relation to a file system. Also, entries are automatically removed from this file when the **fsrmrelation**(1) command is used to remove the last relation from a file system.

NOTE - A line in the file with a first character of '#' will be treated as a comment line. A line with a first character of '/' will be treated as a file system entry. Tabs or spaces may be used between parameters in a file system entry.

NOTE - When an entry is added by the **fsaddrelation**(1) command, the entry is added with the default parameters (shown below). If those parms are to be changed, edit the file and make the desired updates.

WARNING - Do not add or remove entries from the file by hand. Allow that to be done automatically by the **fsaddrelation**(1) and **fsrmrelation**(1) commands. You may, however, update file system parameters by modifying the file.

In order to manage file data and maintain adequate disk space for a file system, the TSM software will relocate data blocks of existing files from one affinity to another or truncate data blocks of existing files from disk. These operations are performed automatically but can be applied manually by using the **fsrelocate**(1), **fsrmcopy**(1) or **fspolicy**(1) commands. Certain criteria apply for file data to be eligible for relocation or truncation (see man pages for the above commands).

The automatic relocation and truncation of managed files is accomplished via **fspolicy**(1) commands. A set of daemons issue these commands as they are needed. The parameters in the filesystems configuration file specify the target fill level to be used by **fspolicy**(1) for each file system. These parameters are used by two policy schemes: Space Based and Min Time. In general, both schemes will generate a list of candidates sorted by access time from oldest to youngest to relocate or truncate. The access time is saved in candidate tables when a file is stored. To eliminate unnecessary database activity, it is not kept current upon each file access. The candidate table access time is updated as needed when relocation and truncation policies are run. This means there is no guarantee that files will get processed in the exact access age order.

Policy Schemes:

Spaced Based Scheme:

This scheme is used to automatically maintain the file system fill level within a specified range. Relocation or truncation policies are started when the file system fill level goes above the *High Use* percentage.

Then a fixed number of candidates is generated that have reached the oldest minreloctime or mintrunctime for all policy classes. The number of candidates is determined by a system parameter. If no candidates are found, then the next oldest minreloctime or mintrunctime is used until some candidates are found. The files in the candidate list are then relocated or truncated in an attempt to reach the *Low Use* file system fill percentage. The processing will stop after the list of candidates has been processed, even if the *Low Use* goal has not been met.

There may be occasions where the fill level is not maintained within the range. Emergency relocation and truncation policies are started when the file system indicates it is out of space or the file system used space is greater than 99% of the file system capacity. This means the file system **df** command could indicate that there is some available space but reports its usage to be at 100%. When either of these conditions occur, a "quick" list of the largest files available is built. The files in the list are then relocated or truncated in an attempt to reach the *Low Use* file system fill percentage. Processing will stop after the "quick" list of candidates has been processed, even if the

Low Use goal has not been met.

Min Time Scheme:

Mintime relocation and truncation policies are started daily at a specific time defined by system parameters. A mintime policy is run for each policy class and will generate a list of candidates that have reached their minreloctime or mintrunctime, as specified by the policy class. The candidates are processed until the target fill percentage is reached or there are no more candidates to process. When *Enable Min Use* is true, the target fill level is *Min Use*, otherwise the *Low Use* value is used.

This scheme helps to manage the files that never make it into candidate lists processed by space based policies. These policies are run to more accurately apply the relocation and truncation parameters defined in the configured policy classes and to pro-actively keep the fill level below the *High Use* percentage.

This scheme also provides more flexibility in having old files truncated from disk. You can set *Min Use* lower than *Low Use* so that old files will be truncated even if the file system is already below *Low Use*. Setting *Min Use* to 0 would cause all valid candidates to be truncated regardless of the file system fill level.

File System Configuration Parameters

Each entry in the configuration file is a line of fields separated by spaces/tabs of the form:

Mount_Point Low_Use High_Use Enable_Min_Use Min_Use Enable_Truncation

Low Use

The target fill level for a relocation or truncation policy.
(Default: 75) It is given as a percentage of used blocks.

High Use

The fill level at which the space-monitoring daemon automatically runs a space based policy attempting to reach the *Low Use* fill level.
(Default: 85) It is given as a percentage of used blocks.

Enable Min Use

This flag indicates which fill level mintime policies will try to achieve. If set to *true*, then *Min Use* will be used for the target file system fill level. If set to *false*, then *Low Use* will be used.
(Default: true)

Min Use

The target fill level for a mintime truncation policy (a policy that uses the -m option) when *Enable Min Use* is true.
Note - If *Enable Min Use* is false, this parameter has no effect, and the *Low Use* value will be the target for any mintime truncation policies run on the file system.
(Default: 75) It is given as a percentage of used blocks.

Enable Truncation

This flag indicates whether file truncation is enabled (*true*) or disabled (*false*) for the given filesystem.

When truncation is enabled, files are automatically truncated in order to free up space. A user action causing a no-space condition will be blocked until free space is available.

When truncation is disabled, no automatic truncation takes place. This means emergency truncation policies are not run either. A user action causing a no-space condition will fail with ENOSPC error.
Note - when truncation is disabled it is still possible to truncate files using **fsrmdiskcopy**(1) utility.
(Default: true)

ENVIRONMENT

fspolicy(1) behavior can be adjusted by overriding any of the following system parameters in the *fs_sysparm_override* file. See *fs_sysparm.README* file for more details of each of them.

POLICY_MAX_TRUNC_ENTRIES
: The maximum number of files to truncate each time for a space based file system truncation policy. It is also used for class based policy queries.
default: 3000 files

POLICY_MAX_RELOC_ENTRIES
: The maximum number of files to relocate each time for a space based file system truncation policy. It is also used for class based policy queries.
default: 3000 files

POLICY_TRUNC_MINSIZE
: Minimum file size in bytes to be passed by daemons to fspolicy when mintime truncation policies are run. Emergency truncation policies will ignore this.
default: 1

DAILY_TRUNC_START_TIME
: This specifies the starting time for the class-based daily truncation mintime policies. The value is a time, formatted as HHMM.
default: 0000 (or midnight)

DAILY_RELOC_START_TIME
: This specifies the starting time for the class-based daily relocation mintime policies. The value is a time, formatted as HHMM.
default: 0000 (or midnight)

EXAMPLES

The following filesystems config settings will be used for the examples:

```
# File System          Low     High    Enable        Min     Enable
# Mount Path           Use     Use     Min Use       Use     Truncation
# ------------         --------------  -----------------     ----------
/stornext/snfs1        75      85      false         65      true
/stornext/snfs2        75      85      true          35      true
/stornext/snfs3        75      85      true           0      true
/stornext/snfs4        75      85      true           0      false
```

Example 1

Assume the snfs1 file system is currently 90% full, and there are valid candidates that can be used to manage the file system down to 65%; Also assume a spaced based policy generates a candidate list that can only reclaim 10% disk space.
A space based policy will stop at 80% fill level. A mintime policy will stop at the 75% *Low Use* level even though there are more valid candidates. *Low Use* is used because *Enable Min Use* is false.

Example 2

Assume the snfs2 file system is currently 50% full, and there are valid
candidates that can be used to manage the file system down to 20%. A spaced based policy will not run because the current fill level is below the 75% *Low Use* setting. A mintime policy will stop at the 35% *Min Use* level even though there are more valid candidates.

Example 3

Assume the snfs3 file system is currently 90% full, and there are valid candidates that can be used to manage the file system down to 20%. Also assume a spaced based policy generates a candidate list that can reclaim 30% disk space. A space based policy will run and stop at 75% even though it found enough candidates to reduce the fill level to 60%. A mintime policy will stop at 20% because that's all the candidates that there are to process.

Example 4

Assume the snfs4 file system is currently 90% full, and there are valid candidates that can be used to manage the file system down to 20%. Neither space based or mintime policies will run because *Enable Truncation* is false. The other 4 settings for snfs4 don't matter as they are never used in this scenario.

SEE ALSO

fspolicy(1), **fsrelocate**(1), **fsrmcopy**(1), **fsclassinfo**(1), **fsaddclass**(1), **fsmodclass**(1)

NAME

fsactivevault − Run an active vault policy

SYNOPSIS

fsactivevault [**-archive** *a1,...*] [**-vault** *dest*] [**-copy** *c1,...*] [**-used** *size*] [**-remaining** *size*] [**-age** *age*] [**-sort** *column*] [**-migrate**|**-nomigrate**] [**-pending**|**-nopending**] [**-highmark** *pct*] [**-lowmark** *pct*] [**-fullpct** *pct*] [**-report**] [**-include-policy** *p1,...*] [**-exclude-policy** *p1,...*] [**-capacity**] [**-dryrun**] [**-limit** *num*] [**-notify** *level*] [**-noheader**] [**-debug**] [**-help**] [**-policy** *name*]

DESCRIPTION

The **fsactivevault**(1) command calls **vsmove**(l) to automatically vault qualifying media. Media are vaulted when the percentage of the used capacity of the Storage Manager license meets or exceeds the **ACTIVEVAULT_HIGH_USE** sysparm value. Media will continue to be vaulted by **fsactivevault**(1) until the used capacity is below the **ACTIVEVAULT_LOW_USE** sysparm threshold. By default, only media that meet or exceed the **ACTIVEVAULT_FULL_PERCENT** sysparm value will be vaulted. Active Vault policies may be scheduled with the **fsschedule**(1) command.

The administrator must manually complete the **vsmove**(l) operation through the Library Operator Interface (LOI) in the StorNext GUI. When used in conjunction with the Quantum Scalar library Active Vault feature and SNAPI, the media will automatically move to the Active Vault after completing the **vsmove**(l) operation through the LOI. Otherwise, the media will be in the library mailbox and need to be manually removed.

OPTIONS

-archive *a1,...*

Media to be vaulted are selected from this list of archives. Multiple archives may be specified by either listing them as a comma separated list or by specifying multiple **-archive** options. At least one archive name is required unless **-report** is also specified.

-vault *dest*

The destination archive where to vault media. The **-vault** option is required unless **-report** is also specified.

-copy *c1,...*

A list of copy numbers to query on. If no **-copy** option is specified, then media belonging to any copy may be moved to the vault. Multiple copies may be specified by either listing them as a comma separated list or by specifying multiple **-copy** options.

-used *size*

Select only media that have used *size* capacity. *size* is in bytes by default, but a suffix of **B**(ytes), **K**(ibibytes), **M**(ebibytes), **G**(ibibytes) or **T**(ebibytes) may be used to specify capacity.

-remaining *size*

Select only media that have *size* capacity remaining. *size* is in bytes by default, but a suffix of **B**(ytes), **K**(ibibytes), **M**(ebibytes), **G**(ibibytes) or **T**(ebibytes) may be used to specify capacity.

-highmark *pct*

Override the **ACTIVEVAULT_HIGH_USE** sysparm value to start vaulting if the used capacity of the Storage Manager license is at or above the *pct* percent.

-lowmark *pct*

Override the **ACTIVEVAULT_LOW_USE** sysparm value to stop vaulting if the used capacity of the Storage Manager license is below the *pct* percent.

-fullpct *pct*

Override the **ACTIVEVAULT_FULL_PERCENT** sysparm value to consider vaulting media that is at or above the *pct* percent.

-age *age*

Vault media according to age. *age* by default is in seconds, but a time unit may also be provided to specify **seconds**, **days**, **weeks**, **months** or **years**. A specific date may also be specified with the **YYYY:MM:DD:hh:mm:ss** format.

-sort *column*
Sort results based according to *column* where *column* can be **age**, **id**, **full**, **remaining** or **used**.

age will sort by last access time of the media.

id will sort by media ID.

full will sort by the percentage full of the media. This is the default behavior if no **-sort** option is specified.

remaining will sort by space remaining on the media.

used will sort by the amount of space used on the media.

-migrate
Select from media in the MIGRATE media class.

-nomigrate
Ignore media in the MIGRATE media class.

-include-policy *p1***,...**
Select media belonging to the list of policy classes. If this option is used, only media belonging to the list will be selected. Multiple policies may be specified either by using a comma separated list or by including multiple **-include-policy** options.

-exclude-policy *p1***,...**
Excludes media belonging to the listed policy classes. The **_adic_backup** policy is excluded unless explicitly included by **-include-policy**. Multiple policies may be specified either by using a comma separated list or by including multiple **-exclude-policy** options.

-limit *num*
Limit the number of vaulted media to *num*.

-dryrun
Show what media would be vaulted according to the selection criteria without actually calling the **vsmove**(l) command to vault the media.

-notify *level*
Set the email notification level for active vault policy admin alerts where *level* is either **none**, **error**, **warn**, or **info**.

none will suppress all email notifications.

error will only send notifications when an error occurs, such as database errors or licensing errors.

warn will send notifications for warnings, such as being unable to vault enough media to satisfy the low water mark.

info will cause email notifications to be sent indicating that the active vault policy completed successfully. The default notification level is **warn**.

-debug Enable extra debug output.

-help Display help.

-policy *policyname*
The name of the Active Vault policy to use for email notifications.

REPORTING OPTIONS

-report Generate a media report based upon the selection criteria. This does not call the **vsmove**(l) command.

-pending
Select media where a vaulting operation is pending.

-nopending
Select media where no vaulting operation is pending.

-capacity
Report the current licensed capacity and total available licensed capacity.

-noheader
Do not display header or result count in the media report.

ENVIRONMENT

The following parameters are configurable in the *fs_sysparm* file.

ACTIVEVAULT_HIGH_USE
Percentage of used licensed capacity at which to begin the Active Vault policy.

ACTIVEVAULT_LOW_USE
Percentage of used licensed capacity at which to stop the Active Vault policy.

ACTIVEVAULT_FULL_PERCENT
Percentage value used to check against to determine if a medium is full enough for vaulting consideration by Active Vault.

EXAMPLES

Example 1.
Vault all copy #2 media in the i6k archive to the vault01 archive that are in the MIGRATE class.

```
fsactivevault -a i6k -v vault01 -c 2 -migrate
```

Example 2.
Create a scheduled policy that runs everyday at 2 am that will vault media that are at least 500 MiB in size and are older than 2 months. Limit the number of vaulted media to 10. Media are to be vaulted in order of access time from least recently accessed to most recently accessed. The default high and low thresholds and full percent apply.

```
fsschedule -a -n mypolicy -f activevault -p daily -t 0200 -- \
   -a i6k -v vault01 -age 1month -used 500M -sort age -limit 10
```

Example 3.
The capacity for the Storage Manager is at or near the licensed capacity. To reclaim licensed capacity an administrator might decide to vault media belonging to a policy class named "old_data" that has not been accessed in the last 10 days that is also over 1 GiB in use. The selection criteria is sorted by the size to reduce the amount of vaulting that may be needed. The fullpct value is reduced to 1% to override the default ACTIVEVAULT_FULL_PERCENT sysparm to help make sure that there are qualified media.

```
fsactivevault -a i6k -v vault01 -include-policy old_date -used 1G \
    -age 10days -sort size -fullpct 1
```

Example 4.
Query media to determine which media will be vaulted next, excluding all media belonging to policy class "important".

```
fsactivevault -report -exclude-policy important
```

Example 5.
Compression is enabled on drives and the data compresses at a high data compression ratio. To avoid vaulting media with a large amount of space remaining, the administrator sets the remaining size constraint and sets the fullpct to 0.

```
fsactivevault -a i6k -v vault01 -remaining 500M -fullpct 0
```

Example 6.

Vault media that have not been accessed since a specific date of "Jan 15 12:30:00 2012".

```
fsactivevault -a i6k -v vault01 -age 2012:01:15:12:30:00
```

EXIT STATUS

0	Success completion.
1	There was an error processing the arguments.
2	The vaulting license is invalid.
11	There were not any media detected that matched the policy criteria.
12	The Active Vault policy was unable to vault enough media to get below the high water mark.
13	The Active Vault policy was unable to vault enough media to get below the low water mark.
255	An internal fatal error prematurely terminated **fsactivevault**(1).

NOTE

By default the fsactivevault utility will use the licensed capacity as defined in the license.dat file OR if no license.dat file the licensed capacity will be the maximum value. The **ACTIVEVAULT_MAX_CAPACITY** sysparm can be used to override this capacity.

FILES

/usr/adic/TSM/config/fs_sysparm.README

SEE ALSO

vsmove(l), **fsschedule**(1), **fsschedlock**(1)

NAME

fsaddclass − Create and define a new policy class.

SYNOPSIS

fsaddclass *class*
[**-F** *type*] [**-s** *softlimit*] [**-h** *hardlimit*]
[**-x** *maxcopies*] [**-d** *defaultcopies*] [**-t** *mediatype*]
[**-v** *drivepool*] [**-m** *minstoretime*] [**-c** *mintrunctime*] [**-k** *maxversions*]
[**-f i**|**p**] [**-W** *clnvertime*] [**-r c**|**s**] [**-a** *affinity*...**[**-i *minreloctime***]]**
[**-G y**|**n**] [**-V y**|**n**] [**-D y**|**n**] [**-K y**|**n**] [**-H y**|**n**] [**-p yes**|**no**] [**-z** *minsetsize* -g *maxsetage***]**
[**-S** *stubsize*] [**-R** *affinity*] [**-T ANTF**|**LTFS**] [**-L** *drivelimit*]
[**-A y**|**n**] [**-e** *encryptiontype*] [**-M** *mkeyname*]
[**-O** *retrieveorder*] [**-U** *putstreams*] [**-I** *getstreams*]
[**-Z** *multistreamsize*] [**-Q y**|**n**]

DESCRIPTION

The **fsaddclass**(1) command creates a new policy class definition. For each of the optional arguments that are not entered, the Tertiary Manager software will substitute default values. Most default values can be modified in a system parameter file.

This command accepts upper case or lower case input, but class names will always be converted to lower case.

OPTIONS

class Policy class. If the policy class already exists, the command is rejected. A policy class is a maximum of 16 alphanumeric characters. The special "-", ".", and "_" characters are also permitted.

-a *affinity*...
A space-separated list of disk affinities that the files in this policy class will traverse throughout their life cycle. Valid entries include any of the affinities already configured in the managed file systems or the word **none**. The first affinity in this list will be considered the default disk affinity for this policy class, the affinity in which files initially will be created. When they become eligible candidates for relocation, the files will be moved to the next disk affinity in the list. The word **none** may be specified in place of the affinity list to indicate no automatic relocation is to occur for this policy class. NOTE: Presently a maximum of two affinities are supported, including the default disk affinity. If the **-a** option is not used, no automatic relocation will occur for this policy class.

-R *affinity*
The affinity to retrieve a truncated file to. This will override the default affinity.

-d *defaultcopies*
The total number of copies that will be stored (including the primary copy) for each file in this policy class. The *defaultcopies* option can be set equal to, but not exceeding, the *maxcopies* setting. It cannot be set to less than one. If the **-d** option is not used, the default number of copies will be specified by the system parameter CLASS_DEFAULT_COPIES.

-f i|**p** The file retention policy for the policy class. The files can be truncated immediately (**i**) or at policy application time (**p**) once all file copies are stored on a medium. If the **-f** option is not used, the file retention policy will be specified by the system parameter CLASS_FILE_CLEANUP.

-h *hardlimit*
The maximum number of media that are allowed in this policy class. When the hard limit is reached, a warning message is sent to the *syslog* and to the user specified as the e-mail contact for this policy class. Files can still be stored in the policy class, as long as there is room on the media that have already been used to store files in that policy class. If the **-h** option is not used, the *hardlimit* will be specified by the system parameter CLASS_HARDLIMIT.

-c *mintrunctime*

The minimum time that a stored file must reside unaccessed on disk before being considered a candidate for truncation (the clearing of disk blocks). A file will not have its disk blocks truncated (by a truncation policy) until it has remained unaccessed on disk for this amount of time. After that time, a truncation policy will consider the file a valid candidate for truncation, but it may or may not actually be truncated. That will depend on the current file system fill level and the file system configuration parameters. NOTE: An "emergency" truncation policy ignores this time. If the **-c** option is not used, the default *mintrunctime* will be specified by the system parameter CLASS_TRUNCTIME. The minimum value allowed for this time is 5 minutes. See SETTING CLASS TIMES below for more info on time format and usage.

-W *clnvertime*

Each class can have an *clnvertime* interval for the cleanup of old versions of files within the class by the **fsclean**(1) command. This is in addition to a system-wide cleanup interval that is described in the **fsschedule**(1) man page. The interval is up to five digits plus a single-letter interval factor for days (**d**), weeks (**w**), months (**m**), or years (**y**). Setting the interval to zero turns off per-class cleanup and leaves the files in the class subject to the system-wide cleanup interval.

-i *minreloctime*

The minimum time that a file must reside unaccessed on disk before being considered a candidate for relocation (the moving of data blocks from one disk affinity to another). A file will not have its data blocks relocated (by a relocation policy) until it has remained unaccessed on disk for this amount of time. After that time, a relocation policy will consider the file a valid candidate for relocation. The file may or may not actually be relocated at that time, depending on the current file system fill level and the file system configuration parameters. NOTE: An "emergency" relocation policy ignores this time. The **-a** option is required to use this option. If the **-i** option is not used, the default *minreloctime* will be specified by the system parameter CLASS_TIERTIME. The minimum value allowed for this time is 5 minutes. See SETTING CLASS TIMES below for more info on time format and usage.

-m *minstoretime*

The minimum time that a file must reside unmodified on disk before being considered a candidate for storage on media. A file will not be stored (by a store policy) until it has remained unmodified on disk for this amount of time. After that time, the next policy run will attempt to store the file. If the **-m** option is not used, the default *minstoretime* will be specified by the system parameter CLASS_MINTIME. The minimum value allowed for this time is 1 minute. See SETTING CLASS TIMES below for more info on time format and usage.

-S *stubsize*

The truncation stub size (in kilobytes). This value is used to determine the number of bytes to leave on disk when files are truncated. This value will be the minimum number of bytes left on disk (the value will be rounded up to a multiple of the file system block size). This value is not used when a stub size has been explicitly set for a file with **fschfiat**(1). If the **-S** option is not used, the default will be specified by the system parameter CLASS_TRUNCSIZE.

-r c|**s** Media classification after cleanup policy has been performed. When all files are deleted from a medium, the medium can revert back to the owning policy class blank pool (**c**) or to the system blank pool (**s**). If the **-r** option is not used, the default media classification will be specified by the system parameter CLASS_MEDIA_CLEANUP.

-s *softlimit*

The warning limit for the number of media that can be allocated in this policy class. When the soft limit is reached, a warning message sent to the *syslog* and to the user specified as the e-mail contact for this policy class. The value of *softlimit* must be less than or equal to the value of *hardlimit*. If the **-s** option is not used, the default *softlimit* will be specified by the system parameter CLASS_SOFTLIMIT.

-t *mediatype*

Defines the type of media to be used for the policy class. Depending on the type of platform used, the following media types are supported by Tertiary Manager software:

AWS
AZURE
LATTUS
GOOGLE
GOOGLES3
S3
S3COMPAT
LTO
LTOW
3592
T10K
SDISK

If this policy class is being configured to only support disk-to-disk relocation, then **DISK** is also valid, but the **-a** option must also be specified.

The **fsstore**(1) command can override this parameter with the **-t** option. If the **-t** option is not used, the default media will be specified by the system parameter CLASS_DEF_MEDIA_TYPE.

-v *drivepool*

The Media Manager drive pool group used to store or retrieve data for this policy class. This drive pool name must be defined within the Media Manager software before any media operations can occur for this policy class. The special "_" character is permitted to identify the drive pool group. If the **-v** option is not used, the default drive pool group will be specified by the system parameter CLASS_DRIVEPOOL.

-x *maxcopies*

The maximum number of copies (including the primary copy) that are allowed for each file in this policy class. The *maxcopies* value cannot be less than one. NOTE: If the copy setting for a particular file is adjusted using the **fschfiat**(1) command, it cannot exceed the value defined by the *maxcopies* setting. If the **-x** option is not used, the maximum number of copies will be specified by the system parameter CLASS_MAX_COPIES.

-k *maxversions*

This is the maximum number of inactive versions to keep for a file (the current version is active, all others are inactive). When a stored file is modified, the version number is immediately incremented for the file. Old versions are kept around until the **fsclean**(1) command is used to clean those versions from Tertiary Manager media. At the time a new version is stored, the oldest version will be deleted if maxversions has been reached. The **fsclean**(1) command will not be required to clean that version. If maxversions is defined as 0, only the active version will be retained and if removed from the file system, it will still be recoverable unless the **-D** option is set on the file. If the **-k** option is not used, the default max versions to keep will be specified by the system parameter CLASS_MAX_VERSIONS.

-p yes|no

This option decides if we allow the policy engine to automatically store files for this policy class. If the **-p** option is not used, the default will be specified by the system parameter CLASS_AUTOSTORE.

-z *minsetsize*

The minimum set size of the policy class. It will be specified in the form of [0-999][MB|GB]. When this option is specified with a non-zero value, the store candidates in the policy class have to add up to *minsetsize* before any of them can be stored by the policy engine. This option works in conjunction with **-g** option. Specifying zero value for this option will disable the check, which requires **-g** be set to zero as well. If the **-z** option is not used, the default will be specified by the sys-

tem parameter CLASS_MINSETSIZE, which is always in MB and does not require the unit suffix.

-g *maxsetage*

Candidacy time limit (in hours) of the policy class. This option works in conjunction with **-z** option so that files will not sit forever on the candidate list because the *minsetsize* has not been reached. As soon as any file on the store candidate list in the policy class is *maxsetage* old, all of the files should be stored. Specifying zero will disable the check, which requires **-z** be set to zero as well. The maximum value for this option is 720 hours, which equals to 30 days. If the **-g** option is not used, the default will be specified by the system parameter CLASS_MAXSETAGE.

-T ANTF|LTFS

Specifies the media format type that will be used when formatting or selecting media. **ANTF** is the Quantum internal tape format. **LTFS** is the Linear Tape File System specification tape format. A value of **ANTF** will format media with a single partition containing ANTF volume labels. This data partition will store StorNext file data. A value of **LTFS** will format media with two partitions containing LTFS volume labels. The index partition will store LTFS metadata and the data partition will store StorNext file data. The **LTFS** media format type is only applicable to LTO media generation 5 and beyond (e.g. LTO-5, LTO-6, etc). Any attempt to assign **LTFS** to non-LTO media types will result in an error. The backup policy class _adic_backup does not support a media format type of **LTFS**. Any attempt to assign **LTFS** to _adic_backup will result in an error.

If the **-T** option is not specified, the default will be determined as follows:

A media type of Object Storage will default to **NONE**.
A media type that does not support **LTFS** will default to **ANTF**.
A media type that supports **LTFS** will default to the system parameter CLASS_MEDIA_FORMAT.

-L *drivelimit*

The maximum number of drives to use when the policy is run. Specifying **none** will disable the drive limit. If the **-L** option is not specified, all available drives will be used.

-O *retrieveorder*

A comma-separated list of copy numbers specifying the order in which copies will be selected when retrieving files. Specifying **none** will re-enable the default retrieve order. If the **-O** option is not specified, the default retrieve order will be used.

-G y|n Generate and maintain a checksum for each stored file. If this option is enabled, a checksum will be generated as each file is written (stored) to the associated protection tier. The checksum will be retained for use in the checksum-validation phase of subsequent retrieve operations. This option can be overridden by the FS_GENERATE_CHECKSUM parameter in *fs_sysparm*. See *fs_sysparm.README* file for details. The **fsfileinfo**(1) command can be used to determine if a file has a corresponding retained checksum value.

-V y|n Verify the checksum of each retrieved file. If this option is enabled, a checksum will be generated as each file is read (retrieved) from the associated protection tier. The generated checksum will be compared to the corresponding (created when the file was stored) retained value. The retrieve will fail and a RAS ticket will be opened if the checksum does not match. No compare will occur if no retained value exists (checksum generation was not enabled when file was stored). This option can be overridden by the FS_VALIDATE_CHECKSUM parameter in *fs_sysparm*. See *fs_sysparm.README* file for details.

-D y|n Remove database information when a file is removed. If this option is enabled, then when a file is removed from the file system, the corresponding database information indicating where the file was stored will also be removed, and the file will NOT be recoverable through **fsrecover**(1). If this option is disabled, then the database entries will be retained and the file is recoverable through **fsrecover**(1).

-K y|n This indicates that default retry logic will not be peformed for retrieve failures. It can be useful for applications that have their own specialized retry logic based on the type of the

data being stored (such as erasure coded data). **-H y|n** Enable (**y**) or disable (**n**) Dynamic Library Pooling for this policy class. Default value is **n**. If enabled, the Tertiary Manager will direct store requests to specific drive pools on a rotating basis. DLP drive pool sets are configurable for each copy number via the DLP_COPY*_DRIVEPOOL_SET sysparms. See *fs_sysparm.README* for details. If disabled, no drive pool rotation will be performed, and the default drive pool will be used for store requests.

-Q y|n When storing to an object storage system, enabling this option will cause metadata, including the copy number, last modification time, file offset, file key, path, segment number, and version, to be stored along with the object. File metadata can be added or updated using **fsobjmeta**(1). Note, the path will be URL encoded if it contains non-ASCII characters.

-A y|n Enable Alternate Store Location support. If this option is enabled, files created under this policy class will be eligible for an additional remote copy to be made to the configured Alternate Store Location. Note: This is a licensed feature.

-e *encryptiontype*

The encryption algorithm to be applied to the content of files stored into an Object Storage system. This option is ignored for all other media types. Currently, the following encryption types are supported:

none	No encryption
SSE_S3	Server-side AES256 encryption with S3-managed keys
SSE_KMS	Server-side AES256 encryption using the AWS Key Management Service (KMS)

For the **SSE_KMS** encryption type, the default customer master key (CMK) can be used for encryption or a CMK can be created using the AWS Key Management Service and specified with the **-M** option.

If this option is not specified, the encryption type will be set to **none**.

-M *mkeyname*

The Master Key name that has been created using the AWS Key Management Service and requested for server-side encryption.

-U *putstreams*

Defines the number of streams to use for stores when a file in the request is equal to or larger than the size defined with option **-Z**. The value must be in the range of 0 and 64. The default value is 0 if not specified.

-I *getstreams*

Defines the number of streams to use for retrieves when a file in the request is equal to or larger than the size defined with option **-Z**. The value must be in the range of 0 and 64. The default value is 0 if not specified.

-Z *multistreamsize*

Defines the minimum file size for using multiple streams. The value of *multistreamsize* cannot be less than 20MiB. It can include one of the following suffixes:

B	bytes
KB	kilobytes (1000)
KiB	kibibytes (1024)
MB	megabytes (1000^2)
MiB	mebibytes (1024^2)
GB	gigabytes (1000^3)
GiB	gibibytes (1024^3)

TB terabytes (1000^4)
TiB tebibytes (1024^4)

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

SETTING CLASS TIMES

The policy time values associated with a class: *minstoretime*, *mintrunctime* and *minreloctime* can all be set in units of minutes, hours or days. To specify minutes put an '**m**' suffix on the value, to specify hours put an '**h**' suffix on the value and for days use a '**d**' suffix. Note that if the unit suffix is not specified the *minstoretime* value defaults to units of minutes while the others default to units of days. Some valid examples for policy times:

15m - 15 minutes
3h - 3 hours
7d - 7 days
10 - 10 minutes for *minstoretime* and 10 days for the other times

Store policy commands are kicked off automatically every minute or so as long as files are being created. Therefore setting a value for *minstoretime* to just a few minutes will usually result in a store policy command starting the store operation for the files in a class within the time requested. Note however if files are being created slowly the software may wait up to 5 minutes before kicking off a store policy in an attempt to get a larger number of files to store at one time. Also be aware that other factors like system load, media availability etc can affect the time the stores will actually occur.

Truncation and relocation policies are only run automatically once a day at midnight. (There are also policies that run as file system fill levels warrant but the time when these occur is not scheduled. See the **filesystems**(4) man page for more info on the space based policies.) If there is a real need to have file data truncated at a specific time after last access then two steps are necessary, the setting of the class time and the scheduling of the policy commands. As an example let us assume you want to truncate class data for class1 after being unaccessed after one hour. The first step would be to set the *mintrunctime* to **1h** (or **60m**). Next you must schedule via cron a truncation policy to run every half hour. (Note here that even with policies running every half hour; a file may wait up until the time between policies beyond the trunctime before truncation occurs. For example if policies run on the half hour, a file is created at 01:00:01am, it will not be truncated until 2:30am since at 2:00am it is 1 second short of being an hour old.)

When setting up the policy cron job it is probably easiest to set up a simple shell script to wrap the policy so that the processing environment is set up correctly. For example set up a script under TSM: */usr/adic/TSM/util/truncPolicy* The contents of the script may look like:

```
#!/bin/sh
#
. /usr/adic/.profile
/usr/adic/TSM/exec/fspolicy -t -c class1 -o 0
```

Note the last argument to the policy command '**-o 0**'. This tells the policy to keep truncating files until it runs out of candidates or the file system reaches 0 percent full. If you look at the **filesystems**(4) man page it indicates the automatic nightly policy only truncates to the Min Use percentage and then quits even if more valid candidates are present. If the desire is to truncate all candidates then the '**-o 0**' is needed. Truncating all files for a class should be done carefully as there will be an expense to retrieving those files back if needed.

Only one truncation policy is allowed to run at a time. This is due to potential conflicts in candidate management between policies and also for performance reasons. If it is desired to run multiple policies 'at the same time' then just put multiple policies in the truncate script and have them run sequentially. Be sure and not place an ampersand after the commands or some will fail because they are locked out by the currently running policy. Also be sure when setting up the cron jobs not to have the scheduled scripts run too close together. The User's Guide contains more info on the scheduling of truncation policies. An example of a script with multiple scheduled policies would be:

```
#!/bin/sh
#
. /usr/adic/.profile
/usr/adic/TSM/exec/fspolicy -t -c class1 -o 0
/usr/adic/TSM/exec/fspolicy -t -c class2 -o 0
/usr/adic/TSM/exec/fspolicy -t -c class3 -o 0
```

The next step is to create the actual cron entry. Run **crontab -e** and set the entry to look something like this:

```
00,30 * * * * /usr/adic/TSM/util/truncPolicy
```

One last thing to note on scheduling 'extra' truncation or relocation policies: there is an expense to running these commands as they get and check their candidate lists, even if no files are actually truncated or relocated. This is especially true for sites where millions of files are resident on disk at one time. See the User's Guide for more recommendation info on the scheduling of these policies.

SEE ALSO

filesystems(4), **fsmodclass**(1), **fsrmclass**(1), **fsclassinfo**(1), **fschfiat**(1), **fsstore**(1), **fsrelocate**(1), **fsversion**(1), **fsclean**(1), **fsschedule**(1), **fsfileinfo**(1), **fspolicy**(1), **fskey**(1), **fsobjmeta**(1),

NAME

fsaddrelation − Add a directory-to-policy class association.

SYNOPSIS

fsaddrelation [**-F** *type*] **-c** *class directory*

DESCRIPTION

The **fsaddrelation**(1) command associates a directory with a policy class. Classes of data are delineated by assigning the directories on disk to policy classes. The directory specified in *directory* must not be superior to any directory that already has a directory-to-policy class relationship, nor may it be subordinate to any directory that has a relationship to a different class. The policy class relation point cannot be a directory in the root file system, and it is recommended that the command be executed before adding any files to the directory.

If the command must be run on a populated directory, it is recommended that the **fsaddrelation**(1) command be performed when the system is comparatively idle, understanding that the command is likely to take considerable time to complete. If the command is interrupted while running, it can leave content under that directory as unmanaged. If the command is rerun after encountering a failure on a populated directory, it will attempt to process any files and directories that were not processed during the failed execution.

Part of the processing done on a populated directory is to validate the hard links under that directory for multi-link files. If there are links under that directory, to files that are not under that directory, then the add relation processing will fail. This is due to a restriction in StorNext which does not allow hard links across policy class boundaries. If the addrelation was allowed to proceed then the result would be a file with a link in a managed directory and another in a non-managed directory. Also note that this restriction does not apply to symbolic links.

Adding a directory relationship to a policy class causes the directory to become a migration directory under Tertiary Manager software control. A migration directory is one in which the files stored by client users are Tertiary Manager controlled media. The relationship can be removed by using the **fsrmrelation**(1) command after all files subordinate to the policy class relation point are removed.

fsaddrelation(1) will fail if *directory* has an affinity. If *directory* has an affinity associated with it, first use the **cvaffinity**(1) command to remove its affinity. Upon successful completion of **fsaddrelation**(1), if the *class* has any affinities, the directory will have its affinity set to the first affinity in the class affinity list.

Be aware that if *directory* has subordinate directories already in existence when this command is run, the subordinate directories will retain their current affinity association. In that case, the creation of any new files in those directories may result in allocations to unexpected stripe groups. Therefore, it is again strongly recommended that **fsaddrelation**(1) be executed before adding any files or subdirectories to *directory*.

When the first relation is added to a directory in a file system, that file system becomes a managed file system. An entry for the file system is added to the file system configuration file: $TM_DIR/config/filesystems. The entry is added with system defaults for values. If it is desired to change the defaults, edit the file and make the desired updates. The file itself has a description of the configuration values. When the last relation is removed from a file system, the entry is removed from the configuration file.

The identifier specified in *class* determines how media will be selected for use in migrating files in the specified directory. Only media containing files associated with the same policy class, or system blanks, will be selected.

The first file written to a medium taken from the system blank pool will determine which policy class the medium is associated with.

This command accepts upper case or lower case input.

OPTIONS

directory

Path name of the directory to be associated with the policy class. The directory must already exist and the name must be less than 256 characters long. The entire path name need not be entered. If preceded by a slash (/) in the command definition, the full path name (starting from the root directory) is required as input to the command. Otherwise, the Tertiary Manager command expands the

directory name using the current working directory as the parent.

-c *class* Policy class to be associated with the directory. A policy class name can be a maximum of 16 alphanumeric characters. The special "-", ".", and "_" characters are also permitted.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

SEE ALSO

fsrmrelation(1), **fsclassinfo**(1), **fsdirclass**(1), **filesystems**(4)

NAME

fsaffdf − Generate a report on disk space usage for a StorNext file sytem, including a breakdown of disk usage per stripe group and affinities.

SYNOPSIS

fsaffdf *mount_point...*

DESCRIPTION

The **fsaffdf**(1) command displays disk space usage information for the StorNext file systems located at the specified mount points. For each file system, the report also includes a breakdown of the disk space used by each data stripe group and the associated affinities.

Stripe group information is not displayed for metadata/journal or offline/non-writable stripe groups.

OPTIONS

mount_point...
The StorNext file system mount points

REPORT STATUS

Overall Filesystem Information

FileSystem
File system name

BlkSize File system block size (in bytes)

TotalBlks
Total blocks

FreeBlks
Number of available blocks

UsedSpc
Used space (percentage)

MountPoint
File system mount point

Stripe Group Information

StripeGroup
Stripe group name

TotalBlks
Total blocks in the stripe group

FreeBlks
Number of available blocks in the stripe group

UsedSpc
Used space (percentage) in the stripe group

Affinities
List of affinities associated with the stripe group

EXIT STATUS

0 A report was successfully generated for the specified mount points.

255 A failure occurred.

NAME

fsaffinity − Add or modify affinities in the Tertiary Manager database.

SYNOPSIS

fsaffinity -a *affinity* [**-f**]

fsaffinity -r *affinity***new_name** [**-f**]

fsaffinity -d [**-f**]

fsaffinity -l

DESCRIPTION

The **fsaffinity**(1) command populates the Tertiary Manager database with affinities that have been previously configured in managed file systems. This command may also be used to rename, delete, or list existing affinity entries in the database.

OPTIONS

-a *affinity*
Adds *affinity* to the Tertiary Manager database.

-r *affinity new_name*
Renames *affinity* to *new_name* in the Tertiary Manager database.

-d Deletes all affinity entries from the Tertiary Manager database.

-l Lists all affinities currently in the Tertiary Manager database.

-f Forces the addition, renaming, or deletion of affinities, bypassing file system affinity verification.

USAGE

Use this command after adding or modifying affinities in your managed file system configurations. This command will make the Tertiary Manager aware of the affinities in use by the managed file systems.

EXIT STATUS

0 Successful completion.

1 An error occurred.

SEE ALSO

fsaddclass(1), **fsmodclass**(1), **fsclassinfo**(1), **fsfileinfo**(1)

NAME

fsaltnode − Add, modify or delete alternate retrieval location information in the Tertiary Manager database.

SYNOPSIS

fsaltnode -a|**-m**|**-d -n** *remote_node_name*

fsaltnode -a|**-m -p** *local_filesys* **-r** *remote_filesys* [**-n** *remote_node_name*]

fsaltnode -d -p *local_filesys*

fsaltnode -l

DESCRIPTION

The **fsaltnode**(1) command is used to add, modify, or delete alternate retrieval location information in the Tertiary Manager database. The alternate retrieval location is used when a truncated file on a managed filesystem cannot be retrieved from the standard file copies. For example, unavailable tape drives may prevent a retrieval with the standard file copies. If an alternate (remote) node and an alternate (remote) filesystem are specified for the local managed filesystem, the file will be retrieved from that remote filesystem on the remote node.

A single alternate retrieval node name may be added, modified, or deleted by specifying the -n option. This remote node name applies to all filesystems that also have an alternate retrieval filesystem name specified. The existence of the alternate node is not verified when this command is executed. Note also that the contents and version of the file on the alternate node are not verified by the alternate retrieval mechanism. The application software is required to keep the standard filesystem and the alternate retrieval location in sync.

OPTIONS

-a -n *remote_node_name*
: Adds *remote_node_name* to the Tertiary Manager database as the alternate node to be used for all alternate location retrievals.

-m -n *remote_node_name*
: Modifies the Tertiary Manager database to specify *remote_node_name* as the alternate node to be used for all alternate location retrievals.

-d -n *remote_node_name*
: Deletes *remote_node_name* as the Tertiary Manager database as the alternate node to be used for all alternate location retrievals.

-a -p *local_filesys* **-r** *remote_filesys*
: Adds *remote_filesys* to the Tertiary Manager database as the remote filesystem to be used for alternate node retrievals for local filesystem *local_filesys*.

-m -p *local_filesys* **-r** *remote_filesys*
: Modifies the Tertiary Manager database making *remote_filesys* the remote filesystem to be used for alternate node retrievals for local filesystem *local_filesys*.

-d -p *local_filesys*
: Deletes *remote_filesys* as the the remote filesystem to be used for alternate node retrievals for local filesystem *local_filesys*.

-l Lists all filesystems having alternate retrieval location specified.

EXIT STATUS

0 Successful completion.

1 An error occurred.

EXAMPLES

Define an alternate retrieval location node name rabbit as the alternate node for all filesystems:

```
fsaltnode -a -n rabbit
```

Modify the existing alternate retrieval location node name to be squirrel for all filesystems:

```
fsaltnode -m -n squirrel
```

Define the alternate retrieval path for filesystem /stornext/snfs1 to be /backups/dir01. This can only be done if no alternate retrieval path is currently specified for /stornext/snfs1:

```
fsaltnode -a -p /stornext/snfs1 -r /backups/dir01
```

Modify the alternate retrieval path for filesystem /stornext/snfs1 to be /backups/apples:

```
fsaltnode -m -p /stornext/snfs1 -r /backups/apples
```

NAME

fsautoconfig − Automatically configure devices in the Quantum storage subsystem.

SYNOPSIS

fsautoconfig -a

fsautoconfig [**-a**] [**-n** *alias*] **-d** *scsiPath*

fsautoconfig -h

DESCRIPTION

The **fsautoconfig**(1) command gives the ability to automatically configure a single device or all devices not already configured in StorNext. Devices that can be automatically configured include tape drives and media changers / archives, limiting the scope of this automatic configuration to SNSM. At this time only SCSI Archives are supported by this command.

When a non-configured archive is the specified device to auto-configure, the archive along with all media associated with the archive are added. If **-a** is specified then all associated tape drives are added too. When multiple paths are detected for the associated tape drives, the tape drives will be configured across the host buses to ensure that a single host bus doesn't become a limiting factor. By default the name scheme used for the archive is archive<num> unless **-n** is specified. If the archive is already configured or the associated tape drives cannot be determined, the **fsautoconfig**(1) command will fail.

When a non-configured tape drive is specified as the SCSI path, the slot will be detected from the parent archive, and the tape drive will be configured appropriately. The name scheme used is <archive>_dr<num>. If the device is already configured or the slot cannot be determined from the archive, the **fsautoconfig**(1) command will fail.

When **-a** is used without **-d**, all non-configured archives and tape drives will be configured into StorNext. If multiple paths are found to the non-configured devices, load balancing will be used to spread the devices across the host buses from which those devices are being seen for optimal configuration.

A round-robin load balancing scheme is used to spread the devices across the detected host buses. E.g. If there are 2 HBAs that can see the same 4 tape drives, 2 tape drives will be configured through each HBA. The paths can always be changed once configured using the **fsconfig**(1) command.

fsautoconfig(1) requires that the Tertiary Manager, Media Manager, and database software be active.

OPTIONS

-a Configure all non-configured, detected SCSI devices (archives and tape drives) into SNSM.

-d *scsiPath*
The device to configure. If the device is a media changer and **-a** is specified then all drives associated with the changer are also configured.

-n alias This will be used as the device alias instead of the software generated name.

-h Generates the usage report.

SEE ALSO

fsdevice(1), **fsconfig**(1), **fsmedin**(1), **vsarchiveconfig**(l), **vsdrivecfg**(l), **vsaudit**(l), **vsarchiveqry**(l), **vsarcmedclasscreate**(l)

NAME

fsazure − Change or report the tier state of an Azure Block Blob.

SYNOPSIS

fsazure -s | -S *tier* -c *copy filename*

fsazure [**-a** *serverAuth*] -n *container* -o *objectname* -u *URL* [**-C** *certfile* | **-R** *certpath*] -U *username* -P *password* -s | -S *tier*

fsazure -h

DESCRIPTION

The **fsazure** command changes or reports the tier state of an Azure Block Blob. Changing from Archive tier to Hot tier or Cool tier will cause asynchronous rehydration. If the stored data is known to the Tertiary Manager system, a file name can be specified to identify the Azure Block Blob data. Otherwise, the object storage components and security information must be provided. These attributes provide the addressing information and credentials required to access the Azure Block Blob.

Note, this tier feature only applies to Azure Blob storage and General Purpose v2 (GPv2) accounts. General Purpose v1 (GPv1) accounts do not support tiering.

OPTIONS

-u *URL* URL that includes the object identifier of the Azure Block Blob.

-n *container*
Container name.

-o *objectname*
Object name.

-U *username* **-P** *password*
Username and password to access the URL.

-a *serverAuth*
The server authorization setting. Default is 2. The valid options are:

0 0 - none

1 1 - verify peer only

2 2 - verify peer and host

-c *copy* Set or report the tier state for the specified copy.

-C *certfile*
File name to Certificate Authority (CA) certificate(s).

-R *certpath*
Directory path that contains the individual CA certificate files.

-S Change the tier state of the Azure Block Blob to the Hot, Cool or Archive tier. State change is immediate from Cool to Hot or from Hot to Cool. State change from Archive to Cool, or from Archive to Hot will take up to 15 hours.
Note: If the Archive tier is going to be used then the storage class configured with **fsobjcfg**(1) for the related namespace should be set to **azure_block_blob_archive**. If this storage class is not used then objects in the Archive tier are not allowed to be retrieved.

Hot Set object tier to the Hot tier. If object is in Archive tier it will take 15 hours to rehydrate. If the object is in Cool tier, this action is immediate.

Cool Set object tier to the Cool tier. If object is in Archive tier it will take 15 hours to rehydrate. If the object is in Hot tier, this action is immediate.

Archive Set object tier to the Archive tier. This action is immediate.

-s Report the state of the Azure Block Blob object.

filename
The full path name of the file archived to Azure Block Blob storage.

EXIT STATUS

Exit codes for the **fsazure**(1) command are:

0 Command completed successfully.

1 Command failed.

2 Command syntax error.

NOTE

This command applies to Azure Block Blob objects only.

EXAMPLES

Check the tier of file *filename*

```
fsazure -s filename
```

Set the tier of file *filename* to **Archive**

```
fsazure -S Archive filename
```

Rehydrate an Azure Block Blob object by changing its storage tier from Archive to Cool

```
fsazure -S Cool -u https://myaccount.blob.core.windows.net/mycontainer/ \
    AAA987BA-1B48-4E54-86AB-B48AF57381BF  \
    -o AAA987BA-1B48-4E54-86AB-B48AF57381BF -n mycontainer -U myaccount \
    -P MunchedYxUbGcp7V+sf7NBt6CKzIpSundkzpmNdKBlFKf1Y4Ho0pSw7t6j+w==
```

SEE ALSO

fsobjcfg(1)

NAME

fsbulkcreate − Create several test files in a managed file system

SYNOPSIS

fsbulkcreate -d *dir_path* [**-s** *file_size*] [**-n** *num_files*] [**-S**] [**-c** *copies*] [**-v** *versions*] [**-p** *policy_class*]

fsbulkcreate -d *dir_path* **-f** *file_name* [**-n** *num_files*] [**-S**] [**-c** *copies*] [**-v** *versions*]

WARNINGS

This utility should be used with EXTREME CAUTION and only under the guidance of Quantum technical assistance. It is intended to be used by professional services personnel or system testers ONLY for the purpose of demonstrating or testing SNSM on very large file systems.

This utility will stop all storage management software components. Therefore, ensure all users are off all managed file systems before running this utility.

It is strongly recommended that you run a SNSM backup before running this utility. This will enable you to restore your file system to its original state after running this utility.

This utility will drop and re-create critical Tertiary Manager database tables. In case of system failure, database tables could be left in an unusable state. If this occurs, you may restore your system from the previous SNSM backup.

This utility will need exclusive use of the storage management software components and the Tertiary Manager database tables. Therefore, do not attempt to launch more than one instance of this utility at once.

Before running this utility, ensure you have enough disk space in the file system where /usr/adic/mysql/db resides. To create 100 million files with 1 copy and 1 version, you will need approximately 50 GB of disk space. You will also need free disk space on the file system containing *dir_path* sufficient to allocate 100 million inodes.

DESCRIPTION

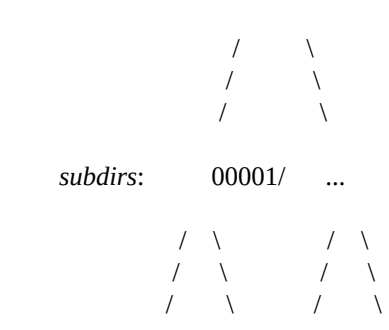
This utility creates a large number of test files on an existing managed file system. Upon successful completion, all new files will appear truncated with the appropriate number of copies and versions stored to tape.

To prevent spending a large amount of time storing every file to tape, all new files will reference the same file data on one tape. This applies to all copies and versions of the new files.

The file that is actually stored to tape is referred to as the "base file". File attributes for the new files are derived from this file. If not provided by the user, this utility will create the base file.

For a balance of performance and namespace readability, new files will be created in a predetermined directory structure. Directly under the directory specified by the user, the utility will create a maximum of 10,000 subdirectories. In each of those subdirectories, a maximum of 10,000 files will be created, for a total maximum of 100,000,000 files.

For example, the full directory structure for 100 million files may appear as follows:



The utility will create the minimum number of subdirectories needed to accommodate the number of files requested by the user. For example, to accommodate 200,000 files, the utility will create only 20 subdirectories.

The subdirectories will be named numerically, starting with "00001" and incrementing up to the last subdirectory created. The files also will be named numerically, starting with "000000001" and incrementing up to the last file created.

If the directory specified by -d *dir_path* does not exist, the utility will create it. If the directory is not managed, the utility will make it into a relation point using the class specified by -p *policy_class*. If the directory is already managed, -p *policy_class* will be ignored.

If the relation point needs to be made, and the class specified under -p *policy_class* does not exist, the utility will create the class. If no class is specified by the user, the default class "bulkcreate" will be used.

The user may specify his own base file with -f *file_name* instead of specifying a class and file size with -p *policy_class* and -s *file_size*. The file name must be a fully qualified path that points to an existing managed file.

OPTIONS

-d *dir_path*
Specifies *dir_path* as the directory to populate. Must be an absolute directory path.

-s *file_size*
Specifies *file_size* as the size in bytes for each new file. (default 1024)

-n *num_files*
Specifies *num_files* as the number of files to create. (default 10000, max 100000000)

-c *copies*
Specifies *copies* as the number of copies to create. (default 1)

-v *versions*
Specifies *versions* as the number of versions to create. (default 1)

-p *policy_class*
Specifies *policy_class* as the data class to create. (default "bulkcreate")

-f *file_name*
Specifies *file_name* as the base file for the new files.

-S Run the command single threaded. By default the command creates a thread for the database operations and performs the file system creates in the main thread. (This option will make the command run more slowly but can aid in debugging issues.)

USAGE

This utility is intended to be used by professional services personnel or system testers ONLY for the purpose of demonstrating or testing SNSM on very large file systems.

This utility will assume the user has already completed the StorNext Configuration Wizard. It will also assume that there are tape media available, that the tape archive is online, and that all SNSM software components are up and responsive. It is assumed that at least one relation point already exists in the file system where bulk create files will be created.

It is strongly recommended that you run a SNSM backup before running this utility. This will enable you to restore your file system to its original state after running this utility.

To clean up the file system, the user always has the option of removing the new files with the *rm* command, but that process can be very time consuming for a large number of files.

Before running this utility, ensure you have enough disk space in the file system where /usr/adic/mysql/db resides. To create 100 million files with 1 copy and 1 version, you will need approximately 50 GB of disk space. Also, the file system that **fsbulkcreate**(1) creates the files in will require enough free space to allocate 100 million inodes.

This utility will reload and index three database tables with pre-existing and newly created records. Because of this, be advised that this utility will take a longer time to run for large pre-existing database tables, specifically the FILEINFO and FILECOMP tables.

BENCHMARKS

Below are the performance results from running this utility on different platforms. The processing times represent the time it took to create 100,000,000 files with 1 copy and 1 version for each file.

machine: Dell Poweredge 1750
operating system: Linux AS
processor(s): 2 x Xeon @ 2.4 GHz
memory: 512 MB RAM
disks: internal SCSI (for managed file system)
processing time: 41:11:40 (HH:MM:SS)

machine: Sun-Fire-V50
operating system: Solaris 9
processor(s): 2 x UltraSPARC IIIi @ 1.28 GHz
memory: 2 GB RAM
disks: internal SCSI (for managed file system)
processing time: 47:48:10 (HH:MM:SS)

EXIT STATUS

0 Successful completion.

1 An error occurred.

FILES

${MYSQL_HOME}
Checked for sufficient space for work area and database.

${MYSQL_HOME}/bulk_create/
Used for work space.

NOTES

For increased performance, this utility will create files in bulk sets. This means that the user may get a few extra files than he actually needs. The same goes for directories. The user may get a few more directories than he actually needs.

The **fsmedinfo -l** command may show duplicate file entries. This is because all file copies will appear to be stored to the same tape. In normal operating conditions, each file copy is stored to a separate tape.

Because file verification remains circumvented for the life of the files, all new files and the base file itself

should all be considered test data only.

This command will not work properly with storage disk or Object Storage media types. It will succeed creating the files, but would fail later on when trying to retrieve the files back from the storage disk or Object Storage media.

SEE ALSO

fsaddclass(1), **fsaddrelation**(1), **fsstore**(1), **fsrmcopy**(1), **fsclassinfo**(1), **fsfileinfo**(1), **fsmedinfo**(1), **dm_info**(1)

NAME

fsbulkload − Utility used by foreign file system migration to load foreign file system information into the database.

DESCRIPTION

The **fsbulkload**(1) is used by foreign file system migration to load foreign file system information into the database.

WARNINGS

Users should not run this command, unless directed to do so by Quantum technical assistance.

NAME

fscancel − Cancels requests.

SYNOPSIS

fscancel [**-F** *type*] *requestID*

DESCRIPTION

The **fscancel**(1) command allows cancellation of media, file, and resource queued requests. The **fsqueue**(1) command displays the pending queues awaiting resources. The request identifier associated with a specific queue is used to cancel that process. Requests waiting for a restore from S3 Glacier or Azure Archive to complete can be removed from the queue but the restore cannot be canceled.

OPTIONS

requestID

The request identifier of the request to be canceled.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsqueue(1), **fsxsd**(1)

NAME

fschdiat − Modify the class attributes of a directory.

SYNOPSIS

fschdiat [**-s e**|**p**] [**-r e**|**p**] [**-t e**|**p**] [**-C y**|**n**] [**-c** *class*] [**-S** *stubsize*] [**-F** *type*] *directory_name...*

fschdiat [**-s e**|**p**] [**-r e**|**p**] [**-t e**|**p**] [**-C y**|**n**] [**-c** *class*] [**-S** *stubsize*] [**-F** *type*] **[-H|-L]**-R *directory*

fschdiat [**-s e**|**p**] [**-r e**|**p**] [**-t e**|**p**] [**-C y**|**n**] [**-c** *class*] [**-S** *stubsize*] [**-F** *type*] **-D** *directory*

fschdiat [**-s e**|**p**] [**-r e**|**p**] [**-t e**|**p**] [**-C y**|**n**] [**-c** *class*] [**-S** *stubsize*] [**-F** *type*] **-B** *batchfilename*

DESCRIPTION

The **fschdiat**(1) command allows modification of the directory attributes pertaining to cleanup and storage policies. It also provides the capability of modifying the directory's class. Policy attributes set for the directory will propagate to any newly created entities created underneath that directory, whether it be another directory or a file. In other words, attributes and class for newly created files and directories are inherited from the parent directory.

Directories which are at the relation point cannot have their class modified by this command. You must use the **fsrmrelation**(1) and **fsaddrelation**(1) to modify the class for a directory which is at the relation point.

OPTIONS

directory_name...

Directory or directories to modify.

-R *directory*

The directory from which to start the recursive operation. All entries from the specified directory and any subdirectories will be processed. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-H This mirrors the functionality provided by the 'find' command and indicates that symbolic links should not be followed. It is only valid with the "**-R**" option. (Ex. ... "**-HR** *directory*" ...)

-L This mirrors the functionality provided by the 'find' command and indicates that symbolic links should be followed. It is only valid with the "**-R**" option. (Ex. ... "**-LR** *directory*" ...) This is the default behavior for all forms of this command which includes cases when **-R** is not specified.

-D *directory*

Only the specified directory and directories in the specified directory will be processed. This is not recursive.

-B *batchfilename*

The batch file contains the list of entries to be processed. Each entry must be listed one per line. Do not enclose the entry in quotes, use escape characters or any character not in the actual name.

-B *batchfilename*

The batch file contains the list of entries to be processed. Each entry must be listed one per line. Do not enclose the entry in quotes, use escape characters or any character not in the actual name. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-s This option indicates how storage policies operate. The *e* argument excludes files from storage when a store policy occurs. The *p* argument stores files by storage policy.

-r This option indicates how relocation policies operate. The *e* argument excludes files from relocation when a relocation policy occurs or if a **fsrelocate**(1) command is issued. The *p* argument relocates files by relocation policy.

-t This option indicates how truncation policies operate. The *e* argument excludes files from truncation when a store and/or cleanup policy application occurs. The *p* argument truncates the file by cleanup policy.

-C This options indicates if the database entries are to be cleaned when the file is removed from the file system. The *y* argument indicates that the database entries will be cleaned and the file will NOT be recoverable by the **fsrecover**(1) command. The *n* argument indicates that the database entries will NOT be cleaned and the file will be recoverable by the **fsrecover**(1) command.

-c This specifies the class that will be associated with the directory.

-S *stubsize*

This indicates the stub size (in kilobytes) and is used to determine the number of bytes to leave on disk when files are truncated. It will be the minimum number of bytes left on disk (the value is rounded up to a multiple of the file system block size). If *class* is specified as the value, then the policy class definitions will be used to determine the stub size.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsaddclass(1), **fsmodclass**(1), **fsstore**(1), **fschfiat**(1), **fsrmrelation**(1), **fsaddrelation**(1), **fsxsd**(1)

NAME

fsCheckAffinities − Verify affinities are configured correctly in StorNext for managed file systems.

SYNOPSIS

fsCheckAffinities [**-ras**]

fsCheckAffinities -report

DESCRIPTION

The **fsCheckAffinities**(1) command will verify that all affinities are configured correctly in StorNext for managed file systems.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

EXIT STATUS

0 If no issues are found.

1 If any issues are found. If the **-ras** option is specified, a RAS Alert will also be generated.

NAME

fsCheckSlotMapping − Verify tape drive slot to device path mapping is configured correctly in SNSM.

SYNOPSIS

fsCheckSlotMapping [**-ras**]

fsCheckSlotMapping -report

DESCRIPTION

The **fsCheckSlotMapping**(1) command will verify that all tape drive slot to device path mappings are configured correctly in SNSM.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

EXIT STATUS

0 If no issues are found.

1 If any issues are found. If the **-ras** option is specified, a RAS Alert will also be generated.

NAME

fsCheckTsmFilesystemsConfig − Verify the Tertiary Manager file systems are identified correctly.

SYNOPSIS

fsCheckTsmFilesystemsConfig [**-ras**]

fsCheckTsmFilesystemsConfig -report

DESCRIPTION

The **fsCheckTsmFilesystemsConfig**(1) command will verify that all Tertiary Manager file systems are identified correctly.

OPTIONS

-ras Used to indicate that a RAS Alert should be generated on any failure.

-report Used to describe the purpose of this command.

EXIT STATUS

0 No issues was found.

1 An issue was found. If the **-ras** option is specified, a RAS Alert will also be generated.

NAME

fschfiat − Modify the class attributes of a file.

SYNOPSIS

fschfiat [**-s e**|**p**] [**-r e**|**p**] [**-t e**|**p**|**i**|**c**] [**-C y**|**n**] [**-A y**|**n**] [**-c** *copies*] [**-a** *class*] [**-S** *stubsize*] [**-F** *type*] *filename...*

fschfiat [**-s e**|**p**] [**-r e**|**p**] [**-t e**|**p**|**i**|**c**] [**-C y**|**n**] [**-A y**|**n**] [**-c** *copies*] [**-a** *class*] [**-S** *stubsize*] [**-F** *type*] **[-H|-L]**-R *directory*

fschfiat [**-s e**|**p**] [**-r e**|**p**] [**-t e**|**p**|**i**|**c**] [**-C y**|**n**] [**-A y**|**n**] [**-c** *copies*] [**-a** *class*] [**-S** *stubsize*] [**-F** *type*] **-D** *directory*

fschfiat [**-s e**|**p**] [**-r e**|**p**] [**-t e**|**p**|**i**|**c**] [**-C y**|**n**] [**-A y**|**n**] [**-c** *copies*] [**-a** *class*] [**-S** *stubsize*] [**-F** *type*] **-B** *batchfilename*

DESCRIPTION

The **fschfiat**(1) command allows modification of the file attributes pertaining to cleanup and storage policies. The file cleanup policy attribute affects how a file is truncated when cleanup policy is applied. A file can be truncated immediately (**-t** *i*), by policy (**-t** *p*), or excluded (**-t** *e*) from file truncation following a store. The file storage policy attribute is used to exclude (**-s** *e*) a file from being stored or consider the file for storage (**-s** *p*). The file relocation policy attribute is used to exclude (**-r** *e*) a file from being relocated or to consider the file for relocation (**-r** *p*). The number of file copies produced on a medium during storage policies can also be modified.

If a file has been excluded from truncation by the **fschfiat**(1) command, the retention (**-f**) option of the **fsstore**(1) command cannot apply to a file. If the file attribute allows truncation and the file is a candidate for truncation following a store, the retention option can be applied. The **-t** *p* option of the **fschfiat**(1) command allows the file attribute to be reset to the default state of having the file data removed from disk when the cleanup policy is applied.

The number of copies specified for the **fschfiat**(1) command cannot exceed the maximum number of copies allowed (*maxcopies*) for the policy class that is associated with the file's parent directory.

The policy attributes can also be modified by changing the policy class of the file. Then the rules for the new class will be applied to the file during storage and cleanup policies.

OPTIONS

filename...

One or more file(s) having the attributes changed.

-R *directory*

The directory from which to start the recursive operation. All entries from the specified directory and any subdirectories will be processed. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-H This mirrors the functionality provided by the 'find' command and indicates that symbolic links should not be followed. It is only valid with the "**-R**" option. (Ex. ... "**-HR** *directory*" ...)

-L This mirrors the functionality provided by the 'find' command and indicates that symbolic links should be followed. It is only valid with the "**-R**" option. (Ex. ... "**-LR** *directory*" ...) This is the default behavior for all forms of this command which includes cases when **-R** is not specified.

-D *directory*

Only entries in the specified directory will be processed. This is not recursive.

-B *batchfilename*

The batch file contains the list of entries to be processed. Each entry must be listed one per line. Do not enclose the entry in quotes, use escape characters or any character not in the actual name. Success for individual items (files, directories, etc) will not be reported. Instead an overall

progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-c *copies*

Number of copies of the file(s) to be stored. The value is the total number of copies, including the primary copy of the file. This number cannot exceed the number of copies defined in the policy class *maxcopies* parameter. If the number of copies stored is less than the number specified by the policy class definition or by the **fschfiat**(1) command, the remainder of the copies are stored when the storage policy is applied. Once a file is located on tape only, its file copy attribute cannot be increased unless that many copies already exist on media. To increase the file copy attribute, you must first retrieve the file.

-s This option indicates how storage policies operate on the file. The *e* argument excludes the file from storage when a store policy occurs. The *p* argument stores the file by storage policy.

-r This option indicates how relocation policies operate on the file. The *e* argument excludes the file from relocation when a relocation policy occurs or if a **fsrelocate**(1) command is issued. The *p* argument relocates the file by relocation policy.

-t This option indicates how truncation policies operate on the file. The *e* argument excludes the file from truncation when a store and/or cleanup policy application occurs. The *i* argument truncates the file immediately when stored to a medium. The *p* argument truncates the file by cleanup policy. The *c* argument temporarily clears the indication that this file met truncate exclusion criteria defined in the excludes.truncate file. If the file is modified and then stored again then the file will marked as excluded provided it still meets the criteria. This indicator is independent of the settings made be the other arguments of this option.

-C This options indicates if the database entries are to be cleaned when the file is removed from the file system. The *y* argument indicates that the database entries will be cleaned and the file will NOT be recoverable by the **fsrecover**(1) command. The *n* argument indicates that the database entries will NOT be cleaned and the file will be recoverable by the **fsrecover**(1) command.

-A y|n Enable Alternate Store Location support. If this option is enabled, the specified files will be eligible for an additional remote copy to be made to the configured Alternate Store Location. Alternate Store Location can only be enabled on files having an associated policy class with Alternate Store Location enabled.

-a This specifies the class that will be associated with the file. The rules for the new class will be applied to the file by the storage and truncation policies. Any copies which exist at the time of the class change will remain and are considered valid copies. If the number of copies changes from the old class to the new, and the copies required for the new class increases, then the file will be updated to indicate that not all copies have been stored.

-S *stubsize*

The truncation stub size (in kilobytes). This value is used to determine the number of bytes to leave on disk when files are truncated. It will be the minimum number of bytes left on disk (the value is rounded up to a multiple of the file system block size). If *class* is specified as the value, then the policy class definitions will be used to determine the stub size.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the

XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsaddclass(1), **fsmodclass**(1), **fsstore**(1), **fschdiat**(1), **exclusions**(4)

NAME

fschmedstate − Change the policy class or state of media.

SYNOPSIS

fschmedstate [**-F** *type*] **-s** *state mediaID...*

fschmedstate [**-F** *type*] **-c** *class mediaID...*

fschmedstate [**-F** *type*] **-b** *mediaID...*

DESCRIPTION

The **fschmedstate**(1) command can be used to change the policy class of blank media or to change certain media state values. The state values that can be changed are those that are displayed in the **fsmedinfo**(1) report under the **Media Status**, **Mark Status**, **Suspect Count**, **Write Protect**, and **Ownership** fields.

Media Status

Media can be set to **Avail** or **Unavail**. The **Unavail** status will prevent further access to the media. This includes storage, retrieval, and media movement requests.

Mark Status

When media fail to format, their mark status is automatically set to **Error**, which will disallow further access to those media. A system administrator can use **fschmedstate**(1) to change the mark status of such media back to **Unmarked** to allow the media to be accessed again. The admin might do this for the purpose of debugging the failure. This command can also be used to reset **Check-Out** and **Export** mark statuses back to **Unmarked**. The **Check-Out** status can be reset as long as the medium is not **Out of FS**.

Suspect Count

A medium is designated as suspect when it incurs an I/O or positioning error. A suspect medium is used, thereafter, as read-only. The system may subsequently increment the suspect count for the medium if another read or positioning error occurs when it is mounted in a drive. The system administrator can use the logs to determine whether an error was caused by the drive or by the medium. If the medium is not at fault, **fschmedstate**(1) can be used to reset the medium's suspect count back to zero to allow write access to the medium.

Write Protect

Media write protection can be set or cleared. Setting media as write-protected will prevent the Tertiary Manager software from choosing those media for file storage. Files can be read from write-protected media, but additional data cannot be written to those media.

Ownership

A medium's ownership must be set to **Imported** in order to process import events generated in the namespace importation. An ownership of **Imported** indicates that the medium is owned by another Storage Manager system and has been imported locally in read-only mode. When media are in this state, the contained files are available for file retrieves, but store operations to these media are not permitted. Any type of cleaning operation that would delete files from the media will also be denied. Reports from **fsmedinfo**(1) will show the media as write-protected as a reminder that they cannot be written to. Changing media from **Imported** to **Owned** should only be done when recommended by Quantum support, as this could create duplicate file keys in the media's LTFS namespace which could break access to the file data on that media.

OPTIONS

mediaID...

One or more 16-character media identifiers. Multiple media identifiers must be separated by spaces. The number of media identifiers that can be entered is limited to 99.

-s *state* Update media to the specified *state*. Valid values:

protect Set write protection. Files can be read, but no data can be written to the media.

unprotect Clear write protection, allowing writes to media. This implicitly sets any **Imported** media to **Owned**.

imported Set media ownership to **Imported**. This implicitly sets media as write-protected.

owned Set media ownership to **Owned**. This implicitly clears the write protection on the media. This should only be done if recommended by Quantum support.

avail Set media status to **Avail** to allow it to be used for storage and retrieval.

unavail Set media status to **Unavail** to disallow it from being used for storage and retrieval.

unmark Reset media mark status from **Check-Out**, **Export** or **Error** to **Unmarked**. Media marked as **Check-Out** can only be reset if they are still located within the Tertiary Manager system.

unsusp Reset media suspect count to zero.

-c *class* Change the policy class of blank media to the specified *class*.

-b Change the policy class of blank media to system blank pool.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsmedinfo(1), **fsmedout**(1)

NAME

fschstate − Change the state of a storage component in the Quantum storage subsystem.

SYNOPSIS

fschstate [**-F** *type*] **-s** *state componentalias*

DESCRIPTION

The **fschstate**(1) command changes the state of a specific storage component configured in the Quantum storage subsystem.

The **fschstate**(1) command can be executed when Tertiary Manager software is active or nonactive. Only storage subsystems states can be changed if Tertiary Manager or Media Manager software is down. Drive component changes require both software components to be active.

If a component is taken to the off-line state, Tertiary Manager software does not attempt any processing using that component. Changes in drive components and storage subsystems are reflected through the Media Manager software as well. If the Media Manager operator changes a drive's state to off-line, the Tertiary Manager software will reflect this change of state. This also works the same if the Tertiary Manager system administrator change the state of a drive or storage subsystem.

OPTIONS

componentalias

The alias for storage subsystem and drive components. The system administrator configures the possible values for component aliases during system configuration or by using the **fsconfig**(1) command. If the *componentalias* contains spaces, use single quotes around the words.

-s *state* The desired state of the drive components or subsystems. The valid values for drive components is *MAINT*, *ON*, or *OFF*. Valid values for subsystems is *ON* or *OFF*.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsconfig(1), **fsstate**(1)

NAME

fsclassinfo − Report policy class processing parameters, associated directory paths, and affinity lists.

SYNOPSIS

fsclassinfo [**-l**] [**-F** *type*] [*class ...*]

DESCRIPTION

For one or more policy classes, the **fsclassinfo**(1) command by default displays a short report that includes the processing parameters for each policy class. The **-l** option specifies a detailed listing for each policy class, consisting of the top level directory paths, the affinity list, and the copy configuration information.

If no policy classes are specified, a short report is displayed for all policy classes. If the **-l** option is specified without a policy class, a long report is displayed for all policy classes.

This command accepts upper case or lower case input. However, the input policy class is converted to lower case.

REPORT STATUS

The processing parameters listed by the **fsclassinfo**(1) command are as follows:

Soft Limit — The warning limit for the number of media that can be allocated in this policy class.

Hard Limit — The maximum number of media that are allowed in this policy class.

Drive Pool — The Media Manager drive pool group used to store or retrieve data for this policy class.

Target Stub Size (KB) — The size (in kilobytes) used to determine the number of bytes to leave on disk when files are truncated. This value will be the minimum number of bytes left on disk as the truncation processing will round the value up to a multiple of the file system block size.

Def Copies — The total number of copies that will be stored (including the primary copy) for each file in this policy class.

Max Copies — The maximum number of copies (including the primary copy) that are allowed for each file in this policy class.

Max Inactive Versions — The maximum number of inactive versions to keep for each file associated with the policy class (the current version is active, all others are inactive)

Media Type — The type of media to be used for the policy class.

File Cleanup — The file retention policy for the policy class. The output will either be **IMMEDIATE**, indicating that the files will be truncated immediately after being stored or **MINTIME**, indicating that the files can be truncated at the policy application time once all file copies are stored on a medium.

Media Cleanup — Indicates if the media should remain associated with the policy class or if it should be returned to the general scratch pool when all data has been logically removed from the media.

Store Min Time (mins, hrs or days) — The minimum time that a file must reside unmodified on disk before being considered a candidate for storage on media. A file will not be stored (by a store policy) until it has remained unmodified on disk for this amount of time. After that time, the next policy run will attempt to store the file. The 'm', 'h' or 'd' suffix on the reported value indicates minutes, hours or days.

Store Max Set Age (hrs) — Candidacy time limit (in hours) of the policy class. As soon as any file on the store candidate list in the policy class has remained unmodified for this amount of time, all of the files on the store candidate list should be stored.

Store Min Set Size (MB) The minimum set size (in MB) of the policy class. The store candidates in the policy class have to add up to this size before any of them can be stored by the policy engine.

Store Automatically Enable/disable automatic store. It decides if we allow the policy engine to automatically store files for this policy class.

Reloc Min Time (days, hrs or mins)
The minimum time that a file must reside unaccessed on disk before being considered a candidate for relocation. A file will not have its disk blocks relocated (by a relocation policy) until it has remained unaccessed on disk for this amount of time. After that time, a relocation policy will consider the file a valid candidate for relocation, but it may or may not actually be relocated. That will depend on the current file system fill level and the file system configuration parameters. The 'd', 'h' or 'm' suffix on the reported value indicates days, hours or minutes. NOTE: An "emergency" relocation policy ignores this time.

Trunc Min Time (days, hrs or mins)
The minimum time that a stored file must reside unaccessed on disk before being considered a candidate for truncation. A file will not have its disk blocks truncated (by a truncation policy) until it has remained unaccessed on disk for this amount of time. After that time, a truncation policy will consider the file a valid candidate for truncation, but it may or may not actually be truncated. That will depend on the current file system fill level and the file system configuration parameters. The 'd', 'h' or 'm' suffix on the reported value indicates days, hours or minutes. NOTE: An "emergency" truncation policy ignores this time.

Generate Checksum The state of Checksum Generation for the policy class. If **ENABLED**, checksums will be generated when files are stored by this policy class. The checksums will be retained for use in subsequent checksum validation during retrieve operations. If **DISABLED**, no checksums will be generated during stores. Note: policy-based Checksum Generation can be overridden by the **FS_GENERATE_CHECKSUMS** environment variable in *fs_sysparm*. See *fs_sysparm.README* file for details.

Validate Checksum The state of Checksum Validation for the policy class. If **ENABLED**, checksums will be generated when files stored by this policy class are retrieved. The generated checksums will be compared to the values generated when the files were stored. If a miscompare is detected a RAS ticket will be opened and, if there are additional copies of the file, the retrieve will be retried with another copy. If **DISABLED**, no checksums will be generated during retrieves and no validation will occur. Note: policy-based Checksum Validation can be overridden by the **FS_VALIDATE_CHECKSUMS** environment variable in *fs_sysparm*. See *fs_sysparm.README* file for details.

Dynamic Library Pooling
If **ENABLED**, the Tertiary Manager will direct store requests to specific drive pools on a rotating basis. DLP drive pool sets are configurable for each copy number via the DLP_COPY*_DRIVEPOOL_SET sysparms. See *fs_sysparm.README* for details. If **DISABLED**, no drive pool rotation will be performed, and the default drive pool will be used for store requests.

Expire Time (days, hrs, or mins)
The time from the last access time that a copy will be a candidate for deletion. A value of zero indicates the copy will never be a candidate for deletion.

Retrieve Order The order in which copies will be selected when retrieving files.

Alternate Store Location +**Disabled**|**Enabled** to show whether the Alternate Store Location option has been configured for the class.

Encryption	The encryption algorithm to be applied to the content of the file stored into an Object Storage system. This parameter is ignored for all other media types and is valid only if supported by the Object Storage system.
Master Key Name	The Master Key name that is used for the server-side AES256 encryption using the AWS Key Management Service (KMS).
Skip Retrieve Retry	This indicates if the default retrieve retry logic will be skipped.
Media Format	The media format type that will be used when formatting or selecting media.
Drive Limit	The maximum number of tape drives to use when the policy is run. **none** indicates drive limits are disabled.
Expire After Access	The minimum time per copy that a stored file must reside without being accessed before the copy becomes eligible for deletion.
Recreate Copy	Indicates if copies are to be recreated when referenced after being expired.
PUT Streams	The number of streams to use for stores when a file in the request is equal to or larger than the minimum size for using multiple streams.
GET Streams	The number of streams to use for retrieves when a file in the request is equal to or larger than the minimum size for using multiple streams.
Multi Stream Size	The minimum file size for using multiple streams.
Store Metadata	For stores to an object storage system, indicates if metadata is to be stored along with the object.

OPTIONS

class... One or more specified policy classes to be listed. Multiple policy classes must be separated by a space. A policy class name is a maximum of 16 alphanumeric characters. The special "-", ".", and "_" characters are permitted. If one or more policy class names in a list are invalid, an error message is generated, stating which policy class names were not found, and the report is generated for the remaining policy class names in the list. If *class* is not specified, a report is generated for all policy classes.

-l Long report. For the specified policy classes, this option provides all of the processing parameters, all associated directory paths, the disk affinity list, and the copy configuration information. The parameters listed are the *Soft Limit, Hard Limit, Media Type, Drive Pool, Default Copies, Max Copies, File Cleanup, Media Cleanup, Store Min Time, Store Max Set Age, Store Min Set Size, Store Automatically, MinRelocTime, MinTruncTime, Encryption, Erasure Coded Files, Master Key Name, Affinity List, Media Format, Drive Limit, Expire After Access, Recreate Copy, PUT Streams, GET Streams, Multi Stream Size, Store Metadata*, and the top-level directories associated with each policy class.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serv-

ing as an alternative to XML. See http://json.org for more information.

SEE ALSO

filesystems(4), **fsaddclass**(1), **fsmodclass**(1)

NAME

fsclassrnm − Rename an existing policy class

SYNOPSIS

fsclassrnm *oldclass newclass*

DESCRIPTION

The **fsclassrnm**(1) command changes a policy class. This command causes all files and media that were in the *oldclass* to be assigned to the *newclass*. If the policy class specified as the *newclass* already exists, the command will fail.

OPTIONS

oldclass

Policy class to be changed - The identifier specified in *oldclass* cannot be the same as that specified in *newclass*.

newclass

New policy class for the old policy class association. A policy class name can be a maximum of 16 alphanumeric characters. The special "-", ".", and "_" characters are also permitted.

SEE ALSO

fsclassinfo(1)

NAME

fsclean − Clean inactive versions from the Tertiary Manager, and delete Object Storage objects

SYNOPSIS

fsclean [**-F** *type*] **-m** *mediaID*... [**-t** [*endtime*]] [**-d**] [**-P**] [**-I** | **-B**]

fsclean [**-F** *type*] **-s** *filesystemmountpoint*... [**-t** [*endtime*]] [**-d**] [**-P**] [**-B**]

fsclean [**-F** *type*] **-c** *class*... [**-t** [*endtime*]] [**-d**] [**-P**] [**-B**]

fsclean [**-F** *type*] **-t** [*endtime*] [**-d**] [**-P**] [**-B**]

fsclean [**-F** *type*] **-r** [*mediaId*] [**-l**] [**-I** | **-B**]

fsclean [**-F** *type*] **-r** [*mediaId*] [**-p** *filename*] [**-P**] [**-I** | **-B**]

fsclean [**-F** *type*] **-C** [*requestId*] [**-W**]

fsclean [**-F** *type*] **-D** *directory* [**-a**] [**-R**] [**-I**]

DESCRIPTION

The **fsclean**(1) command removes (cleans) file versions by removing the internal data that references inactive version copies held on various media types. It also deletes objects when the copies are on Object Storage.

Every Tertiary Manager file has a set of copies on a collection of media tiers for (potentially) multiple versions of the file. The Tertiary Manager system tracks version copies along with the file pathname, timestamps, and version numbers. When the file is *active* in the directory name space, one of the versions is designated as *current*. The file's other versions are designated as *inactive*.

Deleted files are recoverable from inactive version copies with the **fsrecover**(1) command until their version copies are removed. The designated *current* version can be selected from the set of inactive versions with the **fsversion**(1) command.

The various options of the **fsclean**(1) command can be used to remove some or all inactive versions by media ID, time, directory, or file system criteria. In this way, **fsclean**(1) can be used to bring down the total number of inactive versions of a file to a number lower than the **Max Inactive Versions** defined in the policy class (see **fsclassinfo**). When operating on the files within a directory, the **fsclean**(1) command can also remove the current version and remove the file itself. When all of the version copies on a tape medium have been cleaned, the tape is returned to the blank-media pool.

The **fsclean**(1) command is run automatically as a feature of the **fsschedule** system-maintenance program to remove inactive file versions based on age, and to do follow-up processing for media that had **fsrminfo** processing done on them.

Versions can be cleaned by *media ID*, by *class*, by *file system mount point*, or by *endtime* criteria. To limit what is cleaned, the *-t* [endtime] option may be used with the *-m*, *-c*, and *-s* options. To clean all inactive file versions, use the **fsclean**(1) command with the **-t** option and no endtime specified. To clean only inactive versions for files that have been deleted, use the **fsclean**(1) command with the **-d** option and one of the above described options.

OPTIONS

-a When used with the **-D** option, active files in the specified directory get deleted, and all files that were deleted from that directory get cleaned.

-B Delete Object Storage objects using an asynchronous background task that continues after the clean processing completes and the command exits. A request ID is printed for use in tracking the background task. Without this option, the command does not exit until all cleaning and object deletions have completed. Production Tertiary Manager functionality can proceed while the background object deletions are running. This option has no effect if the media being cleaned is not an Object Storage type.

-c *class*...
Designate one or more policy classes (separated by spaces) for which to clean associated media. Use with the *-t* option to set a time-cutoff criterion for cleaning.

-C [*RequestID...*]

Report on the background object-deletion tasks started with the **-B** option. The report shows the percent completion of outstanding tasks. When one or more request IDs are provided, the report only covers those background tasks. Without the *requestId*, all in-progress task requests are reported.

NOTE: There is a period of a few seconds at the start of a background object-deletion task during which the task might not show up in this report.

-d Set a deleted-file criterion for selecting version copies to clean if they meet all the selection criteria. The **-d** option can be used with the **-c**, **-m**, **-s**, and **-t** options. Without this option, version copies for both *active* and deleted files are eligible for cleaning.

-D *directory*

Designate a directory to clean the files it contains. Versions are cleaned, records that reference the cleaned version copies are purged from the database (as with **-P**), and Object Storage objects are deleted in the background (as with **-B**). The directory can be active or deleted. The specified directory gets removed if it is empty and is not a relation point. The **fsrmrelation**(1) command must be run on a relation point directory before it can be removed. When used with the **-R** option, the entire directory hierarchy within the specified directory is cleaned.

Files in the directory are not recoverable following use of this option.

NOTE: Active files that have been removed immediately prior to running **fsclean**(1) may not get cleaned if file-deletion processing has not completed. Resubmitting the command resolves this issue. This effect is mitigated by using the **-a** option to delete the active files in the directory.

NOTE: Terminating the Tertiary Manager system before clean processing has completed does no harm, but the **fsclean**(1) command must be resubmitted on the unfinished directory to complete the clean processing.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

-I Do not delete the objects from Object Storage while cleaning version-copy components from the database. This option has no effect if the media being cleaned is not an Object Storage type.

WARNING: Cleaning permanently removes the StorNext information needed for deleting objects from Object Storage. Do not use this feature unless there is method outside of StorNext for using and eventually deleting the objects.

-l When used with the *-r* option, print a listing of media where rminfo processing has been run, and do not do the clean processing.

-m *mediaID...*

Designate one or more media (separated by spaces) to clean. Use with the *-t* option to set a time-cutoff criterion for cleaning.

-p *filename*

When used with the *-r* option, print into *filename* the list of files that are truncated and do not have all current-version copies (per policy). Those files should be retrieved so the missing version copies can be created. The report format is described at the bottom of this page.

-P Purge the recovery records from the database for files that have no remaining version-copy segments. This time-consuming process can be skipped if a quick response from the command is desired (for example: when there is an urgent need to clean old version copies to reuse their media for new version copies). This option is used with the *-r* or *-t* options to select the files for purge processing. In the interim between deleting version-copy segments and purging the recovery records, the affected files appear to be recoverable but they are not.

Note: The *clnver* **fsschedule**(1) feature performs recovery-record purges on a schedule.

-r [*mediaID*]

Finish the clean processing for files that had the **fsrminfo**(1) processing done on them. Cleaning includes the deletion of files that are truncated and have no version copies and the deletion of references to those files in the Tertiary Manager database. When the *-l* option is also specified, a list of related media is printed, and clean processing is not done.

When a *mediaID* is specified, **fsclean**(1) processing is applied only to version copies on the specified media.

The *-r* option is the only option that allows clean processing for Write Once Read Multiple (WORM) media.

-R When used with the **-D** option, the entire hierarchy starting with the specified directory gets cleaned.

-s *filesystemmountpoint...*

Designate one or more file systems (separated by spaces) for which to clean associated media. Use with the *-t* option to set a time-cutoff criterion for cleaning.

-t [*endtime*]

Set a point-in-time criterion for selecting version copies to clean if they meet all the selection criteria. All version copies that were made inactive at or before the *endtime* value are eligible to be selected. When the time specified is in the future, the current time is used, which is the default time value. Without this criterion, all versions are eligible for selection. The **-t** option can be used with the **-c**, **-d**, **-m**, and **-s** options.

The format for the time parameter is YYYY:MM:DD:hh:mm:ss where:
YYYY = Numeric, year

The following are optional; defaults are shown:,
MM = Numeric, two-digit month
(default:01 (January)),
DD = Numeric, two-digit day
(range: 01-31, default:01),
hh = Numeric hour
(range: 00-23, default:00),
mm = Numeric minute
(range: 00-59, default:00),
ss = Numeric second
(range: 00-59, default:00)

-W When used with the **-C** option, the command waits until the background object-deletion tasks complete. Periodic reports show the progress of the processing.

NOTE: There is a period of a few seconds at the start of a background object-deletion task during which the task might not show up in this report.

Report File Format

When using the **-r -p** option pair, a report file gets generated to list the media-location information for version-copy segments of the files being cleaned and their relative positions on those media. Using this information (with the aid of Quantum support), files can be grouped for optimal retrieve performance. When using the **-p** option, the following information gets generated for each file having missing copies:

Number of segments
: Shows if multiple media are required for retrieving a file.

Media Index
: Shows the internal ID (not the barcode media ID), used by StorNext.

Cumulative Data Block Number
: Shows where the file is located on the media.

Filename
: Full pathname of the file

SEE ALSO

fsmedinfo(1), **fsrecover**(1), **fsrminfo**(1), **fsrmrelation**(1), **fsschedule**(1), **fsversion**(1), **fsclassinfo**(1)

NAME

fsconfig − Configure or report on tape drive and subsystem components in the Quantum storage subsystem.

SYNOPSIS

fsconfig [**-p** *driveID*] [**-v** *devicepath*] [**-r** *drivetype*] [**-e** *delaytime*] [**-c y|n**] [**-F** *type*] **-a -h** *componentID* **-i** *componentalias* **-t** *componenttype*

fsconfig [**-p** *driveID*] [**-v** *devicepath* [**-f**]] [**-r** *drivetype*] [**-e** *delaytime*] [**-c y|n**] [**-F** *type*] [**-i** *componentalias*] **-m -h** *componentID*

fsconfig [**-F** *type*] **-d -h** *componentID*|**-i** *componentalias*

fsconfig [**-h** *componentID*] [**-F** *type*] **-s**

fsconfig [**-h** *componentID*] [**-F** *type*] **-L**

fsconfig [**-F** *type*] **-R**

fsconfig [**-h** *componentID*|**-i** *componentalias*] [**-F** *type*]

DESCRIPTION

The **fsconfig**(1) command adds, modifies, deletes, or reports on tape drive and subsystem component configuration settings for the Quantum storage system. By specifying the **-a**, **-m**, or **-d** options, hardware components can be added, modified, or deleted to reflect the actual physical configuration of the Quantum storage system. The **-s** option will scan the configured drives and update the device path if it is found to be wrong. The **fsconfig**(1) command should not be used to change the state of a hardware component. For that purpose, the **fschstate**(1) command should be used.

The **fsconfig -a** command adds a new component to the Quantum storage system. The *componentID*, *componentalias*, and *componenttype* are required values. (Note: The *componentID* cannot be modified after being added.) When adding a new tape drive, the corresponding subsystem (e.g. V0) must be configured beforehand. A new tape drive will also require the *driveID* and *devicepath* values. To add a tape drive, the Tertiary Manager software must be active. For a subsystem, it can be active or inactive.

The **fsconfig -m** command modifies an existing component in the system. A *componentID* value is required. To modify a tape drive, the Tertiary Manager software must be active. For a subsystem, it can be active or inactive.

The **fsconfig -d** command deletes components from the system. A *componentID* or *componentalias* value is required. To delete a tape drive, the Tertiary Manager software must be active, and no medium can be mounted in the drive. To delete a subsystem, the software can be active or inactive, but the drives associated with the subsystem must be deleted beforehand.

The **fsconfig -s** command updates tape drive device paths for the system and will not add new entries. If enabled in the *fs_sysparm* system parameters file, SCSI-3 persistent reservations will be established for all configured tape drives. This option is typically used when the Tertiary Manager software is started. If Tertiary Manager is running, it will need to be restarted in order for the changes to be applied.

The **fsconfig -L** command releases SCSI-3 persistent reservations and unregisters the reservation key from all configured tape drives. This option is typically used when the Tertiary Manager software is stopped.

The **fsconfig -R** command indicates that the Tertiary Manager subsystem should reload its in memory drive pool list. This should be called when drive pools are created, modified, or deleted using **vspoolcfg**(l).

The **fsconfig** command with no options generates a report showing all tape drive and subsystem components configured in the Quantum storage system. When specified with a *componentID* or a *componentalias* value, the report is limited to that component. Note that component alias names are case sensitive. To generate a report, the Tertiary Manager software can be active or inactive.

REPORT STATUS

The processing parameters listed by the **fsconfig**(1) command are as follows:

Component ID The unique component identifier.

Device pathname	The device path of the tape drive.
Compression	Whether tape drive compression is enabled. NOTE: This does not apply to LTFS media. LTFS media compression is controlled by the System Parameter LTFS_COMPRESSION.
User Alias	The component alias name.
Component Type	The type of component: DRIVE, SUBSYSTEM.
Device serial #	The serial number of the tape drive.
Drive Type	The type of tape drive: LTO, T10K, etc.
Drive ID	The numeric identifier of the tape drive.
Delay Time	The dismount delay time for the tape drive.
Vendor ID	The vendor identifier of the tape drive.
Product ID	The product identifier of the tape drive.
Product Revision	The firmware revision of the tape drive.

OPTIONS

-a Add a new component. The component identifier, component type, and component alias are required values when a new component is added. For a tape drive, the drive identifier and device path fields are also required.

-m Modify a component. The component identifier and at least one modifier is required.

-d Delete a component. The component identifier or component alias is required.

-s Scan for devices and update device paths as needed. Updates only take effect if the Tertiary Manager software is restarted after its execution.

-L Release SCSI-3 persistent reservations and unregister the reservation key from all configured tape drives.

-R Reload the drive pools in memory.

-p *driveID*

Numeric identifier of a tape drive. This maps a drive in the Tertiary Manager database to a Media Manager drive identifier. Drive identifier values must be greater than 0. Duplicates are not allowed.

-v *devicepath*

Device path of the tape drive. The device path is required for configuring new drives and consists of a variable string of up to 255 characters. Duplicates are not allowed, but can be overridden by the **-f** option.

-r *drivetype*

Type of tape drive. A drive type is required for configuring new tape drives only. The valid values are listed below. The default is **LTO**.

3592
LTO
T10K

-e *delaytime*

Dismount delay time (in seconds) for a tape drive. This value must be greater than or equal to 0. The default is obtained from the DRIVE_DELAY_DISMOUNT System Parameter.

-c y|n Enable or disable drive compression. The default is **n** (no compression). NOTE: This compression setting will not affect LTFS media. LTFS media compression is controlled by the System Parameter LTFS_COMPRESSION.

-h *componentID*

Component identifier. Examples: **V0** refers to a subsystem, and **V0,1** refers to a specific drive within that subsystem. Duplicates are not allowed. When specifying a drive component identifier, there is no space between the comma and number.

-i *componentalias*

Component alias. A component alias is a variable string of up to 255 characters. If the component alias contains spaces, use single quotes around the string. Duplicates are not allowed. The component alias string is case sensitive.

-t *componenttype*

Type of component. A component type is a multi-character identifier: **drive** or **subsystem**. For a drive, the subsystem must first be configured. This value can be specified in upper or lower case.

-f Force an update of a duplicate device path while Tertiary Manager software is running. Duplicate device paths are not normally allowed while Tertiary Manager software, this option will override that restriction. This option should not used be used unless directed by Quantum personal. This option is only allowed with the **-m** and **-v** options.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

RESTRICTIONS

All drives in a drive pool must be configured in the Tertiary Manager system before the drives can be accessed.

EXAMPLES

To update the device path to /dev/sg14 for tape drive component V0,1 while Storage Manger is up running,

```
fsconfig -m -h V0,1 -v /dev/sg14
```

To update all device paths while the system is not running

```
fsconfig -s
```

SEE ALSO

fschstate(1),**vspoolcfg**(l)

NAME

fsddmconfig − Configure or report all hosts and devices to be used in distributed data copy operations for the Quantum storage subsystem.

SYNOPSIS

fsddmconfig [**-q**]

fsddmconfig *hostname*

fsddmconfig -a|**-u** [**-s e**|**d**] [**-m** *max_movers*] [**-M** *max_movers]* [**-n** *threshold*] *hostname*

fsddmconfig -d *hostname*

fsddmconfig -a [**-s e**|**d**] **-i** *device_id* **-t** *device_type* [**-p** *path*] *hostname*

fsddmconfig -u [**-s e**|**d**] **-i** *device_id* [**-p** *path*] *hostname*

fsddmconfig -d -i *device_id hostname*

DESCRIPTION

The **fsddmconfig**(1) command reports, adds, modifies and deletes data mover configuration items. Note that this command, and the corresponding configuration info, is not normally needed. The default mode of operation for the Tertiary Manager software is to run all data copy operations locally on the metadata controller (MDC) server. This command is only needed if data copy operations are to be done on any clients in addition to, or instead of, the MDC.

To be able to perform copy operations on another host at least one tape drive, storage disk, or Object Storage device that is configured on the MDC must also be visible on that host. At least one managed file system must be visible on the host as well.

To configure a client to be used for distributed copy operations the following steps must be completed:

- Configure all tape drives, storage disks, Object Storage devices, and managed file systems as usual for use by the Tertiary Manager software. Use the **fsconfig**(1), **fsdiskcfg**(1), **fsobjcfg**(1) or **fsaddrelation**(1) commands respectively.
- Set the DISTRIBUTED_MOVING sysparm to **all** or **threshold**. By default the sysparm has a value of **none**. See NOTES section below.
- Use this **fsddmconfig**(1) command to configure hosts being used for data copy operations.
- Use this **fsddmconfig**(1) command to configure devices on hosts in preparation for data copy operations.

Note that if a host is disabled via **fsddmconfig**(1) it will not be used for moving data even though all devices on the host may be enabled. This makes it easier to disable a host for a time if needed.

If you are planning on using an sdisk via NFS there is a permissions issue. That file system must be exported with the "no_root_squash" option set so that the client will have permission to make required updates.

A device on a host can be automatically disabled by the Tertiary Manager software. This is only done if there is an unrecoverable error on the device that can only be fixed by operator intervention. For example no configured device on the host matches the device identifier requested. The **fsddmconfig**(1) command can be used to re-enable the device when the situation has been resolved.

A host may also be automatically disabled by the Tertiary Manager software. This is done if the version of software on the client does not match the MDC. When the software on the client, or MDC, is updated then the **fsddmconfig**(1) command can be used to re-enable the host. The host may also be disabled when there are communication issues with the mover running on that host and the MDC. For this scenario, the software will continuously try to repair itself.

REPORT STATUS

There are three reporting options for the command. The reports and their corresponding output are shown here. For a more detailed description of the configuration items reported see the OPTIONS section.

At the bottom of each of these three reports is a line that indicates the state of the DISTRIBUTING_MOVING sysparm. An example of this line is:

Distributed Data Mover State: Enabled

If the sysparm is set to "all" then the displayed state is **Enabled**. If the sysparm is set to "threshold" then the displayed state is **Threshold**. If the sysparm is set to "none" then the displayed state is **Disabled**.

NOTE: As indicated in the DESCRIPTION section above, the default mode for copy operations is to run all of them on the MDC. Because the MDC is the default host, it does not show up in any of the reports below. This will be true unless the MDC host is specifically configured via the fsddmconfig command. This restriction applies even to the "Host Report" mentioned below. If the report is issued for the MDC, and the MDC was not specifically configured via fsddmconfig, then the requested report will be empty.

Quick Report

Mover Host The name of the host where distributed moving is configured.

Current State Is the host currently enabled for use?
Enabled - the host is enabled for use
Disabled - the host is not enabled for use

Server Threshold The number of simultaneous mover copy requests to run locally on the MDC before running any of them on a client.

Maximum Movers Server The maximum number of mover copy requests to run simultaneously on the host when the host is the MDC server.

Maximum Movers Client The maximum number of mover copy requests to run simultaneously on the host when the host is a client machine.

Host Report

State Is the host currently enabled for use?
Enabled - the host is enabled for use
Disabled - the host is not enabled for use

Server Threshold The number of simultaneous mover copy requests to run locally on the MDC before running any of the requests on a client. This field is only shown for the MDC host.

Max Movers Allowed The maximum number of mover copy requests to run simultaneously on the host.

For each device configured on the host the following is also shown:

Device Identifier The character string identifying a device.

Type The type of device.
TAPE - a tape drive
SDISK - a storage disk
OBJECT_STORAGE - an Object Storage device
FS - a managed file system

State Is the device currently enabled for use?
Enabled - the device is enabled for use
Disabled - the device is not enabled for use
DISABLED - the device has been automatically disabled

Path The pathname for the device as it is seen on the host to be used.

Default Report

Device Identifier The character string identifying a device.

Type The type of device.
TAPE - a tape drive
SDISK - a storage disk
OBJECT_STORAGE - an Object Storage device
FS - a managed file system

State	Is the device currently enabled for use? Enabled - the device is enabled for use Disabled - the device is not enabled for use DISABLED - the device has been automatically disabled
Host	The name of the host where distributed moving is configured.
Path	The pathname for the device as it is seen on the host to be used.

OPTIONS

Note that with no options or args the **Default Report** is displayed.

hostname

The *hostname* argument can be either IP address or hostname. The hostname can be either a local (single word) hostname or a fully qualified domain name (FQDN). Any *hostname* other than previously stated is considered ambiguous. The Tertiary Manager software will attempt to match a specified FQDN hostname to the host that it belongs to. The match attempt could fail due to improper DNS setup. In such case, users can either ensure that their DNS environments are setup correctly or specify a *hostname* that matches exactly with the output of the **hostname**(1) command when run on the respective host.

If this host is configured with device type **object_storage** and **https**, please refer to the **fsobjcfg**(1) man page for https configuration requirement for DDM hosts.

Note that with no options and just a hostname argument provided the command will display the **Host Report** for that host.

-q Generate the **Quick Report** of hosts configured for distributed moving operations.

-a Add the specified host or device to the configuration.

-u Update the configuration for the specified host or device.

-d Delete the specified host or device from the configuration.

-s e|d Set the state of the specified host or device to enabled (**e**) or disabled (**d**).

Note that disabling the MDC isn't recommended as it will restrict capabilities. Requests that are unable to run on any other DDMs due to threshold limits or them being disabled will be failed instead of using the MDC as a last resort. Additionally, the following requests which are limited to the MDC won't be able to run:

partial file retrieves
foreign file system retrieves
alternate node retrieves.

-m *max_movers*

Set the maximum number of simultaneous mover copy requests allowed on the host when it is a client. A value of **all** indicates no limit.

-M *max_movers*

Set the maximum number of simultaneous mover copy requests allowed on the host when it is running as the MDC server. A value of **all** indicates no limit.

Note that this value can be ignored if the current configuration is such that not all resources are covered. If there is nowhere else to run a request it will run locally on the MDC server providing it is enabled. For example assume a simple configuration with a server and a single client with eight shared tape drives. There can be eight copy operations occurring at the same time. If the maximum client movers is set to two and the maximum server movers is set to four, when the seventh and eighth simultaneous copy requests are run they will default to running on the MDC server.

If there is a desire to run requests on all DDM hosts except the MDC server then this value should be set to 0. This won't prevent the MDC from ever being used but will limit the scenarios in

which it is used. The MDC would still be used for requests that are only allowed be run on the MDC. Requests could also be run on the MDC when all other DDMs have hit their threshold limits (as described above) or are in a disabled state.

-n *threshold*

Set the number of simultaneous mover copy requests to run on the server before giving work to the clients. This value is only used when DISTRIBUTED_MOVING is set to **threshold**. When a new host is configured this value defaults to zero. When the value is zero, and DISTRIBUTED_MOVING is set to **threshold**, the system acts like the DISTRIBUTED_MOVING value is **all**. See NOTES section below.

-i *device_id*

The character string identifying a device. For tape drives the identifier is the serial number of the drive as reported by **fsconfig**(1). For storage disks it is the alias of the sdisk as reported by **fsdiskcfg**(1). For Object Storage devices it is the alias of the controller I/O path as reported by **fsobjcfg**(1). For managed file systems it is the name of the file system as it appears in the **/usr/cvfs/config/fsmlist** file. Note that there is also a pseudo device identifier "disk2disk" that is used for relocation operations.

-t *device_type*

The type of device to add. Valid values are **tape** for tape drives, **sdisk** for storage disks, **object_storage** for Object Storage devices, and **fs** for managed file systems.

-p *path* The pathname for the device as it is seen on the host to be used. For tape drives the path is as reported by **fsconfig**(1). For storage disks the path is as reported by **fsdiskcfg**(1). For managed file systems the path is the mount point of the file system. For Object Storage devices the path is not valid. Note that the device path values and mount points on the server and client may differ.

NOTES

The DISTRIBUTED_MOVING sysparm is used as the overriding mechanism for enabling or disabling data movement operations on distributed clients. It can be disabled by setting the value to **none**, the default, or be enabled by setting the value to **all** or **threshold**. Note that like many other sysparms, when a change is made the Tertiary Manager software must be cycled.

A **none** value means all copy operations will be done locally on the MDC.

An **all** value means all configured hosts will be treated equally when deciding where to run a copy request. A host is selected based on the fewest number of current active requests. For ties the host that has the oldest allocation time will be used.

A **threshold** value means the MDC is weighted more heavily than distributed clients when deciding where to run a copy request. The copy requests are assigned to the local host until the threshold number is reached and then the **all** strategy is applied. The threshold number defaults to zero but can be configured via **fsddmconfig**(1). To truly operate in **threshold** mode the value for the MDC host must be non-zero. With a zero value the system immediately applies the **all** strategy.

If the value of the sysparm is **none** no distributed operations will be done but running the **fsddmconfig**(1) command will still show what hosts/devices are configured. This allows for disabling all distributed operations for a time without having to disable or delete all individual configuration items. If all items are configured and enabled and no distributed operations are occurring check the DISTRIBUTED_MOVING sysparm.

It is recommended that all MDC hosts are configured for distributed data moving if the system is running in a failover mode.

SEE ALSO

fsconfig(1), **fsdiskcfg**(1), **fsobjcfg**(1), **fsaddrelation**(1)

NAME

fsdefrag − Report or defragment all currently fragmented media.

SYNOPSIS

fsdefrag -d [*mediaID...*] [**-p** *destinationdrivepool*] [**-S** *sourcedrivepool*]

fsdefrag -d [**-v**] [**-n** *numMedia*] [**-p** *destinationdrivepool*] [**-S** *sourcedrivepool*]

fsdefrag -s

fsdefrag [**-v**] [**-n** *numMedia*]

DESCRIPTION

The **fsdefrag**(1) command provides the capability to report or defragment all fragmented media in the system. This command will only report or defragment media that are fragmented based on parameters in *fs_sysparm*.

The **fsdefrag**(1) command and the **fsmedcopy**(1) command are similar in functionality but differ in the following ways:

Reporting
: The **fsdefrag**(1) command will only report media that are fragmented based on the parameters in *fs_sysparm*. This is true even if a mediaid is provided to the command. No report will be given if the media is not fragmented. The **fsmedcopy**(1) command will report on any and all media in the system that contain any data.

Defragmentation
: Again, the **fsdefrag**(1) command will only defrag media that are fragmented based on the parameters in *fs_sysparm*. The **fsmedcopy**(1) command, in replace mode, can be used to replace any media.

NOTE The **fsdefrag**(1) command in defragment mode actually invokes the **fsmedcopy**(1) command to perform the replacement of a media.

Fragmented Media

As indicated above the **fsdefrag**(1) command works on media that are fragmented based on parameters in *fs_sysparm*. The parameters used are:

DFG_FULL_PERCENT
: The percent at which a media is considered full. A media whose fill percentage is greater than or equal to this value is considered full.

DFG_FRAG_PERCENT
: The percent at which a media is considered fragmented. A media whose fragmentation percentage is greater than or equal to this value is considered fragmented. (This is the wasted space in the used portion of the media.)

A media will only be reported or defragmented by the **fsdefrag**(1) command if the media is full AND fragmented. A media which is only full or only fragmented will not be processed by **fsdefrag**(1).

In report mode this command will show the media that are fragmented and that will be defragmented by future defragmentation runs. Fragmented media that are not available for defragmentation are not included in the report by default. A verbose mode exists so even those media are included in the report if desired.

When reporting on fragmented media one column of the report 'Available' is included to show the current availability of the fragmented media. This column is included even in the default report where all media reported are available. Also in that column along with the availability indicator is a '(Vault)' for media that currently reside in a vault archive. Being located in a vault affects availability and how a media will be treated during defragmentation. See DFG_IGNORE_VAULTED_MEDIA below.

No storage disk media or Object Storage media will be defragmented by this process. Tape media become fragmented when **fsclean**(1) is run because db contents are deleted but space on the tape is still used. When **fsclean**(1) is run on an sdisk the data is removed from the media as the db contents are cleaned. This command does however report fragmentation levels on sdisk media in verbose and summary modes for infor-

mational purposes. This can be helpful in indicating that other data is present on the file system besides just the sdisk contents. This command does not report fragmentation levels on Object Storage media since the data is removed from the media when the database is cleaned.

Note that the media determined to be fragmented are sorted by this command. This is true when reporting fragmented media and also when using the **-d** option for initiating the defragmentation of media. The media are sorted from the highest fragmentation percentage to the lowest. Additionally tape media appear in the list before sdisk media when verbose mode is used regardless of fragmentation percentage.

Other Fragmentation Parameters

Some other system parameters also affect defragmentation and the processing done by the **fsdefrag**(1) command. The parameters are:

FS_MAX_ACTIVE_TAPECOPIES

This is the max number of **fsmedcopy** operations that can be run simultaneously. Let's assume that this value is 3 and that there are 10 fragmented media on a system when **fsdefrag**(1) is run; only 3 **fsmedcopy** operations will be run at a time and the other media will be queued by **fsdefrag**(1). Note that 2 drives are required for a **fsmedcopy** operation.

DFG_MAX_MEDCOPY_ATTEMPTS

This is the max number of times that the **fsdefrag**(1) command will invoke the **fsmedcopy**(1) command for a media in an attempt to replace that media. After the max has been reached no further attempts will be made to replace the media until the next **fsdefrag**(1) command is run. Each command will make the max attempts unless a media has become unavailable for mounting.

DFG_IGNORE_VAULTED_MEDIA

The action to take when a fragmented media that is currently in a vault archive is encountered. If the value is true then a media in a vault will be ignored and no defragmentation will be attempted. If the value is false then the media will be processed with all other fragmented media. Note this parameter will also affect what is reported by the command. Vaulted media will only be reported in verbose mode unless this parameter is false.

CLEAN_VERSIONS_EXPIRATION

This parameter is not used directly by the **fsdefrag**(1) command but does affect automated defragmentation operations. It is the age used by the **clnver** schedule item for determining which inactive file segments are old enough for removal from the database. It is this cleaning of the old file segments from the database that causes a media to become fragmented.

Scheduled Tape Defragmentation

Via the **fsschedule**(1) command; Scheduled tape defragmentation can be configured using the **defrag** feature type. The scheduler daemon will execute the **fsdefrag** command per the configured schedule to identify and defragment fragmented tape media. Below are some 'Best Practices' to keep in mind:

The **defrag** schedule item should be scheduled a few hours after the **clnver** item for best results. The two features work together in managing out of date media contents.

The value for DFG_FULL_PERCENT is recommended not to be configured below 50%. If the parameter is set below this value it will likely result in 'thrashing' with media constantly being defragmented.

The value for DFG_FRAG_PERCENT is recommended not to be configured below 50%. The optimal value will be determined by the type of files being stored to tape. For example if a large percentage of files being stored to tape are temporary files then a higher DFG_FRAG_PERCENT value of 95% will prevent defragmentation from occurring too often. If a large percentage of files are stored but rarely deleted, then a lower DFG_FRAG_PERCENT value like 65% may be configured.

The values for DFG_FULL_PERCENT and DFG_FRAG_PERCENT may require tuning on your system. The defaults for the parameters are set to be conservative so that defragmentation does not occur too often. Once the **defrag** feature has been scheduled you should monitor the number of tape media being defragmented and adjust accordingly. Note that when the feature is turned on for the first time there may be a large number of tapes being processed.

Defragmentation duration time can be adjusted by using the DFG_MAX_MEDIA_TO_DEFRAG parameter to limit the number of tape media processed during each scheduled defragmentation. The DFG_MAX_MEDIA_TO_DEFRAG parameter will determine the **-n** value provided to this command by the scheduler. A value of 0 implies all media and the **-n** is not used.

Note: Tape Defragmentation default parameters are listed in the *fs_sysparm.README* file.

OPTIONS

mediaID...

The media to defragment or to report on. Multiple media identifiers must be separated by spaces. The number of media identifiers that can be entered is limited by the local machine's command-line length.

-d Perform defragmentation on the media. The default is to just report the fragmented media.

-n *numMedia*

Limit the operation, either reporting or defragmentation, to numMedia media. Since the media are sorted by fragmentation level this will result in the most fragmented tape media being reported or defragmented. If zero is provided for numMedia the default command behavior is performed which is to report/defragment all indicated fragmented media.

-p *destinationdrivepool*

Media Manager drive pool group used for the destination media when copying the fragmented media. The drive pool must be defined in Media Manager software. If the **-p** option is not used, the default drive pool group will be specified by the policy class definition. The special "_" character is permitted to identify the drive pool group.

-S *sourcedrivepool*

Media Manager drive pool group used for the fragmented media when copying the media. The drive pool must be defined in Media Manager software. If the **-S** option is not used, the default drive pool group will be specified by the policy class definition. The special "_" character is permitted to identify the drive pool group.

-v This option when used in defragment mode, indicates that media to be left unprocessed due to availability etc. should be reported along with those being defragmented. Note that in defragment mode when this option is used in combination with **-n** the unprocessed media do not count against numMedia. Only media that are to be processed are included in the count.

-v This option when used in report mode indicates verbose output. In verbose mode unavailable fragmented media are included in the report. By default unavailable media (and media that are in a vault if those are being ignored) are not included in the report of fragmented media, regardless of the fragmentation levels. Also included with this option is a list of storage disk media at the end of the report.

-s This option indicates summary report mode. In summary mode only a count of fragmented media is given along with a breakdown of the total count. The total fragmented media are broken into: available, unavailable, storage disks and media in a vault. All media in a vault are included in the vault count and not in the available/unavailable tallies regardless of their availability.

SEE ALSO

fsmedcopy(1), **fsclean**(1), **fsschedule**(1)

NAME

fsdevice − Reports StorNext related SCSI devices

SYNOPSIS

fsdevice -s [**-f**] [**-c** *criteria*]

fsdevice -b [**-l**] [**-c** *criteria*]

fsdevice -f [**-l**] [**-c** *criteria*]

fsdevice -h

DESCRIPTION

The **fsdevice**(1) command reports SCSI devices as detected by the common device infrastructure of the Quantum storage system. By default **fsdevice**(1) will probe all scsi devices seen by the server to obtain the listing of devices. Further inquiries to specific device paths or StorNext configuration databases and files will be used to fill in all information reported. The default command will generate a tabular, parsable report that lists all detected devices that can be or are already used by StorNext. Adding the -l option presents the devices in a human readable format along with displaying archive and tape drive relationships.

The -c option gives the additional ability to filter the detected devices for one or more devices that match the specified criteria. Devices will be printed that match or partially match the serial number, SCSI device path, system device path, vendor, product, device type string (e.g. sequential access, direct access, media changer), configured name or alias of the device, or the unique identifier as shown in the long listing (-l).

fsdevice(1) can be run with Tertiary Manager software active or inactive, but requires the database to operate.

REPORT STATUS

The columns reported by the default **fsdevice**(1) command are as follows:

System Device Path
SCSI Device Path
Vendor
Product
Revision
Serial Number
Host Bus
Channel
SCSI ID
Logical Unit Number (LUN)
Device Type (0 - direct; 1 - sequential; 8 - changer)
Is Configured in StorNext (0 or 1)

OPTIONS

-s Prints the slot to device path mapping for supported scsi archives

-l List devices in a human readable format

-b When multiple paths to the devices are present, this option will report each device only once by first using a round-robin load balancing scheme across the detected host buses. The result will be a unique set of devices reported instead of the duplicated list normally printed in the default report of **fsdevice**(1) or by **fs_scsi**(1).

-f Unlike the default detection that starts with probing all SCSI devices seen from the server, this will look to StorNext's configuration databases and files to detect the devices. SCSI Information Inquiry information will then be used to fill in the information for the devices. The resulting list will be only the items configured into StorNext.

-c *criteria*

The text to filter the devices on. The criteria will be compared against the vendor, product, serial number, device type string (e.g. sequential, direct, changer), system device, SCSI path, configured name, and the unique id. Any partial match will be reported.

EXAMPLES

To look for all Quantum products in a human readable format issue:

```
fsdevice -c Quantum -l
```

To look for all disk devices:

```
fsdevice -c 'direct access'
```

or, more simply:

```
fsdevice -c direct
```

To look for all configured disk devices:

```
fsdevice -fc direct
```

SEE ALSO

fs_scsi(1), **fsconfig**(1), **cvlabel**(8), **vsarchiveqry**(l), **vsdriveqry**(l), **dbdrvslot**(1M)

NAME

fsdirclass − Report the policy class associated with a directory.

SYNOPSIS

fsdirclass [**-F** *type*] *directory*

DESCRIPTION

The **fsdirclass**(1) command displays the directory-to-policy class association of the specified directory. If the directory specified is not associated with a policy class, an error message is issued.

REPORT STATUS

The report produces the following information:

Directory Name
: The directory path

Class
: The policy class name associated with the directory

OPTIONS

directory
: The directory path name for which the associated policy class is returned. The name must be less than 256 characters long. The entire path name need not be entered. The directory path will be resolved using the current working directory. If preceded by a slash (/) in the command definition, the full path name, starting from the root directory, is required as input to the command. Otherwise, the Tertiary Manager software expands the directory name using the current working directory as the parent.

-F *type*
: Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

  TEXT is the "legacy" textual format.

  JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

SEE ALSO

fsclassinfo(1)

NAME

fsdiskcfg − Configure or report all storage disks in the Quantum storage system.

SYNOPSIS

fsdiskcfg -a -p *pathname* [**-c** *copy*] [**-s** *streams*] *alias*

fsdiskcfg -m [**-p** *pathname*] [**-c** *copy*] [**-s** *streams*] [**-I** *uuid*] *alias*

fsdiskcfg -d *alias*

fsdiskcfg -l

fsdiskcfg -r *alias*

fsdiskcfg [*alias*]

DESCRIPTION

The **fsdiskcfg**(1) command adds, modifies, deletes, and reports configuration settings for storage disks in the Quantum storage system. Storage disks are used by the Tertiary Manager software as secondary storage similar to tape media.

For any **fsdiskcfg**(1) command option, the Tertiary Manager software can be active or inactive.

Submitting the **fsdiskcfg**(1) command with no options generates a report showing all Quantum storage disks that are currently configured. A report for one storage disk can be generated by specifying the corresponding *alias*. Alias names are case sensitive and must match exactly or the request will fail. Additionally, the **-l** option can be used to generate a list of file system mount points which could possibly be used as a storage disk.

The **fschstate**(1) command can be used to change the state of a storage disk once it has been configured.

OPTIONS

-a Add a new storage disk. The alias and path are required values. If the command is invoked and any of these parameters are omitted, an error message is returned. The error message identifies the field that was not entered.

-m Modify a storage disk. The alias and at least one modifier is required. The alias cannot be modified.

-d Delete a storage disk. The alias is required. A storage disk can not be deleted if it contains copies of any user data. The **fsrminfo**(1) command can be used to purge any data prior to removing a storage disk from the configuration.

-l List candidates. This will provide a list of file system mount points that could be used for a storage disk. It is possible that some entries in the list are not usable due to permissions. This is validated when a specific path is specified when adding a storage disk. This list will not include file systems which are already being used for a storage disk.

alias Alias. An alias is a variable string of up to 16 characters. If the alias contains spaces, use single quotes around the string. Duplicates are not allowed. This command is case sensitive. The *alias* is used to uniquely identify a storage disk.

-p *pathname*
Storage disk directory pathname. The pathname is required for configuring new storage disks and consists of a variable string of up to 255 characters. Duplicate pathnames are not allowed nor are two pathnames residing under the same mount point. A pathname which resides under a relation point, in a temporary file system, or in the root file system will not be allowed either.

-c *copy* File copy number. Media are associated with a specific file copy number. This is done so all copies are stored to different media. Due to the limited number of storage disks, it may be desirable to pre-configure the copy number that is associated with a storage disk for systems using multiple copies. This ensures that there is a storage disk available for each copy.

-r Refresh the available and used space of the storage disk. This is required when the storage disk space has been reconfigured - extended or reduced.

-s *streams*
Maximum number of I/O streams. This allows the number of concurrent I/O operations for a storage disk to be adjusted. The specified value must be greater than 0.

-I *uuid* Uuid uniquely identifies a specific storage disk in the Quantum storage system. Uuid, if given must conform to RFC 4122 - A Universally Unique IDentifier (UUID) URN Namespace.

RESTRICTIONS

The system will only allow a maximum of 16 storage disks to be configured.

The file system associated with the *pathname* for the storage disk must be mounted when performing add, modify or delete operations.

SEE ALSO

fschstate(1), **fsrminfo**(1)

NAME

fsdismount − Unmounts specified drive or media.

SYNOPSIS

fsdismount [**-w**] *drivealias*

fsdismount -m *mediaID*

DESCRIPTION

The **fsdismount**(1) command unmounts a medium from a drive. The drive state can be on-line or off-line for the **fsdismount**(1) command to be issued. It is recommended that the drive be placed in the off-line state using the **fschstate**(1) command before performing the dismount if multiple requests are allocated against that medium. If the drive was placed in the off-line state, any additional queued request against that medium will be suspended until the drive is transitioned back to the on-line state.

If the drive with the medium to be dismounted is in use (retrieve or store) when the **fsdismount**(1) command is executed, an error message will be returned. The error message notifies the user that the **fsdismount**(1) command has failed because medium is being accessed or delayed in dismount. Executing the **fsstate**(1) report for the drive shows the status state of the drive.

The **-w** option forces the **fsdismount**(1) command to wait for the medium to enter the appropriate state instead of failing. Once the drive enters the *DELAYED* or *FAILED* status state, the medium will be dismounted from the drive.

If **-m** option is used to dismount a specific media ID, then it is unnecessary to run the **fschstate**(1) command if the **fsmount**(1) command was used to get the media mounted. The **fsstate**(1) report will show the status state for drive with this media mounted as *USER MOUNT*. If the media was mounted with **fsmount**(1) then the **-m** option must be used to dismount the media. If the media was mounted by other Tertiary Manager processing then all rules and restrictions when dismounting by drive alias apply.

OPTIONS

drivealias

Drive alias. A drive alias is a variable string of up to 255 characters. If the drive alias contains spaces, use single quotes around the string.

-w Suspends dismount request until drive completes all queued requests and enters appropriate state for dismount to occur.

-m *mediaId*

Dismount by media id. The media identifier must not be longer than 16-characters and will be known by the Media Manager software but may or may not be known to the Tertiary Manager software.

NOTES

The Media Manager software should not be used to dismount a medium from a drive when the Tertiary Manager was used to mount it.

This command is not valid for storage disks.

SEE ALSO

fschstate(1), **fsconfig**(1), **fsmount**(1)

NAME

fsdrvclean − Cleans the specified drive.

SYNOPSIS

fsdrvclean *drivealias*

DESCRIPTION

The **fsdrvclean**(1) command mounts a cleaning medium into the specified drive. If the drive state is not on-line or cleaning is in progress for the drive, then the request will fail.

OPTIONS

drivealias

Drive alias. A drive alias is a variable string of up to 255 characters. If the drive alias contains spaces, use single quotes around the string.

NAME

fsdumpfind − Utility used by foreign file system migration to dump the namespace information needed by SNSM.

DESCRIPTION

The **fsdumpfind**(1) utility is used by foreign file system migration to dump the namespace information needed by SNSM.

WARNINGS

Users should not run this command, unless directed to do so by Quantum technical assistance.

NAME

fsdumpnamespace − get namespace data from a foreign server

SYNOPSIS

fsdumpnamespace *foreignHost foreignDumpDir foreignWorkingDir outputDir*

DESCRIPTION

This script will put dump namespace scripts on the foreign host. Then it will execute the scripts on the foreign host, dumping the foreign namespace. The results are then retrieved and put into *outputDir*/**dump.txt** and *outputDir*/**dumpErrors.txt**. The **dump.txt** file can then be used as input for the **fsimportnamespace**(1) command.

The *foreignDumpDir* is the root directory for the namespace dump. It must be an absolute directory name (e.g. /usr/local instead of local").

The *foreignWorkingDir* will be used as storage for the dump scripts and for the the dump data. Please ensure that the *foreignWorkingDir* has sufficient free space. The files in the *foreignWorkingDir* are not cleaned up after the command completes.

The *outputDir* will be used as the output destination for the dump.txt and dumpError.txt files. Please ensure that the *outputDir* has sufficient free space. After the command completes, it is strongly recommended that the **dump.txt** and **dumpError.txt** files are archived.

The amount of space can vary based on lengths of filepaths, but estimate .5GB per million files and directories to be dumped (on both the *foreignWorkingDir* and the *outputDir*).

Note that the dump process can take a long time, depending on the system size and speed.

The caller's **netrc**(4) file must be set up for the FTP to the foreign host. The caller should also have a **rhosts**(4) file set up so that **rsh**(1) can be used to execute commands on the foreign host.

NAME

fsechostderr − Utility used by foreign file system migration to echo what is passed to it to standard error.

SYNOPSIS

fsechostderr *text*

DESCRIPTION

The **fsechostderr**(1) utility is used by foreign file system migration to echo what is passed to it to standard error.

WARNINGS

Although running **fsechostderr**(1) utility stand alone will not harm anything, it is used by foreign file system migration and is not intended to be run stand alone.

NAME

fsexclusioncheck − checks files against exclusion criteria

SYNOPSIS

fsexclusioncheck -h

fsexclusioncheck [**-ds**] [**-t** *type*] [**-c** *criteria_file*] *file...*

DESCRIPTION

The **fsexclusioncheck**(1) command will allow you to verify that the criteria in the exclusion files will work as expected. Criteria specifications can be checked against file names/paths without impacting the system if the alternate criteria file is specified. The command outputs the complete file path(s) and text indicating if they meet or do not meet the exclusion criteria.

OPTIONS

-h Displays usage

-d Turns on debug logging, which appears in /tmp/logs.

-s Silently executes the check without generating any output

-t *type* The valid types are *store*, *truncate*, and *postrestore*. The default is *truncate*. The selected type will determine which exclusion criteria file is to be used unless the **-c** option is used. It will also affect how the specified files are processed. For store exclusions only the file name is used, while the other exclusion types use the full path name.

-c *criteria file*

This specifies an alternate criteria file to use instead of the one corresponding to the type of exclusion check being requested. This is useful for validating criteria before putting them into production. The default exclusion files used for each type are:

store	*/usr/adic/TSM/config/excludes.store*
truncate	*/usr/adic/TSM/config/excludes.truncate*
postrestore	*/usr/adic/TSM/config/excludes.postrestore*

file... The file name/path to validate.

RETURN VALUE

0 When none of the specified files meet the exclusion criteria.

1 When all of the files meet the criteria.

2 When some of the files meet the criteria, or any error.

EXAMPLES

Example 1: Checking a file that meets store exclusion criteria against the default store exclusion file. The following shows the command and its output:

```
example% fsexclusioncheck -t store /sn/fs1/policyclass/myfile
/sn/fs1/policyclass/myfile is excluded
```

Example 2: Checking a file that meets truncation exclusion criteria against the default truncation exclusion file. The following shows the command and its output:

```
example% fsexclusioncheck /sn/fs1/policyclass/myfile
/sn/fs1/policyclass/myfile is excluded
```

Example 3: Checking files that do not meet truncation exclusion criteria based on criteria in the testCriteria exclusion file. The following shows the command and its output:

```
fsexclusioncheck -t truncate -c testCriteria fileA fileB
/sn/fs1/policyclass/fileA is not excluded
/sn/fs1/policyclass/fileB is not excluded
```

SEE ALSO

exclusions(4)

u.

NAME

fsexpcopy − Expire copies of a file

SYNOPSIS

fsexpcopy -c *copynum* [**-F** *type*] *filename...*

fsexpcopy -c *copynum* [**-F** *type*] **[-H|-L]**-R *directory*

fsexpcopy -c *copynum* [**-F** *type*] **-D** *directory*

fsexpcopy -c *copynum* [**-F** *type*] **-B** *batchfilename*

fsexpcopy -S [**-F** *type*]

DESCRIPTION

The **fsexpcopy**(1) command allows a user with the UNIX write-to-file permission to expire numbered copies from all versions of the specified file(s). The copy numbers must be configured for expiration.

The copy expiration feature provides for automatic removal of copies based on time since last access of a file. This simplifies the management of high-performance, low-volume storage tiers. For example, consider a class policy that calls for the storage of two copies: one on object store and one on tape. The class copy configuration could state that the object store copy should be expired after a file has not been accessed for thirty days. This would free space on the object store for use by recently accessed files, which could improve retrieve-latency performance for data removed from disk.

When a file is created or modified, all copies for the current version must first be created before any copies can be expired. Copies can be manually expired before the expiration interval has elapsed. Note when running without the **-S** option, policy settings are ignored and copies are immediately expired. When the last remaining copy is expired, the file is removed from disk regardless if the file's data is on disk or truncated. When a copy has been expired, it is not stored again unless: 1) the file has the re-creation option configured for the copy, and 2) the file is retrieved or truncated while its atime value plus expiration interval is in the future.

Configuration of the expiration feature is done with the **fsmodclass**(1) command on a per-class per-copy basis. Each copy can be configured with an expiration interval and a yes/no switch for automatic re-creation when a file's access time is updated. Expiration of the last remaining copy requires the **-X** option to be set for the class policy.

In the case of an expired copy that is specified to be re-createable and the expiration interval has not elapsed, an attempt to remove data from disk will be refused, and re-creation of the copy will be started.

Copies may not be expired for Write Once Read Multiple (WORM) media types.

OPTIONS

-c *copynum*

The *copy number* of *filename* to expire. When expiring the last remaining copy of the file for the current version, the file is removed from disk regardless if there is data on disk. In this case the **-X** option of fsmodclass must be on. Otherwise, the expiration will not be allowed.

filename...

One or more files to be processed for copy expirations. Each file path must terminate within a migration-directory hierarchy. If multiple file names are specified, the names must be separated by spaces.

-R *directory*

The directory from which to start the recursive operation. All entries from the specified directory and any subdirectories will be processed. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-H This mirrors the functionality provided by the 'find' command and indicates that symbolic links should not be followed. It is only valid with the "**-R**" option. (Ex. ... "**-HR** *directory*" ...)

-L This mirrors the functionality provided by the 'find' command and indicates that symbolic links should be followed. It is only valid with the "**-R**" option. (Ex. ... "**-LR** *directory*" ...) This is the

default behavior for all forms of this command which includes cases when **-R** is not specified.

-D *directory*
Only entries in the specified directory will be processed. This is not recursive.

-B *batchfilename*
The batch file contains the list of entries to be processed. Each entry must be listed one per line. Do not enclose the entry in quotes, use escape characters or any character not in the actual name. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-S This option selects scheduled processing for automatic expiration of copies based on time since last access of a file, as configured with the **fsmodclass**(1) command. The selection of files to be processed is driven by per-file database rows that are created when a file has had all of its copies created. The command with this option is normally run as a scheduled process, but it can be run manually or in a script. Note that at most one instance of the command is allowed to be running, and the command can take a long time to complete. The scheduled run of the **fsexpcopy**(1) command is configured with the **expcopy** feature of the **fsschedule**(1) command.

Caution: When a copy expiration criteria is added to a policy class, it will not apply to files that have already been stored. The automatic expiration generated by this option will not process those files. To apply the copy expiration criteria to these files you will need to call the following command:

fspolicy -b -y *filesystem mount point* **-C**

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

NOTE

Caution should be used when running the command without the **-S** option. Without the **-S** option, policy settings are ignored and copies are expired immediately. If the setting for the associated policy class is set to expire all copies, then the file could be completely removed from the system and not recoverable. (A user cannot even use the **fsrecover**(1) command to bring a file back.) Please refer to the **-X** option in the **fsmodclass**(1) for the setting that will allow the last copy to be removed.

SEE ALSO

fsclassinfo(1), **fsfileinfo**(1), **fsmodclass**(1), **fspolicy**(1), **fsschedule**(1), **fsstore**(1), **fsxsd**(1)

NAME

fsexport − Export media from the Tertiary Manager system.

SYNOPSIS

fsexport [**-F** *type*] [**-r**] [**-l**] [**-p** *reportfile*] [**-R**] [**-y**] **-m** *mediaID***...**

fsexport [**-F** *type*] [**-r**] [**-l**] [**-p** *reportfile*] [**-s** *drivepool*] [**-v** *drivepool*] **-T** ANTF|LTFS [**-t** *mediatype*] [**-x**] **-b** **-m** *mediaID***...**

fsexport [**-F** *type*] [**-r**] [**-l**] [**-p** *reportfile*] [**-s** *drivepool*] [**-v** *drivepool*] **-T** ANTF|LTFS [**-t** *mediatype*] [**-x**] **-b** [**-H**|**-L**] *pathname***...**

fsexport [**-F** *type*] [**-r**] [**-l**] [**-p** *reportfile*] [**-s** *drivepool*] [**-v** *drivepool*] **-T** ANTF|LTFS [**-t** *mediatype*] [**-x**] **-b** [**-H**|**-L**] **-B** *batchFile*

fsexport [**-F** *type*] [**-r**] [**-l**] [**-p** *reportfile*] [**-s** *drivepool*] [**-v** *drivepool*] **-T** ANTF|LTFS **-d** *destMediaID* **-m** *mediaID***...**

fsexport [**-F** *type*] [**-r**] [**-l**] [**-p** *reportfile*] [**-s** *drivepool*] [**-v** *drivepool*] **-T** ANTF|LTFS **-d** *destMediaID* [**-H**|**-L**] *pathname***...**

fsexport [**-F** *type*] [**-r**] [**-l**] [**-p** *reportfile*] [**-s** *drivepool*] [**-v** *drivepool*] **-T** ANTF|LTFS **-d** *destMediaID* [**-H**|**-L**] **-B** *batchFile*

DESCRIPTION

The **fsexport**(1) command will logically export storage media that contain file data from the Tertiary Manager system. After running the **fsexport**(1) command, the media can be physically removed either by using Media Manager commands or by using the Library Operator Interface (LOI) in the StorNext GUI. Once media are exported from one Tertiary Manager system, they may be imported into another Tertiary Manager system using the **fsimport**(1) command.

As part of the export process, the **fsexport**(1) command will automatically remove any truncated files from disk if the exported media represent all existing copies of those files. Once the export process completes successfully, any exported media cease to be known to the Tertiary Manager system.

The **fsexport**(1) command will generate a manifest file for each media id that is exported. These manifest files are used by **fsimport**(1) to aid in the import process. By default, the manifest files are deposited in the */usr/adic/TSM/internal/manifests* directory. It is the user's responsibility to maintain and clean out any files from this directory as needed. Manifests can be removed either via the SN GUI or via command line.

The Tertiary Manager system supports segmented files on ANTF media but not on LTFS media.

OPTIONS

-m *mediaID...*

A list of one or more media IDs separated by spaces, up to a maximum of 99 media IDs. When used with neither the **-b** nor the **-d** options, the **-m** option specifies the exact list of media to be exported. After the export process completes, a manifest file will be saved to the */usr/adic/TSM/internal/manifests* directory. These manifest files can be used by **fsimport**(1) to speed up the process of importing the media and its contents into another Tertiary Manager system.

When used with the **-b** option, the **-m** option specifies a list of source media, the data for which will be copied to blank media chosen by the Tertiary Manager system. After the copy operation completes, the media containing the newly copied data will be exported. The source media will remain in the Tertiary Manager system.

When used with the **-d** option, the **-m** option specifies a list of source media, the data for which will be copied to the medium specified by the **-d** option. After the copy operation completes, the medium specified by the **-d** option will be exported. The source media will remain in the Tertiary

Manager system.

pathname...
A list of one or more file and/or directory pathnames. This specifies a list of files to be exported. A copy of each file must already reside on storage media to qualify for export. If a directory is specified, all files found in a recursive search of that directory will be exported. Duplicate entries will be ignored.

When used with the **-b** option, file data will be copied from its source media to blank media chosen by the Tertiary Manager system. After the copy operation completes, the media containing the newly copied data will be exported. The source media will remain in the Tertiary Manager system.

When used with the **-d** option, file data will be copied from its source media to the medium specified by the **-d** option. After the copy operation completes, the medium specified by the **-d** option will be exported. The source media will remain in the Tertiary Manager system.

-B *batchFile*
A text file containing a list of file and/or directory pathnames. This batch file specifies a list of files to be exported. Each file or directory entry in the batch file must be listed on its own line. Comments are denoted by lines that begin with the '#' character. A copy of each data file must already reside on storage media to qualify for export. If a directory is specified, all files found in a recursive search of that directory will be exported. Duplicate entries will be ignored.

When used with the **-b** option, file data will be copied from its source media to blank media chosen by the Tertiary Manager system. After the copy operation completes, the media containing the newly copied data will be exported. The source media will remain in the Tertiary Manager system.

When used with the **-d** option, file data will be copied from its source media to the medium specified by the **-d** option. After the copy operation completes, the medium specified by the **-d** option will be exported. The source media will remain in the Tertiary Manager system.

-H This mirrors the functionality provided by the 'find' command and indicates that symbolic links should not be followed when processing a directory or file.

-L This mirrors the functionality provided by the 'find' command and indicates that symbolic links should be followed when processing a directory or file. This is the default behavior for all forms of this command where path names are to be processed.

-b Specifies that the source data is to be copied to blank media. Once the copy operation completes, the media containing the newly copied data will be exported. The media containing the source data will remain in the Tertiary Manager system.

-d *destMediaID*
Specifies the destination media ID to which the source data will be copied. If this medium does not have enough space to hold all the files and fills up, then additional blank media will be chosen by the Tertiary Manager system to continue the copy operation. Once the copy operation completes, all of the destination media will be exported. The media containing the source data will remain in the Tertiary Manager system. When **-d** *destMediaID* is specified, all source media or files must belong to the same policy class and have the same copy id.

-s *drivepool*

Used with the **-b** or **-d** option to specify the source drive pool to be used by the media from which data will be copied during the export process.

-v *drivepool*

Used with the **-b** or **-d** option to specify the destination drive pool to be used by the media to which data will be copied during the export process.

-T ANTF|LTFS

Specifies the media format type that will be used when formatting or selecting media. **ANTF** is the Quantum internal tape format. **LTFS** is the Linear Tape File System specification tape format. A value of **ANTF** will format media with a single partition containing ANTF volume labels. This data partition will store StorNext file data. A value of **LTFS** will format media with two partitions containing LTFS volume labels. The index partition will store LTFS metadata and the data partition will store StorNext file data. The **LTFS** media format type is only applicable to LTO media generation 5 and beyond (e.g. LTO-5, LTO-6, etc). Any attempt to assign **LTFS** to non-LTO media types will result in an error.

Used with the **-b** or **-d** option to specify the data format type for the media to be exported.

-t *mediatype*

Used with the **-b** option to specify the tape media type for the media to be exported. The following tape media types are supported by Tertiary Manager software when **-T ANTF** is used:

LTO
LTOW
3592
T10K

when **-T LTFS** is used then the mediatype must be **LTO**.

-x Used with the **-b** option to specify that each source media shall have its own dedicated destination media to which its data will be copied. In other words, the data from multiple source media will not be consolidated to the same destination media.

-R When a specific list of media is being exported with the **-m** option, the -R option specifies that the export process will automatically remove all files from disk -- whether they are truncated or not -- if the exported media represents the only existing copies of those files.

-y Suppress all prompts and assume a yes response for any prompts that would normally appear.

-r Generate a report of the total number of files per medium that would be exported, but do not actually export any files or media. All validations that would normally be performed are also performed in report mode.

-l Used with the **-r** option to generate a longer report that contains a list file names along with the total number of files that would be exported without actually exporting any files or media.

-p *reportFile*

Used with the **-r** option to specify an output file to which the report is written. The output file will contain media ids, file counts per media id, and optionally (with the **-l** option) the list of file names per media id. All other output will go to stdout. When used with both the **-r** and the **-l** options, the generated report file can essentially serve as an input batch file to be used with the **-B** option.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

NOTES

The Tertiary Manager system supports segmented files on ANTF media but not on LTFS media.

SEE ALSO

fsimport(1)

NAME

fsextlog − Extract the contents of a system log file.

SYNOPSIS

fsextlog *logfilename* [**-t** *starttime* [*endtime*]]

DESCRIPTION

The **fsextlog**(1) command extracts information from any of the Tertiary Manager system logs. The information is returned to *stdout*. If no options are specified, all of the information in the specified log is extracted by default.

The time range option (**-t** *starttime*) extracts only information stored in the log over the time frame specified. If specified, the *starttime* and *endtime* must be before the current time, and the *starttime* must be before the *endtime*. If no *endtime* is specified, *endtime* defaults to current time.

OPTIONS

logfilename

File name of a Tertiary Manager log file - The full file path name does not have to be entered. If preceded by a slash (/) in the command definition, the full path name, starting from the root directory, is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent.

-t *starttime* [*endtime*]

Time range on which to report. The format for the time parameter is MM:DD:hh:mm:ss, where:
MM = Numeric, two-digit month
(default:01 (January)),

The following are optional; defaults are shown:,
DD = Numeric, two-digit day
(range: 01-31, default:01),
hh = Numeric hour
(range: 00-23, default:00),
mm = Numeric minute
(range: 00-59, default:00),
ss = Numeric second
(range: 00-59, default:00)

NAME

fsfilecopy − Copies files for replacement or creating a new copy.

SYNOPSIS

fsfilecopy [-H|-L] -R *directory*|**-D** *directory*|**-B** *batchfilename*| *filename*...
-r |-n *newcopynum* **-d** *destinationmediaID* [**COPY OPTIONS**]

fsfilecopy [-H|-L] -R *directory*|**-D** *directory*|**-B** *batchfilename*| *filename*...
-r |-n *newcopynum* [**-b**] [**-t** *mediatype*] [**COPY OPTIONS**]

COPY OPTIONS

[**-T ANTF**|**LTFS**] [**-G y**|**n**] [**-I**] [**-O** *ops_limit*] [**-c** *copynum*]
[**-s** *sourcedrivepool*] [**-v** *destinationdrivepool*]
[**-e** *encryptiontype*] [**-M** *mkeyname*]

DESCRIPTION

The **fsfilecopy**(1) command will either replace a specific copy or create a new copy of files by duplicating them on another medium. Files located on the original media are copied onto a blank or nonblank medium. By default the primary copy of the file(s) is used unless the **-c** option is specified. If the requested copy is not available the **fsfilecopy**(1) command will fail. Files located on the source media are copied onto blank (**-b**) or nonblank medium. If a nonblank medium (**-d**) is chosen, files from the source medium will be appended to the destination medium. If sufficient room is not available on the destination medium, a message will be returned showing the number of file(s) not copied.

Non-active (deleted) files are not copied. These files remain on the original medium and cannot be accessed or copied to another medium unless the file(s) is recovered using the **fsrecover**(1) command. If the non-active files are no longer wanted, then **fsclean**(1) can be used to remove inactive file versions from the source media.

Execution of the **fsfilecopy** command may be time consuming if a large number of files are specified. The extensive database activity caused by **fsfilecopy** impacts the performance of most other Tertiary Manager commands. It is recommended that **fsfilecopy** be run during a time of little or no use of Tertiary Manager software if a large number of files need to be copied.

Copying files from S3 Glacier or Azure Archive media is not supported.

Replacing a Copy

The **-r** option is used to replace a file or files on a medium by moving data to a different medium. Files located on the source medium are copied onto blank (**-b**) or nonblank medium.

Creating a new Copy

The **-n** option is used to create a new copy of files providing the specified new copy does not exceed the policy class max copies setting. The new copy value can be no larger than one more than the current default copies setting for the file. For example if a file has default copies of 2 then the new copy value can be no larger than 3, so copies can not be skipped.

The new copy option will only make copies for files that are truncated. If a file is not truncated, then it will be updated such that it will be a storage candidate for the specified new copy. Then it is left for the storage policies to store.

OPTIONS

-r Copies specified file(s) to a medium and deletes all specified file(s) on the original medium.

-n *newcopynum*
Creates a new copy of the specified file(s) on a medium or adds file(s) as storage candidates. The original medium remains unchanged.

-c *copynum*
Specify file copy number to replace or to generate the new copy. This option affects all specified files. Without this option the primary copy of the file(s) is used.

-b Specify blank media for destination copy. This option is not supported for Object Storage media.

-d *destinationmediaID*
Destination media identifier. Allows user to specify where file(s) will be copied.

-R *directory*
The directory from which to start the copy operation. All files from the specified directory and any subdirectories will be copied. Depending upon the number of files in the directory and subdirectories, running this option may use extensive Tertiary Manager resources.

-H This mirrors the functionality provided by the 'find' command and indicates that symbolic links should not be followed. It is only valid with the "**-R**" option. (Ex. ... "**-HR** *directory*" ...)

-L This mirrors the functionality provided by the 'find' command and indicates that symbolic links should be followed. It is only valid with the "**-R**" option. (Ex. ... "**-LR** *directory*" ...) This is the default behavior for all forms of this command which includes cases when **-R** is not specified.

-D *directory*
Only entries in the specified directory will be processed. This is not recursive.

-B *batchfilename*
The batch file contains a list of files to be copied. File names should be listed one per line. Each line in the file is read as a string of characters, without interpretation, up to the line terminator. You should not enclose the file name in quotes, use escape characters or any character not in the actual file name.

filename...
The list of files to be copied. The full file path name does not have to be entered. The file name will be resolved from the current working directory. If preceded by a slash (/) in the command definition, the full path name starting from the root directory is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent. Multiple filenames must be separated by spaces

-s *sourcedrivepool*
This option will copy files using only drives for the source media that are associated with the specified drive pool. Media Manager drive pool group from which the source drive will be selected when copying the specified files. The drive pool must be defined in Media Manager software. If the **-s** option is not used, the default drive pool group will be specified by the policy class definition. The special "_" character is permitted to identify the drive pool group.

-t *mediatype*
Defines the type of medium to be copied to. Depending on the platforms used, the following media types are supported by Tertiary Manager software:

AWS
AZURE
LATTUS
GOOGLE
GOOGLES3
S3
S3COMPAT
LTO
LTOW
3592
T10K
SDISK

If **-t** is not specified, the policy class definition will be used.

-v *destinationdrivepool*
This option will copy files using only drives for the destination media that are associated with the specified drive pool. Media Manager drive pool group from which the destination drive will be se-

lected when copying the specified files. The drive pool must be defined in Media Manager software. If the **-v** option is not used, the default drive pool group will be specified by the policy class definition. The special "_" character is permitted to identify the drive pool group.

-T ANTF|LTFS

Specifies the media format type that will be used when formatting or selecting media. **ANTF** is the Quantum internal tape format. **LTFS** is the Linear Tape File System specification tape format. A value of **ANTF** will format media with a single partition containing ANTF volume labels. This data partition will store StorNext file data. A value of **LTFS** will format media with two partitions containing LTFS volume labels. The index partition will store LTFS metadata and the data partition will store StorNext file data. The **LTFS** media format type is only applicable to LTO media generation 5 and beyond (e.g. LTO-5, LTO-6, etc). Any attempt to assign **LTFS** to non-LTO media types will result in an error.

If the **-T** option is not specified, the default will be determined as follows:

A media type of Object Storage will default to **NONE**.
A media type that does not support **LTFS** will default to **ANTF**.
A media type that supports **LTFS** will default to the policy class copy definition.

-G y|n Generate and maintain a checksum for each copied file. When this option is enabled, a checksum will be generated as each file is written (stored) to the new media. Policy class settings are not used when this option is not specified, so it defaults to 'n'. Any file being copied that already had a checksum generated will ignore this option and continue to use the existing value.

-I This option can be used to ignore the cleanup of the Object Storage when the source is an object media. If the source is not an object media then this option is ignored. When this option is provided, the cleanup of the Tertiary Manager database will still take place but there will be no attempt to delete the objects from the Object Storage. This option could be useful when migrating data off of an Object Storage appliance that will no longer be used.

-O *ops_limit*

This provides a means of throttling the request by limiting the number of I/O operations, requests, that are generated. The default for this is 10.

For requests that don't involve tape devices, limiting the number of I/O requests will also restrict the number of streams used. The number of streams that can be used will be the lesser of the *ops_limit* value or the max streams defined for the device. The stream limit is applied to each source and destination device so *ops_limit* number of streams will be used for each. For example: if **-O 3** is specified then 6 streams would be used; 3 for the source and 3 for the destination. However, if the max streams defined for one of the devices is smaller, then the request would be further restricted by that. Using the same example, if the source device was configured with max streams of 2, then only 2 streams would be utilized for both source and destination devices.

When setting this value for non-tape devices keep in mind that **fsfilecopy**(1) requests run at a lower priority than store and retrieve requests so they will only use available streams when there are no store/retrieve requests requiring those resources.

For requests using tape devices this will be the number of queued I/O requests. Tape I/O requests are processed sequentially so there is no benefit in adjusting this value. It is recommended to use the default setting.

-e *encryptiontype*

The encryption algorithm to be applied to the content of files stored into an Object Storage system. This option is ignored for all other media types. Currently, the following encryption types are supported:

none	No encryption
SSE_S3	Server-side AES256 encryption with S3-managed keys

SSE_KMS Server-side AES256 encryption using the AWS Key Management Service (KMS)

For the **SSE_KMS** encryption type, the default customer master key (CMK) can be used for encryption or a CMK can be created using the AWS Key Management Service and specified with the **-M** option.

If this option is not specified, the encryption type will be set to **none**.

-M *mkeyname*

The Master Key name that has been created using the AWS Key Management Service and requested for server-side encryption.

SEE ALSO

fsclean(1) **fsmedout**(1), **fsmedinfo**(1), **fsrecover**(1), **fschmedstate**(1), **fskey**(1)

NAME

fsfileinfo − Generate a report about files known to the Tertiary Manager.

SYNOPSIS

fsfileinfo [**-c**] [**-e**] [**-o**] [**-u**] [**-F** *type*] *filename...*

fsfileinfo [**-c**] [**-e**] [**-o**] [**-u**] [**-F** *type*] **[-H|-L]**-R *directory*

fsfileinfo [**-c**] [**-e**] [**-o**] [**-u**] [**-F** *type*] **-D** *directory*

fsfileinfo [**-c**] [**-e**] [**-o**] [**-u**] [**-F** *type*] **-B** *batchfilename*

DESCRIPTION

The **fsfileinfo**(1) command reports the current location(s) of files, whether on disk, on a particular medium, or not in the SNSM system. It also shows file attribute information.

REPORT STATUS

The processing parameters listed by the **fsfileinfo**(1) command are as follows:

Filename	The full path name where the file is located
Stored Name	The full path name of the file when it was stored. It will be different than *Filename* when the file has been renamed.
Last Modification	The date and time the file was last modified
Owner	The owner of the file
Group	The group to which the file belongs
Access	The file permission for the file
Class	The policy class governing the file
Store	How storage policies operate on the file. Values are: **MINTIME**, **NEVER**, **EXCLUDE**
Trunc	How truncation policies operate on the file. Values are: **MINTIME**, **IMMEDIATE**, **NEVER**, **EXCLUDE**
Reloc	How relocation policies operate on the file. Values are: **MINTIME**, **NEVER**
Clean Database	Indicates if database entries are cleaned when file is removed
Affinity	The disk affinity on which the file resides
Media	The media identifier and media copy number; (1) is the primary copy, (2) is the secondary, etc.
File size	The size of the file in bytes
Location	The location where the file data currently resides. Values are: **DISK**, **DISK AND ARCHIVE**, **ARCHIVE** (where **ARCHIVE** indicates TAPE, SDISK, or Object Storage media).
Existing Copies	The number of media copies that currently exist for the file
Target Copies	The number of media copies that should exist for the file
Expired Copies	The number of expired media copies along with the list of copy numbers that have expired and no longer exist for the file.
Existing Stub (KB)	The existing stub size (in kilobytes). This is the actual stub size if the file is only located on storage media, otherwise it is 'n/a'.
Target Stub (KB)	The truncation stub size (in kilobytes). This value is used to determine the number of bytes to leave on disk when the file is truncated. It will be the minimum number of bytes left on disk (the value is rounded up to a multiple of the file system block size).

Alternate Class	The policy class which is used instead of the primary class when policies are run
Checksum	**Y**|**N** to show whether a checksum value has been generated for the file
Encryption	**Y**|**N** to show whether encryption was applied to the Object Storage copy of the file
Alt Store Copy	**Disabled**|**Enabled** to show whether the file is eligible for an Alternate Store Location copy. If **Enabled**, additional information will be displayed to indicate whether a copy of the file is **Needed** or **Exists**.
Object Ids	**Y**|**N** to show whether the file has any objects stored to the Object Storage

OPTIONS

filename...

The path name of at least one file is required. The full file path name does not have to be entered. The file name will be resolved from the current working directory. If preceded by a slash (/) in the command definition, the full path name starting from the root directory is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent. Multiple filenames must be separated by spaces.

-R *directory*

The directory from which to start the recursive operation. All entries from the specified directory and any subdirectories will be processed.

-H This mirrors the functionality provided by the 'find' command and indicates that symbolic links should not be followed. It is only valid with the "**-R**" option. (Ex. ... "**-HR** *directory*" ...)

-L This mirrors the functionality provided by the 'find' command and indicates that symbolic links should be followed. It is only valid with the "**-R**" option. (Ex. ... "**-LR** *directory*" ...) This is the default behavior for all forms of this command which includes cases when **-R** is not specified.

-D *directory*

Only entries in the specified directory will be processed. This is not recursive.

-B *batchfilename*

The batch file contains the list of entries to be processed. Each entry must be listed one per line. Do not enclose the entry in quotes, use escape characters or any character not in the actual name.

-c If checksum was turned on for the file when stored and this option is specified, the checksum value generated for the file will be displayed.

-e If encryption was turned on for the file when stored and this option is specified, the encryption type for the file will be displayed.

-o If the file has a copy or more stored to Object Storage, then the object ids will be displayed. Offset information will also be displayed for each object id.

-u If the file has a copy or more stored to Object Storage, then the object URLs will be displayed. Object Storage configuration information is also displayed.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to

JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsmedinfo(1), **fsclassinfo**(1), **fsxsd**(1)

NAME

fsfiletapeloc − Generate a report about a file's tape copy location.

SYNOPSIS

fsfiletapeloc [**-c** *copy*] [**-F** *type*] *filename*

DESCRIPTION

The **fsfiletapeloc**(1) command reports the location information of the file's on-tape copies. The report will list all the segments belonging to the specified file copy.

REPORT STATUS

The following information is reported for each copy segment:

segnum	The segment number
media ID	The media ID where the segment resides
library ID	The library ID that the media belongs to
format	The format type of the tape (XML/JSON output only). Valid values: **ANTF**, **LTFS**, **AXF**, **DL64K**, **DL512K**, **UNKNOWN**
start blk	The block number where the segment starts
offset	The offset from the starting block where the segment starts. Note, the offset will always be zero for LTFS formatted tapes.
segsize	The length of the segment
blk size	The I/O block size for the media.

OPTIONS

filename

The path name of the file is required. The full file path name does not have to be entered. The file name will be resolved from the current working directory. If preceded by a slash (/) in the command definition, the full path name starting from the root directory is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent.

-c *copy* The copy id to generate report for. If not specified, the information for the primary copy will be reported.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

NOTES

This command is only valid for file copies on tape media. It will not report anything for file copies on storage disks or Object Storage.

SEE ALSO

fsfileinfo(1)

NAME

fsforeignstatus − Get status on the foreign background migration

SYNOPSIS

fsforeignstatus [**-errored**] [**-waiting**] [**-trashed**]

DESCRIPTION

This will provide a report on the foreign files background migration, on a per file system basis. By default, just the file counts for each category are shown.

Each one of the optional command line parameters will provide a full remote file listing for that category. The optional command line parameters can be abbreviated.

Errored files should be addressed. It is possible that they might never be migrated so they should be cleaned up. For example, if the file is removed from or changed on the foreign file system it will not be migrated. That file should be manually moved over or destroyed so that the background migration does not try to continually retrieve that file.

NAME

fsformat − Formats blank media for use.

SYNOPSIS

fsformat [**-f**] [**-T ANTF**|**LTFS**] *mediaID...*

fsformat -c *class* [**-q** *quantity*] [**-f**] [**-T ANTF**|**LTFS**]

fsformat -q *quantity* [**-f**] [**-T ANTF**|**LTFS**]

DESCRIPTION

The **fsformat**(1) command provides the capability to format one or more media based on the media identifier, policy class, or quantity.

Media are considered formatted after the volume label is written and can be formatted by class or by specifying a specific media identifier.

The **-q** option specifies quantity of media to be formatted in a class or in the general pool of blank media. If the **-q** option is not specified for **fsformat**(1) class, all blanks associated with the policy class specified will be formatted.

If format processing, either manual or automatic, failed for a particular medium, that medium is marked for ejection from the Quantum storage subsystem and is no longer considered as available for file storage. The user would have to change the state of the medium before the next format of the medium can be attempted.

OPTIONS

mediaID...
: One or more media identifier(s) to be formatted. Multiple media identifiers must be separated by spaces.

-c *class* Specifies the policy class the medium is to be associated with when formatted.

-q *quantity*
: Specified quantity of media to be formatted.

-f Forces the formatting of blank media that was previously used or formatted. Typically this option should not be needed, but may be required when introducing media that were used in another SNSM system. Format failure of a pre-formatted medium without the force option prevents accidentally formatting a medium containing data.

-T ANTF|**LTFS**
: Specifies the media format type that will be used when formatting or selecting media. **ANTF** is the Quantum internal tape format. **LTFS** is the Linear Tape File System specification tape format. A value of **ANTF** will format media with a single partition containing ANTF volume labels. This data partition will store StorNext file data. A value of **LTFS** will format media with two partitions containing LTFS volume labels. The index partition will store LTFS metadata and the data partition will store StorNext file data. The **LTFS** media format type is only applicable to LTO media generation 5 and beyond (e.g. LTO-5, LTO-6, etc). Any attempt to assign **LTFS** to non-LTO media types will result in an error.

  If the **-T** option is not specified, the default will be specified by the system parameter CLASS_MEDIA_FORMAT.

NOTES

This command is not valid for storage disks.

SEE ALSO

fsmedin(1)

NAME

fsgetclasses − Report all policy classes with association points in a file system.

SYNOPSIS

fsgetclasses [**-F** *type*] *filesystem*

DESCRIPTION

The **fsgetclasses**(1) command lists all policy classes that have association points within a specified file system. Policy class association points are added using **fsaddrelation**(1).

REPORT STATUS

The report produces the following information for each policy class that has at least one association in the file system:

File System Name
: The name of the mount point

Class
: The policy class name

OPTIONS

filesystem
: Name of the file system for which to generate policy class association report

-F *type*
: Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

  TEXT is the "legacy" textual format.

  JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

SEE ALSO

fsdirclass(1), **fsaddrelation**(1), **fsclassinfo**(1)

NAME

fsgetforeign − A data copying utility used by the foreign file system migration feature.

DESCRIPTION

As part of the foreign file system migration feature, the **fsgetforeign**(1) utility script is executed automatically by the Tertiary Manager system to retrieve file data from a foreign file system. The script uses several sysparms (configuration variables) that can be modified to affect its operation, including timeouts, debugging, etc.. The script can be examined to find particulars and descriptions.

WARNINGS

Users should not modify this script nor run this script directly unless directed by — or under the supervision of — Quantum technical assistance.

If this script is modified, ensure that it properly handles special characters such as ` and **$.**

NAME

fshistrpt − Report history of hardware component state changes or media status changes.

SYNOPSIS

fshistrpt -h [*componentID...*] [**-t** *starttime* [*endtime*]] [**-f** *logfilename*]

fshistrpt -m *mediaID...* [**-t** *starttime* [*endtime*]] [**-f** *logfilename*]

DESCRIPTION

The **fshistrpt**(1) command lists the state changes of specified media or hardware components. The **-h** or **-m** option is required to indicate the report type.

The information gathered by **fshistrpt**(1) is maintained in the Tertiary Manager log files. The historical data for the media status is maintained in the *$FS_HOME/logs/history/hist_03* file. The historical data for components is logged to the *$FS_HOME/logs/history/hist_04* file. These are the default log files used, however a different log file can be specified using the **-f** option.

State changes that may occur for components are *on* and *off*. These changes occur by system administrator execution of the **fschstate**(1) command or internally by Tertiary Manager software. Valid state changes for media are *unavailable*, *available*, *protect*, *unprotect*, *unsuspect*, and *unmark*. Media state changes can occur using the **fschmedstate**(1) command or can be initiated by Tertiary Manager software.

The time range option (**-t** *starttime*) extracts only information over the time frame specified. If specified, the *starttime* and *endtime* must be before the current time, and the *starttime* must be before the *endtime*. If no *endtime* is specified, *endtime* defaults to current time.

OPTIONS

-h Historical report of component state changes.

componentID...

One or more component identifiers from which to give a historical report of the state changes. Because the component alias is configurable, the component identifier is given to obtain complete historical information.

-m *mediaID...*

One or more media identifiers from which to obtain a historical report of media status changes. Multiple media identifiers must be separated by spaces.

-t *starttime* [*endtime*]

The time range in which to print entries. If no end time is specified, the current time is used. The format for the time parameter is MM:DD:hh:mm:ss, where:

MM = Numeric, two-digit month
(default:01 (January)),

The following are optional; defaults are shown:,
DD = Numeric, two-digit day
(range: 01-31, default:01),
hh = Numeric hour
(range: 00-23, default:00),
mm = Numeric minute
(range: 00-59, default:00),
ss = Numeric second
(range: 00-59, default:00)

-f *logfilename*

File name of a Tertiary Manager log file - The full file path name does not have to be entered. If preceded by a slash (/) in the command definition, the full path name, starting from the root directory, is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent.

SEE ALSO

fschstate(1), **fschmedstate**(1)

NAME

fsimport − Import media and ingest content into the Tertiary Manager system

SYNOPSIS

fsimport [**OPTIONS**] -m *mediaid* [-m *mediaid***]...** [*filename* **...**]

fsimport [**OPTIONS**] -d *device* [*filename* **...**]

fsimport [**OPTIONS**] -i *manifest* [-i *manifest***]...** [*filename* **...**]

fsimport -S

DESCRIPTION

fsimport(1) will import media and ingest content into the Tertiary Manager system. It also provides administrative operations such as mounting and dismounting media, mounting and unmounting LTFS file systems on LTFS formatted media, and listing the contents of media. While multiple media can be specified on the command line, **fsimport**(1) will process each medium one at a time in the order submitted.

For **fsimport**(1) to operate correctly, the specified media must have already been entered into the Media Manager system. The **fsimport**(1) command will automatically enter media as needed from the Media Manager into the Tertiary Manager. Media will be entered as formatted, write-protected media via the **fsmedin**(1) command.

The Tertiary Manager system supports segmented files on ANTF media but not on LTFS media. If segmented files that span multiple ANTF media are being imported through the StorNext GUI, all media must be imported from the same tape library.

The **fsimport**(1) command supports ingesting content from media through either a "file ingest" operation or a "media ingest" operation.

A "file ingest" operation copies files from the specified media into a destination directory and then removes the media from the system. This mode of operation is available only for LTFS media. The **fsimport**(1) command will mount each medium in a drive and FUSE mount the medium as a file system. It will then copy the content from the FUSE file system mount point to the destination directory. For "file ingest", the Tertiary Manager attaches no dependencies to the specified media, allowing each medium to be removed from the system via **fsmedout**(1) after it is processed. Once an entire "file ingest" operation completes, the files are eligible to be stored to different media according to configured Tertiary Manager policies.

A "media ingest" operation ingests each medium’s content and permanently imports the medium itself into the Tertiary Manager system. The **fsimport**(1) command will mount each medium in a drive and populate the database with knowledge of the files on the medium. The destination directory is then populated with truncated files. After "media ingest" completes, the files may be retrieved from the imported media at the user’s discretion.

When **fsimport**(1) finishes, it will by default dismount media that it mounted and unmount LTFS file systems that it mounted during its operation. Also when **fsimport**(1) finishes a "file ingest" operation, it will by default issue **fsmedout**(1) on any media that it entered via **fsmedin**(1).

While the **fsimport**(1) command can be invoked without specifying a *manifest* file for each medium, it is highly recommended that a *manifest* file always be used when one is available. If a *manifest* file is not specified, **fsimport**(1) will resort to scanning the entire medium to gather the necessary information to perform a "media ingest" operation. If **fsimport**(1) has to scan the medium for a "media ingest" operation, 1) it will not be able to import file checksums, 2) it will likely take longer to complete its operations, and 3) it will not be able to tell the user which media ids are needed to completely ingest files that span media. Additionally, when the user merely needs a listing of the contents of each media, that list can be generated much faster when a *manifest* file is specified.

If the **fsimport**(1) process fails during "media ingest", any database entries that were inserted will be removed by the next scheduled **fsclean**(1) process.

If certain content fails to ingest in an initial run of **fsimport**(1), the user can make another attempt to ingest the content by reinvoking **fsimport**(1) using the same destination directory. Any files that were already successfully populated in the destination directory will be skipped.

The **fsimport**(1) command can perform other administrative tasks such as mount and dismount media in drives, mount and unmount LTFS file systems, and list media contents. To perform these operations, specify an *ingesttype* type of "none".

The **fsimport**(1) command logs to */usr/adic/TSM/logs/tac/fsimport.log*. It also logs status information of the current operation to */usr/adic/TSM/internal/status_dir/fsimport.status*.

OPTIONS

filename...

Ingest only the fully qualified files or directories specified in the *filename* list. A *filename* of "--" will cause **fsimport**(1) to stop processing command line options and treat all remaining command line arguments as files or directories to be ingested.

-m *mediaid*, **--mediaid** *mediaid*

Operate on the medium *mediaid*. If the medium is not known to the Tertiary Manager system, it will be entered via *fsmedin -i mediaid*

-d *device*, **--devpath** *device*

Operate on the medium mounted in the drive *device*. **fsimport**(1) will terminate if there is no medium mounted at *device*.

-i *manifest*, **--importfile** *manifest*

Import the medium identified in the *manifest* file.

-S, **--status**

Get status for the currently running import or the results of the last import.

-p *dir*, **--destdir** *dir*

Ingest data into the destination directory *dir*. The destination directory must be associated with a relation point for "media ingest". For an *ingesttype* of "files", the destination directory is not required to be associated with a relation point. However, files ingested to a directory that is not associated with a relation point will not be managed by the Tertiary Manager system.

-s *dir*, **--strippath** *dir*

Strip the *dir* directory prefix off of the source content path when ingesting data. Only paths that begin with the directory prefix *dir* will be ingested. This can also be used to do a selective ingest of content, as paths not beginning with the path *dir* will be excluded from the ingest operation. Source content paths will have the *dir* prefix removed when ingested into the destination directory. If **--strippath** is used, then any supplied *filename* on the command line or any supplied file or directory in the *batchfile* must start with the strippath *dir* directory.

-v *drivepool*, **--drivepool** *drivepool*

Mount media in the specified *drivepool*.

-B *batchfile*, **--batchfile** *batchfile*
Ingest only the fully qualified files and directories listed in the file *batchfile*.

-t *type*, **--ingesttype** *type*
Identify how to ingest media content. Valid options are "files", "media", or "none". If the *ingesttype* is "files" or "media", then the *destdir* option must also be specified. The *ingesttype* of "files" is not supported for ANTF media at this time.

-T *format*, **--format** *format*
Specifies the expected media *format* type of the media to be imported. Valid options are **LTFS**, **ANTF**, and **AUTODETECT**. Defaults to **AUTODETECT**.

--copynum *num*
Identify the copy number for files when ingesting media. Defaults to copy 1.

--import_uid *uid*
Set ingested files to owner *uid* where *uid* is the numeric user ID. For LTFS, this defaults to 0. For ANTF, this defaults to the value scanned from the media.

--import_gid *gid*
Set ingested files to group *gid* where *gid* is the numeric group ID. For LTFS, this defaults to 0. For ANTF, this defaults to the value scanned from the media.

--import_mode *mode*
Set ingested file permissions to *mode* where *mode* is an octal numeric value from 0000 to 7777. For LTFS, this defaults to 0440. For ANTF, this defaults to the value scanned from the media.

--dryrun
Show the ingest operations that would take place without actually ingesting the media content.

--fusemount
FUSE mount the LTFS file system. This is implicitly set for file ingest from LTFS media. This option can be used to mount a LTFS medium and list its contents in order to produce a listing of files for selective file ingest.

--fuseumount
FUSE unmount the LTFS file system when done. This is implicitly set if the LTFS file system was not FUSE mounted prior to calling **fsimport**(1).

--nofuseumount
Do not unmount LTFS file system when done.

--dismount
Dismount the medium from drive with **fsdismount**(1) when done. This is implicitly set if the requested medium is not mounted in a drive prior to calling **fsimport**(1) , or if the medium was dismounted and remounted.

--nodismount
Do not dismount the medium from drive when done.

--medout
Call **fsmedout**(1) when done. This is implicitly set if the medium was not entered with **fsmedin**(1) prior to calling **fsimport**(1). This option is only valid for file ingest.

--nomedout
Do not call **fsmedout**(1) when done.

--recoverfiles
Recover files with **fsrecover**(1) during media ingest. The default behavior is to recover files during media ingest.

--norecoverfiles
Do not recover files during media ingest.

--noschemacheck
Do not validate the database schema version of the manifest file against the database schema version on the destination system. Use this option only at the direction of Quantum technical support.

The **fsimport** command verifies that the database schema version in the manifest file matches the database schema version of the system where the tapes are being imported. If the schema versions do not match, **fsimport** will fail with an error message as in the following example:

Manifest file 'exportManifest.000125' schema version '5.3.0' does not match database schema version '6.3.0' on this system.

The **--noschemacheck** option bypasses the schema check. This option is intended for situations in which the schemas differ, but the differences are in database tables not used by **fsimport**. If the schemas for the database tables listed below are the same in the source and target systems, the **--noschemacheck** option can be safely used to bypass the **fsimport** schema check.

tdlmdb.mediacriteria
tmdb.dirpath
tmdb.filecomp
tmdb.filecomp_ltfs
tmdb.fileinfo
tmdb.mediadir
tmdb.rmdirinfo
tmdb.rmfileinfo

--norequiredmedia
Do not require that all media be specified to account for files that span media. This allows for files that do not span media to be successfully ingested without requiring other media to be specified. However, this does mean that any files that span media will not be successfully ingested unless all their media are specified.

-l, **--list** List files on the medium.

-o *file*, **--outfile** *file*
Generate a manifest *file* that can be used for importing media with the **--importfile** option.

--tmpdir *dir*
Use *dir* as a temporary working directory. Use this option if the default working directory is low on disk space. The default working directory is */usr/adic/TSM/internal/fsimport*.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

--verbose
Display extra output.

--debug
Enable extra debug messages.

-h, **--help**
Print a usage message and exit.

FILES

/usr/adic/TSM/internal/fsimport/<mediaid>/medinfo.cache
/usr/adic/TSM/internal/fsimport/<mediaid>/<mediaid>.antf_raw_scan
/usr/adic/TSM/internal/fsimport/<mediaid>/<mediaid>.ltfs_index
/usr/adic/TSM/internal/fsimport/<mediaid>/<mediaid>.scan
/usr/adic/TSM/internal/fsimport/<mediaid>/mnt/
/usr/adic/TSM/internal/locks/fsimport.lf
/usr/adic/TSM/internal/status_dir/fsimport.status
/usr/adic/TSM/logs/tac/fsimport.log

EXAMPLES

Example 1

Perform a "media ingest" operation using a manifest file for medium 000076 and strip off the namespace of the ingested files:

```
fsimport -t media -i exportManifest.000076 -p /stornext/snfs1/import3 \
         -s /stornext/orig_fs1/dir1
```

Example 2

List the files on medium 000077 and save the output to a file:

```
fsimport -l -i exportManifest.000077 > batchfile
```

Example 3

Modify the output of the list option and use it as a batchfile for selective content ingest as part of a "media ingest" operation:

```
fsimport -t media -i exportManifest.000077 -p /stornext/snfs1/import4 \
         -B batchfile
```

Example 4

Perform a "file ingest" operation without a manifest for LTFS medium 000099:

```
fsimport -t files -m 000099 -p /stornext/snfs1/import1
```

Example 5

Assuming LTFS medium 000100 has previously been entered into the Tertiary Manager system, perform a "file ingest" operation on it and remove it from the system when done:

```
fsimport -t files -m 000100 -p /stornext/snfs1/import2 --medout
```

Example 6

Remove an LTFS medium from the system following a "file ingest" operation in which the medium and LTFS file system were left mounted:

```
fsimport -m 000101 --fuseumount --dismount --medout
```

Example 7

Check the status of a running **fsimport**(1) process:

```
fsimport -S
```

NOTES

The *ingesttype* of "files" is not supported for ANTF media at this time.

The Tertiary Manager system supports segmented files on ANTF media but not on LTFS media. If segmented files that span multiple ANTF media are being imported through the StorNext GUI, all media must be imported from the same tape library.

SEE ALSO

fsdismount(1), **fsexport**(1), **fsmedin**(1), **fsmedout**(1), **fsmount**(1), **fsrecover**(1), **fsclean**(1), **ltfs**(8)

NAME

fsimportnamespace − take a dumped foreign server namespace and import the namespace into SNMS

SYNOPSIS

fsimportnamespace -relationPoint *dir* **-policy** *policy* **-namespaceFile** *file* **-hostname** *host* [**-continue**] [**-bypassdb**]

DESCRIPTION

This program takes a namespace dump file of a foreign server from the output of **fsdumpnamespace**(1). The entire namespace will be recreated in a StorNext filesystem. After the program completes the new namespace can be used like a normal StorNext filesystem. As files are used they will be migrated over from the foreign system as necessary. Also, there will be a background process that will migrate the unused files from the foreign server.

Please be aware that the import can take a long time to run. On a fast machine, using the **-bypass** option, it can take about one hour per 1.25 million files and directories. However, times vary depending on the hardware being used.

As this program runs it will build up temporary database files that will be loaded into the database after the namespace is created on disk. These temporary database files will be stored one directory above the relation point. It is recommended that they be archived after the import completes successfully.

Also, please note that the database must have enough disk space available for the import. The size required can vary greatly, but estimate .75 gigabytes per million files and directories imported. The space must be available on the filesystem that contains the StorNext installation.

OPTIONS

Options can be abbreviated to any length.

−relationPoint *dir*
: The relation point for the imported namespace. The relation point directory must be empty, unless the continue option is present.

−policy *policy*
: The policy class for the imported namespace.

−namespaceFile *file*
: The dumped namespace information *file* (*dump.txt*, usually) from the output of **fsdumpnamespace**(1).

−hostname *host*
: The foreign server's hostname.

−continue
: This option should only be used if a previous import ran for some time and then was interrupted for some reason. This option will cause the program to pick up the import at the point where the previous import stopped. It should only be used if the previous import ran for a significant amount of time. Continue mode can only be used if the file system has not been modified in any way since the previous import.

−bypassdb
: This option can be used to delay the loading of the database files after import. Using this option and then running the program indicated at the completion of **fsimportnamespace**(1) will be faster that running without this option. It is recommended to not use this option unless working directly with Quantum technical assistance personnel. To use this option the following conditions must be met:
- The Storage Manager software must be shut down.
- Backups must be performed before running **fsimportnamespace**(1)
- Backups must be performed after completing the database loading via another script (the location of which is shown after **fsimportnamespace**(1) completes).

RESTRICTIONS

This script must be run as root.

Hard links will be imported as individual files, not links.

Files with special characters will be imported with question marks replacing the special characters, but they will get errors when they are migrated. They will show up as errors in the **fsforeignstatus**(1) output and should be cleaned up accordingly.

SEE ALSO

fsdumpnamespace(1), **fsforeignstatus**(1)

NAME

fsimport_diva − convert DIVA metadata to StorNext metadata import files

SYNOPSIS

fsimport_diva [**-h**] -f *DIVA Metadata file* -l *Tape Group file* -m *Mount Point Directory* [**-s** *Media List File* **]** [**-t** *Seconds* **]**

DESCRIPTION

fsimport_diva(1) translates the comma-separated-values file (CSV) output from a query on a DIVA database into a collection of CSV files for import into StorNext Storage Manager metadata. Each DIVA object corresponds to a StorNext directory containing the object's component files. The pathname to the object directory is as follows:

```
/file_system_mount_point/relation_point/base_directory/tape_group_name\
/year/month/day/object_name.object_category
```

There is at least one file per object, but there can be a hierarchy of directories and files in a single object. After these CSV files have been imported, the hierarchy of StorNext directories and file stubs is created with the **fsrecover**(1) command so that file contents can be retrieved from DIVA tape media. The steps for getting the DIVA media into Storage Manager are documented elsewhere.

OPTIONS

-f *DIVA Metadata file*
: Specifies the DIVA metadata-query CSV file.

-l *Tape Group file*
: Specifies the CSV file of DIVA tape groups to import.

-m *Mount Point Directory*
: Specifies the path segment from the root directory to the file-system mount point directory.

-s *Media List File*
: Optionally specifies a file containing the set of bar codes in StorNext, one per line. These will be checked as DIVA media are imported to ensure that there are no conflicts between the two sets of media. A warning will be printed, and the **fsimport_diva**(1) command will quit without importing metadata if there is a conflict. No bar-code conflict checking is done if this is not specified.

-t *Seconds*
: Specifies the Linux epoch value in seconds to be applied to the First, Last, ACC, and Intro fields in the Storage Manager *mediadir* table. The current time is used if this is not specified.

TAPE GROUP FILE

The values per row in the Tape Group File are as follows:

Tape Group Name
: This matches the ON_MEDIA_NAME field from the dp_object_instances table in DIVA. Rows in the DIVA Metadata file are imported if they match one of the specified tape-group names. Otherwise, they are ignored, and can be imported in a subsequent run. A tape group must never be imported more than once.

  The set of all tape-group names can be displayed with the following query in the DIVA database:

```
select MD_MEDIA_NAME from dp_media;
```

StorNext Class ID
: Each tape group can be associated with an existing Storage Manager class policy. A policy can be used by more than one tape group.

Relation-Point directory sub path
: The path below the file-system mount point to the relation-point directory.

Base directory sub path

The path below the relation-point directory to the DIVA object directories.

QUERY

The following query on DIVA tables produces the necessary input file:

```
select
  ON_MEDIA_MD_ID tgId,
  AO_ID aoId,
  OC_ID compId,
  ON_ID onId,
  TE_ID elemId,
  ON_MEDIA_NAME tapeGrp,
  OC_COMPONENT_ORDER_NUMBER cOrd,
  ON_INSTANCE_ORDER_NUMBER iOrd,
  TE_ELEMENT_ORDER_NUMBER eOrd,
  unix_timestamp(AO_DATE_ARCHIVE) date,
  concat_ws('\',
    ON_MEDIA_NAME,
    year(AO_DATE_ARCHIVE),
    month(AO_DATE_ARCHIVE),
    day(AO_DATE_ARCHIVE),
    concat_ws('.',
      AO_OBJECT_NAME,
      AO_CATEGORY)
  ) pathname,
  OC_COMPONENT_NAME compNm,
  CH_CHECKSUM_VALUE cksum,
  TE_BEGIN_POS begin,
  TE_END_POSITION end,
  TE_COMPONENT_STOP_POSITION stopPos,
  TA_BARCODE barcode,
  TA_BLOCK_SIZE blkSz,
  TP_MEDIA_TYPE,
  TA_FORMAT fmt
from
  dp_object_instances
  join dp_archived_objects on ON_OBJECT_AO_ID = ao_id
  join dp_tape_instn_cmpt_elems on ON_ID = TE_INSTANCE_ON_ID
  join dp_object_components on TE_COMPONENT_OC_ID = OC_ID
  left join dp_checksums on AO_ID = CH_OBJECT_AO_ID and CH_COMPONENT_NAME =
OC_COMPONENT_NAME
  join dp_tapes on TE_TAPE_TA_ID = TA_ID
  join dp_tape_properties on TA_MEDIA_TYPE_TP_ID = TP_ID
where
  ON_SPACE = 'T'
order by
  ON_MEDIA_NAME,
  ao_id,
  OC_COMPONENT_ORDER_NUMBER,
  ON_INSTANCE_ORDER_NUMBER,
  TE_ELEMENT_ORDER_NUMBER
into outfile 'for_fsimport_diva' fields terminated by ',';
```

EXAMPLES

Tape Group File

The following comma-separated row is example of the contents of the Tape Group File:

```
MOVIE_CLASSICS_OFF_SITE,class_off_site_archive,relpt_off_site_archive,OFF_SITE
```

SEE ALSO

diva_sm_dbload(1), **diva_db_export**(1), **sm_diva_media_cleanup**(1), **sm_diva_media_import**(1),

NAME

fslocate − Locates files/directories based on file keys.

SYNOPSIS

fslocate -i [**-f** *mount_point* ...] [**-d**] [**-h**]

fslocate [**-f** *mount_point* ...] [**-d**] [**-h**] **-k** *key*[**,***pkey*] ...

DESCRIPTION

The **fslocate**(1) command provides an interface to find the paths of files or directories based on file keys. The user can query for directories or files. They can also specify an optional list of file system mount points. If mount points are not specified, the utility will get a list of file system mount points from the database.

The utility also provides an interactive interface. The key list option cannot be used in conjunction with the interactive mode.

OPTIONS

-d This option denotes directory lookups.

-f *mount_point* ...
This option contains a list of file system mount points which will be examined.

-k *key*[**,***pkey*] ...
This option contains a list of file keys with optional parent keys.

-i This option specifies that the interactive interface be used to query for keys.

-h This option specifies usage.

NAME

fsloglevel − Dynamically enables or disables specific log levels for TSM logs.

SYNOPSIS

fsloglevel -s on|off -l *level*... **-p** *process*...

fsloglevel -s on|off -l *level*... **-a**

fsloglevel -d *dumpLevel* **-p** *process*...

fsloglevel -d *dumpLevel* **-a**

DESCRIPTION

The **fsloglevel**(1) command controls the logging output of the Tertiary Manager resident processes. This command is usually used with assistance from Quantum technical assistance personnel to aid in obtaining additional information while investigating a potential problem.

Etac messages (messages with log levels from 8-16) are written to the $FS_HOME/logs/tac/tac_00 file. These messages provide internal details about Tertiary Manager processing.

Trace messages (messages with log levels from 17-26) are written to files in the $FS_HOME/logs/trace directory. These messages provide further internal details about Tertiary Manager processing.

History messages (messages with log levels from 27-36) are written to files in the $FS_HOME/logs/history directory. These messages provide command line logging and status information for Tertiary Manager client processing.

Perf messages are written to the $FS_HOME/logs/perf directory. These messages provide performance data points for the Tertiary Manager system. These logs are not needed during normal operations.

OPTIONS

-s *state* Used to enable or disable log levels of a Tertiary Manager resident process. Valid values are *on* or *off*.

-l *level* The log levels to be altered. Valid values are *tac*, *trace*, *hist*, *perf*, and integer values in the range *8-46*. Reference */usr/adic/TSM/logs/log_params* for further description of the log levels.

-d *dumpLevel*

The level of process information to dump to logs. Valid values are:
qustat : for statistics
thread : for thread information
config : for configuration information
internal : for internal information
all : for all of the above

-a Indicates all Tertiary Manager resident processes will be affected by this command.

-p *process*

The resident process name for which log messages will be changed.

WARNINGS

If all log levels are enabled, large quantities of logging information will be generated, potentially consuming a large portion of disk space. Keeping these log levels to a minimum during normal operations will save both disk space and time by not having to continually clean up the Tertiary Manager logs.

NAME

fsmedcopy − Move media content, defragment media, or generate a media fragmentation report.

SYNOPSIS

fsmedcopy [**COPY OPTIONS**] [**-b**] **-r** *mediaID...*

fsmedcopy [**COPY OPTIONS**] [**-b**] **-t** *mediatype* **-r** *mediaID...*

fsmedcopy [**COPY OPTIONS**] **-d** *mediaID* **-r** *mediaID...*

fsmedcopy [**-F** *type*] [**-f** *fill*] [**-w** *fragmentation*] [*mediaID...* | **-i**]

COPY OPTIONS

[**-F** *type*] [**-a**] [**-I**] [**-c** *class*] [**-s** *drivepool*] [**-v** *drivepool*] [**-u** *runtime*] [**-T ANTF**|**LTFS**] [**-G y**|**n**]
[**-e** *encryptiontype*] [**-M** *mkeyname*] [**-R** *copynum*] [**-O** *ops_limit*] [**-S** *skipScheme***]** [**-A** *file*] [**-B** *block*]

DESCRIPTION

The **fsmedcopy**(1) command can be used to move the contents from one storage medium to another, recreate a medium from another copy, or to generate a fragmentation report.

The files on a particular medium can be moved to other available media using the **-r** option. By default, an available destination medium within the same policy class is chosen. A blank medium is picked if there are no available media within the policy class. A specific destination medium can be specified with the **-d** option, a blank destination medium can be specified with the **-b** option, or a medium of a different type not in the policy class can be specified with the **-t** option.

If a destination medium is not specified, **fsmedcopy** will use up to three destination media to complete the copy process. Typically, only one media is required, but there are cases where more than one destination media might be required. When the destination is specified, the copy process will be stopped after the originally selected destination media runs out of available space. If three media have been used, and the copy has not completed, the copy process will stop, and the command will exit with a return value of 1, indicating failure. (A return value of 0 indicates success.) The command output will also indicate that some files could not be copied. Note that all files copied up to that point will be fine. The command will just need to be restarted if there are still files on the source medium to be copied.

If multiple media IDs are specified, it is recommended that the state of each medium be changed to *protect* using the **fschmedstate -s** command before executing the **fsmedcopy**(1) command. This is only necessary to ensure no more files are stored to the specified source media prior to the copy operation. However, if **-R** is used then the media must be changed to *unavail* instead.

After a file is successfully copied to the new medium, the original file's information in the system is updated with information about the new medium. This logically moves the file from the original medium to the new medium.

If the process is taking too long, it can be canceled using Control-C, or if the request ID is known, the **fscancel**(1) command can be used.

Media Maintenance

If errors occur frequently when attempting to write or read from a particular medium (possibly due to a medium that is reaching the end of its life), a user may want to use the **fsmedcopy -r** command to move all the data to a new or more reliable medium. If the medium can't be read then the **-R** option can used to try to recreate the medium from a different copy.

A medium may contain inactive file versions. Inactive file versions are file copies for which the original file has been changed or deleted since the copy was made. These inactive versions are NOT copied by default during the **fsmedcopy** operation. To change this behavior, use the -a option to copy inactive file versions along with the active version.

Media Defragmentation

Inactive file versions cause a medium to be filled with unusable space that the user may want to reclaim. These inactive versions are NOT copied by default during the **fsmedcopy -r** operation. This process in effect achieves a defragmenting of the media by keeping data for only active file versions.

Note: After all active file versions are copied, the source medium will revert to a blank status only if it does not contain any inactive file versions. If it does contain inactive file versions, the user must clean up the inactive file versions on the source medium with the **fsclean**(1) command in order for the source medium to become blank. This operation can be performed before or after the **fsmedcopy** operation.

Media Defragmentation Report

When no options are specified, the **fsmedcopy** command generates a report of all media in the system that fit the evaluation criteria for wasted space. The user can specify fill level (**-f**) and fragmentation level (**-w**) to use as the criteria, and whether to ignore media in a vault (**-i**). A report on all media in the system may be lengthy, therefore the user may choose to limit the report by specifying only certain media IDs.

The fill level is the percentage of total space used. The fragmentation level is the percentage of inactive data, based on the amount of space used, not the total available space. Based on this criteria, a list of media that are candidates for defragmentation is generated. The user may use this report to determine which media need to be defragmented.

It should be noted that the fill and fragmentation levels are calculated values based on current media characteristics. Internally to the command the values for these calculated percentages are kept to the calculated precision. This can cause some confusion when generating reports while using the (**-f**) or (**-w**) options; that is since the values are rounded when reporting to the nearest hundredths. For example the actual fill percentage for a media may be 50.4999999 percent but is reported as 50.50. If a second report is requested using a fill level of 50.5 the media in question will not be reported in this case since it is actually less than 50.5. Another example of a way in this can be observed is when a media has nothing but wasted space. The calculation for getting wasted space can internally produce a value 99.9999999 percent wasted space; the report however of wasted space will indicate 100 percent. Running a report requesting only media that have 100 percent wasted will not include this media. Note that the rounding done when printing works both ways so the actual values my be slightly larger or smaller than what is shown in the report.

REPORT STATUS

The defragmentation report produces the following information for each medium:

Media ID	The media ID
Fill Level	The fill level of the medium as a percentage
Wasted Space	The amount of wasted space as a percentage
Segment Count	Total number of segments on the medium
Available	The availability of the medium. Values are: **Y**, **N** Additional values for TEXT output only are: **Y (Vault)**, **N (Vault)**
Vaulted	Whether the medium is vaulted (XML/JSON output only). Values are: **Y**, **N**

OPTIONS

mediaID...

The media IDs for which to generate a fragmentation report.

-i Denotes to ignore media in a vault when reporting the media fragmentation.

-f *fill* Fill level threshold between 0 and 100 percent. The percentage of the medium that has been written, including active and inactive file versions. A default of 0 percent is used if not specified. See the Media Defragmentation section above for more information on the fill percentage.

-w *fragmentation*

The percent (0-100) of wasted space out of the filled space on a medium. The percentage is based on the amount of filled media space, not the total capacity of the medium. A default of 0 percent is used if not specified. See the Media Defragmentation section above for more information on the wasted space percentage.

-r *mediaID...*

Initiates file data movement from the specified source media ID(s). Note that copies from S3 Glacier or Azure Archive media are not supported.

-a Indicates all files (active and inactive versions) are to be copied from the source medium. After all versions are copied, the source medium will revert to blank status.

-c *class* Copy only data file(s) belonging to the specified class. Note, this is useful only for non-tape media which are not associated with a single policy class.

-d *mediaID*
Use the given media ID as the destination medium.

-b Use blank media for destination media. This option is not supported for Object Storage media.

-t *mediatype*
The type of destination medium to be used. Depending on the platform used, the following media types are supported by Tertiary Manager software:

AWS
AZURE
LATTUS
GOOGLE
GOOGLES3
S3
S3COMPAT
LTO
LTOW
3592
T10K
SDISK

If **-t** is not specified, the policy class definition will be used.

-s *drivepool*
The drive pool from which the source drive will be selected when copying files from the specified medium. If the **-s** option is not used, the default drive pool will be dictated by the policy class definition. The special "_" character is permitted to identify the drive pool.

-v *drivepool*
The drive pool from which the destination drive will be selected when copying files from the specified media.

If the **-v** option is not used, the default drive pool will be dictated by the policy class definition. Additionally, if the **-R** option is specified, then the drive pool will be based on the copy number of the media being recreated and not the media that is used to make the copy from. The special "_" character is permitted to identify the drive pool.

-u *runtime*
Maximum allowable time in hours for the command to finish.

-T ANTF|LTFS
Specifies the media format type that will be used when formatting or selecting media. **ANTF** is the Quantum internal tape format. **LTFS** is the Linear Tape File System specification tape format. A value of **ANTF** will format media with a single partition containing ANTF volume labels. This data partition will store StorNext file data. A value of **LTFS** will format media with two partitions containing LTFS volume labels. The index partition will store LTFS metadata and the data partition will store StorNext file data. The **LTFS** media format type is only applicable to LTO media generation 5 and beyond (e.g. LTO-5, LTO-6, etc). Any attempt to assign **LTFS** to non-LTO media types will result in an error.

If the **-T** option is not specified, the default will be determined as follows:

A media type that does not support **LTFS** will default to **ANTF**.
A media type that supports **LTFS** will default to the policy class copy definition.

-G y|n Generate and maintain a checksum for each copied file. If this option is enabled, a checksum will be generated as each file is written (stored) to the new media. Policy class settings are not used when this option is not specified so it defaults to 'n'. Any file being copied that already had a checksum generated will ignore this option and continue to use the existing value.

-R *copynum*

This option is intended to be used when there are issues with a medium such that files can not be read from it. Files on the medium can be recreated from alternate media which have different copies of the files. The *copynum* argument specifies which copy to use. For example:
If medium A is bad and has copy 1 files, then specifying **-R 2** will use copy 2 media to recreate a new copy 1 instance of medium A files on other copy 1 media.

If **-R 0** is specified then media associated with any other copy could be used. The means media from different copies could be used to recreate new copies of the bad source medium files.

There are some limitations with this option:

The media to be replaced must have an unavailable status.

If any alternate source media are in an unusable state or vaulted then they will not be used.

The size of the file segments on the alternate source media to be used and the media to be recreated must be the same or else those segments will not be copied. Typically this only occurs when the media types are different for the alternate source media and the media being recreated. A notification will be output if there were any files that could not be copied due to mismatched segment sizes.

-I This option can be used to ignore the cleanup of the Object Storage when the source is an object media. If the source is not an object media then this option is ignored. When this option is provided, the cleanup of the Tertiary Manager database will still take place but there will be no attempt to delete the objects from the Object Storage. This option could be useful when migrating data off of an Object Storage appliance that will no longer be used.

-O *ops_limit*

This provides a means of throttling the request by limiting the number of I/O operations, requests, that are generated. The default for this is 10.

For requests that don't involve tape devices, limiting the number of I/O requests will also restrict the number of streams used. The number of streams that can be used will be the lesser of the *limit* value or the max streams defined for the device. The stream limit is applied to each source and destination device so *limit* number of streams will be used for each. For example: if **-O 3** is specified then 6 streams would be used; 3 for the source and 3 for the destination. However, if the max streams defined for one of the devices is smaller, then the request would be further restricted by that. Using the same example, if the source device was configured with max streams of 2, then only 2 streams would be utilized for both source and destination devices.

When setting this value for non-tape devices keep in mind that **fsmedcopy**(1) requests run at a lower priority than store and retrieve requests so they will only use available streams when there are no store/retrieve requests requiring those resources.

For requests using tape devices this will be the number of queued I/O requests. Tape I/O requests are processed sequentially so there is no benefit in adjusting this value. It is recommended to use the default setting.

-e *encryptiontype*

The encryption algorithm to be applied to the content of files stored into an Object Storage system. This option is ignored for all other media types. Currently, the following encryption types are supported:

none	No encryption
SSE_S3	Server-side AES256 encryption with S3-managed keys

SSE_KMS Server-side AES256 encryption using the AWS Key Management Service (KMS)

For the **SSE_KMS** encryption type, the default customer master key (CMK) can be used for encryption or a CMK can be created using the AWS Key Management Service and specified with the **-M** option.

If this option is not specified, the encryption type will be set to **none**.

-M *mkeyname*

The Master Key name that has been created using the AWS Key Management Service and requested for server-side encryption.

-S skipScheme

This option is only valid when all the source media specified are tape. This indicates the scheme for the set of errors that result with one or more files being skipped when one of those errors is encountered. The copy operation will continue with a subsequent file after the skipped files. The number of times to perform the skip logic for each medcopy operation is controlled by the MEDCOPY_SKIP_RETRIES sysparm. See the fs_sysparm.README file for more information on both of these sysparms. If this option is not specified, then the default behavior is for the copy operation to stop when a error is encountered with the source tape.
The process of skipping over a bad section of tape searching for the next usable location can be time consuming. To help identify when this is occurring the "state" value reported by the **fsqueue -s** command will show a value of *PROCESS* instead of *COPY*. Additionally, there are entries in the */usr/adic/TSM/logs/tac/tac_00* log indicating details on the search processing.
Valid values are:

1 This scheme will limit the number of source tape errors/check conditions that are considered as indications that there may be a bad spot on the tape. The file being processed when one of these errors occurs will be failed. The processing will then continue with the next file that is located at a different tape block address. Additional check conditions can be specified with the MEDCOPY_SKIP_ERRS sysparm. See the fs_sysparm.README file for more information. This scheme is not supported by LTFS formatted source tapes, so for this situation, scheme 1 will behave as if scheme 2 was specified.

2 This scheme will treat any source tape errors/check conditions as indications that there may be a bad spot on the tape. The file being processed when of these errors occurs will be failed. The processing will then continue with the next file that is located at a different tape block address.

-A *file* This option is only valid when one source media is specified for replacement and is tape. The copy processing will start with the next file that is located at a different tape block address after the specified file.

-B *block*

This option is only valid when one source media is specified for replacement and is tape. The copy processing will start with the next file that is located at a different tape block address after the specified block.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

NOTE

fsmedcopy only supports the following source to destination media combinations:

Tape to Tape
Tape to Object Storage
Object Storage (except Glacier) to Object Storage
Storage Disk to Object Storage

EXAMPLES

Perform a fsmedcopy replace with no skip scheme when the source tape has errors. The "aborted due to errors" message indicates that the copy operation was stopped prior to processing all files due to the error that was encountered.

```
fsmedcopy -a -r E00002
FS0589 10 0000001946 fsmedcopy interim: Tertiary Manager software request receive
FS0624 10 0000001946 fsmedcopy interim: File /sn/fs1/tape/f100 copied to medium E
FS0355 10 0000001946 fsmedcopy interim: The last error was repeated for 49 more t
FS0574 10 0000001946 fsmedcopy interim: Copying files from medium E00002 failed.
FS0390 10 0000001946 fsmedcopy failed: 0 out of 1 requests were successful.
```

Perform same fsmedcopy replace with skip scheme 1 when the source tape has errors. Note, that scheme 1 only applies to ANTF tape formats. The "skipped due to errors" message indicates the number of files that were skipped due the reported errors. The "no more skip retries" message indicates the number of files that were not processed due to the skip retry limit being exceeded. Reference the MEDCOPY_SKIP_RETRIES sysparm.

```
fsmedcopy -a -r -S 1 E00002
FS0589 10 0000001965 fsmedcopy interim: Tertiary Manager software request receive
FS0624 10 0000001965 fsmedcopy interim: File /sn/fs1/tape/f110 copied to medium E
FS0624 10 0000001965 fsmedcopy interim: File /sn/fs1/tape/f112 copied to medium E
FS0624 10 0000001965 fsmedcopy interim: File /sn/fs1/tape/f290 copied to medium E
FS0624 10 0000001965 fsmedcopy interim: File /sn/fs1/tape/f340 copied to medium E
FS0624 10 0000001965 fsmedcopy interim: File /sn/fs1/tape/f342 copied to medium E
FS0355 10 0000001965 fsmedcopy interim: 5 files were skipped due to errors with c
FS0355 10 0000001965 fsmedcopy interim: 1 files were failed due to no more skip r
FS0574 10 0000001965 fsmedcopy interim: Copying files from medium E00002 failed.
FS0390 10 0000001965 fsmedcopy failed: 0 out of 1 requests were successful.
```

Perform same fsmedcopy replace with skip scheme 2 when the source tape has errors.

```
fsmedcopy -a -r -S 2 E00002
FS0589 10 0000001986 fsmedcopy interim: Tertiary Manager software request receive
FS0624 10 0000001986 fsmedcopy interim: File /sn/fs1/tape/f110 copied to medium E
FS0624 10 0000001986 fsmedcopy interim: File /sn/fs1/tape/f290 copied to medium E
FS0624 10 0000001986 fsmedcopy interim: File /sn/fs1/tape/f340 copied to medium E
FS0624 10 0000001986 fsmedcopy interim: File /sn/fs1/tape/f342 copied to medium E
FS0624 10 0000001986 fsmedcopy interim: File /sn/fs1/tape/f599 copied to medium E
FS0355 10 0000001986 fsmedcopy interim: 4 files were skipped due to errors with c
FS0574 10 0000001986 fsmedcopy interim: Copying files from medium E00002 failed.
```

```
FS0390 10 0000001986 fsmedcopy failed: 0 out of 1 requests were successful.
```

Perform a fsmedcopy replace with skip scheme 1 when the destination tape has errors.

```
fsmedcopy -a -r -S 2 E00004
FS0589 10 0000002590 fsmedcopy interim: Tertiary Manager software request receive
FS0624 10 0000002590 fsmedcopy interim: File /sn/fs1/tape/f110 copied to medium E
FS0355 10 0000002590 fsmedcopy interim: 147 files were failed due to errors with
FS0574 10 0000002590 fsmedcopy interim: Copying files from medium E00004 failed.
FS0390 10 0000002590 fsmedcopy failed: 0 out of 1 requests were successful.
```

SEE ALSO

fsmedout(1), **fsrecover**(1), **fsmedinfo**(1), **fscancel**(1), **fsqueue**(1), **fsversion**(1), **fsdefrag**(1), **fskey**(1)

NAME

fsmedin − Logically enters media into the Tertiary Manager system from the Media Manager system.

SYNOPSIS

fsmedin -r [**-F** *type*] [**-q** *quantity*] [**-t** *mediatype*]

fsmedin -b [**-F** *type*] [**-q** *quantity*] [**-t** *mediatype*] [**-w**] [**-c** *class*]

fsmedin -b [**-F** *type*] [**-w**] [**-c** *class*] *mediaID...*

fsmedin -b [**-F** *type*] [**-q** *quantity*] [**-t** *mediatype*] [**-c** *class*] [**-T ANTF**|**LTFS**]

fsmedin -b [**-F** *type*] [**-c** *class*] [**-T ANTF**|**LTFS**] *mediaID...*

fsmedin -A [**-F** *type*] *mediaID...*

fsmedin -k [**-F** *type*] *mediaID...*

fsmedin -i [**-F** *type*] *mediaID...*

DESCRIPTION

Logical media entry operations into the Tertiary Manager software are performed with the **fsmedin**(1) command. Media are introduced into the Quantum storage subsystem by an operator using the Library Operator Interface console. The media is physically entered into the desired library and into the correct Media Manager MediaClass group known to Tertiary Manager software, i.e., F0_LTO_ADDBLANK. The **fsmedin**(1) command reclassifies the media to a different Media Manager MediaClass name and makes the media available to the Tertiary Manager software.

The **-q** option limits the *quantity* of media specified and indicates the number of media to be used. If this information is not specified, the system parameter default, VS_DEF_QUANTITY, value is used.

Media can be entered into the Tertiary Manager software by checking in nonblank media (**-r**), importing nonblank media (**-i**), or adding blank media (**-b**).

Additionally, cleaning media can be entered using the **-k** option.

Checking in Nonblank Media

The **fsmedin -r** command allows re-entry of checked out media. Checking in media does not require adjustments to SNFS file systems nor does it cause the medium to be mounted, because the file and media information are still within the Tertiary Manager database, only the status of the medium is updated.

Checked in media can be entered into any storage subsystem controlled by Tertiary Manager software as long as that subsystem is valid for the entered media type. After a medium is checked in, that medium can have files stored and retrieved from it.

Importing Nonblank Media

The **fsmedin -i** command allows the import of nonblank media into the Tertiary Manager system. Media can be exported using the **fsexport**(1) command, which will remove the database entries from the system. Importing media requires adjustments to the Tertiary Manager system and the SNFS file systems using the **fsimport**(1) command because the file and media information are not known within the Tertiary Manager database.

Imported media are logically write protected, the formatting is withheld, and the ownership/state of the media are set to "Imported" to indicate that the files were ingested from another system. Files on an imported medium can be retrieved after **fsimport**(1) has been run on it. The medium is logically read-only and may not be written to while it is in the "Imported" state.

Adding Blank Media

The **fsmedin -b** command is used to add blank media to a storage subsystem. Blank media can be added to the general blank pool or to a specified policy class pool (**-c**). Blank media in the general blank pool are available for use in any policy class. Blank media in a policy class pool are only available for use in that particular policy class.

Blank media is either formatted immediately or withheld from formating until a format request is made. If blank media are added with the withhold formatting option (**-w**), the formatting of the media is withheld

until the media are either chosen for data storage or when the system administrator manually issues the **fsformat**(1) command. If blank media are added without the **-w** option, the media will be immediately formated according to the CLASS_MEDIA_FORMAT system parameter unless overridden by the **-T** format option.

It is recommended that the withhold formatting option be used in order to limit the number of mount operations.

If the media identifiers are entered then those specific media will be added to Tertiary Manager system.

OPTIONS

-b Add blank media.

-r Re-enter media that were previously removed (checked out) from the archive.

-i Import nonblank media.

-k Add cleaning media.

-w Withhold formatting media immediately.

-c *class* Policy class. A policy class name can have maximum of 16 alphanumeric characters. The special "-", "." and "_" characters are also permitted. It is important to note that when using this option, if a quantity is not specified, whatever quantity is specified in VS_DEF_QUANTITY will be used. Unless modified, this quantity has a default of 99. Using this option and the **-q** option will enable you to add a quantity greater than that specified in the hardlimit for the class. Reference the **-h** option in **fsaddclass**(1).

-q *quantity*
Number of media to be added to the storage subsystem. The maximum number of media that can be added is limited to 99 except when adding blank media. The maximum value is 10000 when the **-b** option is used. If **-q** option is not used, the default will be specified by the system parameter VS_DEF_QUANTITY. The default number is usually 99.

-t *mediatype*
Defines the type of media being imported. The following media types are supported by Tertiary Manager software:

LTO
LTOW
3592
T10K

-T ANTF|LTFS
Specifies the media format type that will be used when formatting or selecting media. **ANTF** is the Quantum internal tape format. **LTFS** is the Linear Tape File System specification tape format. A value of **ANTF** will format media with a single partition containing ANTF volume labels. This data partition will store StorNext file data. A value of **LTFS** will format media with two partitions containing LTFS volume labels. The index partition will store LTFS metadata and the data partition will store StorNext file data. The **LTFS** media format type is only applicable to LTO media generation 5 and beyond (e.g. LTO-5, LTO-6, etc). Any attempt to assign **LTFS** to non-LTO media types will result in an error.

If the **-T** option is not specified, the default will be specified by the system parameter CLASS_MEDIA_FORMAT.

-A Intended to be used during the SAM-QFS conversion process to import SAM-QFS media into the Tertiary Manager system. This results with the media being logically write protected and formatted without any formatting actually occurring.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

NOTES

This command is not valid for storage disks.

SEE ALSO

fsmedout(1), **fsformat**(1), **fsfilecopy**(1), **fschmedstate**(1), **fsimport**(1)

NAME

fsmedinfo − Generate a report on media based on their current status.

SYNOPSIS

fsmedinfo [**-l [**-g**] [**-s *starttime***] [**-e *endtime***]]** [**-F** *type*] *mediaID*...

fsmedinfo [**-l [**-g**] [**-s *starttime***] [**-e *endtime***]]** [**-F** *type*] **-B** *batchfilename*

DESCRIPTION

The **fsmedinfo**(1) command produces either a short report or a long report on the specified media, based on the options entered. One or more media identifiers must be entered.

REPORT STATUS

The default short report produces the following information for each medium:

Media ID — Media identifier and copy number of the specified medium. The number in parenthesis indicates the copy ID of the files on the medium: (1) -primary copy, (2) -secondary copy, etc.

Media Type — The medium's physical media type. Values are:

AWS
AZURE
LATTUS
GOOGLE
GOOGLES3
S3
S3COMPAT
LTO
LTOW
3592
T10K
SDISK

Class ID — The medium's associated policy class.

Media Class — The medium's media class (tape only). Example: **F0_LTO_DATA**, etc.

Media Status — The medium's current status for file storage and retrieval. Values are: **AVAIL**, **UNAVAIL**, **PENDING REMOVAL**.

Current State — The state of the archive in which the medium is currently located (tape only). Values are: **On-line**, **Off-line**, **Diagnostic**, **Unavailable**.

Assignment — Whether the medium is available for mounting (tape only). Values are: **Allocated**, **Free**.

Action State — Whether and why the medium is being moved (tape only). Values are: **None**, **Export**, **Migration**, **Import**, **Mount/Move**, **Operator**, **Checkout**, **Checkin**, **Move**, **Entering**.

Mark Status — The medium's current mark status. Values are: **UNMARKED**, **CHECK-OUT**, **PENDING**, **ERROR**.

Suspect Count — The number of times read/write positioning failures were detected for the medium. If the medium is changed to unsuspect using the **fschmedstate**(1) command, the suspect count returns to zero.

Write Protect — Whether or not the medium is write protected. Values are: **Y**, **N**. *Ownership* Whether or not the medium is imported. Values are: **Imported**, **Own**.

Storage Area — Area where medium is located. Examples: **VolSub**, **Storage Disk**, **Object Storage appliance**, etc.

Location State The medium's location relative to its archive (tape only). Values are: **Archive**, **Checkout**, **Intransit**.

Current Archive The archive in which the medium is currently located (tape only).

Pending Archive The archive into which the medium is being moved (tape only).

Medium Location The medium's location within its storage area. Values for tape media are: **SLOT/BIN**, **DRIVE**. For Storage Disk media, it will be the device mount point. For Object Storage media, it will be the namespace.

External Location The medium's location if not currently in an archive (tape only).

Import Date The date and time the medium was added to the archive (tape only).

Last Accessed The date and time the medium was last accessed.

Move Count The number of times the medium has been moved (tape only).

Mount Count The number of times the medium was mounted by the Tertiary Manager.

Formatted Whether or not the medium is formatted. Values are: **Y**, **N**.

Format Type The media format type. Note that this will display the previous media format type when the medium transitions from formatted to unformatted (i.e. formatted tape is cleaned and returned to the system blank pool). The following lists types of media and their supported format types.

Never formatted: UNKNOWN
Tape: ANTF, LTFS, AXF, DL64K or DL512K
SDISK: ANTF
Object Storage: NONE

Number of Segments Total number of segments on the medium that have not been logically removed via **fsclean**(1).

Bytes Used Space used (in bytes).

Space Remaining Space remaining (in bytes). The software will set this value to **123** when an End-Of-Tape(EOT) early warning condition is encountered while writing. This provides an indication of why those media are considered full. The SCSI_NO_SNS_EW_ERRS sysparm can be used change the EOT early warning handling. See *fs_sysparm.README* file for details.

Percentage Used Space used (percentage).

The long report (**-l** option) will add the following information for each file segment:

Segment Offset
The byte offset in the file where the segment begins. (Shown for Object Storage media only.)

Segment Length
The length of the segment in bytes.

Version The version number of the file. If the version number is enclosed by double parentheses this means the segment is for an old version or a deleted file.

Modify/Delete Date
If a file is still current on the disk then the modtime is displayed, for removed files or old versions the delete date is shown. (This column is displayed if neither the **-s** or **-e** options have been used.)

Store/Recover Date
If a file has never been removed or if the file has been removed but has not been recovered, then the store time is displayed. If a removed file has been recovered back to disk then the recover time is shown. (This column is displayed if the **-s** and/or **-e** options are used since the report is sorted by this time.)

Object ID

The object id for the file if stored on Object Storage media. If the ID is not set, then NULL is reported.

Seg:TotSeg

The segment number of the file and the total number of segments that comprise the file.

Key:Pathname (name at store time)

The internal file key for the file and the name of the file at store time. If the file has been renamed since store time, the new name will not be reflected in this report. This column is also preceded by a **'-'** if the file has been deleted.

If there is a failure in generating the path name that a file was stored with one of these strings will be reported for the path:

"PATH UNKNOWN, (name rec not found)"

"<mount_point>/PathUnknown"

The report will still include the key for these files even if the path is not known. If there are files in a report in this state, the system will still continue to operate normally. However, Quantum technical assistance should be notified to verify this is not a symptom of another problem.

OPTIONS

mediaID...

One or more media identifiers on which to report. If multiple media identifiers are entered, the media identifiers must be separated by spaces. The number of media identifiers that can be entered is limited by the local machine's command-line length.

-B *batchfilename*

The batch file contains the list of entries to be processed. Each entry must be listed one per line. Do not enclose the entry in quotes, use escape characters or any character not in the actual name.

-l Produce the long form of the report that contains the same information as the short form, plus a list of the file segments on the medium. (See the description of the fields in the output above.)

-g Used with the long option to report the segment number and the total number of segments for each file on the media to be reported.

-s *starttime*

Used with the long option to indicate a start time of the files on the media to be reported. If this time and/or the end time are used with the (**-l**) option then only the files on the media that were stored or recovered during the specified time window will be displayed. When this option or the **-e** is used, the output will be sorted per file system by the store/recovery time. The format for the time parameter is YYYY:MM:DD:hh:mm:ss where:

YYYY = Numeric, year

The following are optional; defaults are shown:,

MM = Numeric, two-digit month
(default:01 (January)),

DD = Numeric, two-digit day
(range: 01-31, default:01),

hh = Numeric hour
(range: 00-23, default:00),

mm = Numeric minute
(range: 00-59, default:00),

ss = Numeric second
(range: 00-59, default:00)

-e *endtime*

Used with the long option to indicate an end time of the files on the media to be reported. (See the **-s** option description for more time related info.)

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsmedlist(1), **fschmedstate**(1), **fsclean**(1), **fsxsd**(1)

NAME

fsmedlist − Generate a report of storage media grouped by policy class.

SYNOPSIS

fsmedlist [**-c** *class*...] [**-g**] [**-l**[**dhzokeqmfbpisaunx**]] [**-F** *type*]

DESCRIPTION

The **fsmedlist**(1) command generates a report of Tertiary Manager storage media. The report groups the media by policy class, including the **genscratch** pool which contains tape media that have yet to be associated with a class and the **NA** pool which contains non-tape media that span multiple classes. The **fsmedlist**(1) command supports a short report format and a long report format, as explained below.

Short Report

The short report lists the total number of media in each policy class and the number of those media belonging to each of the categories shown below. Note: Not all of these categories are mutually exclusive, which means a storage medium can be counted in more than one category.

Drive	Residing in a drive
Slot	Residing in a slot/bin
Exit	Exiting the Tertiary Manager system
Out	Checked out of the Tertiary Manager system
Blank	Blank media
Prot	Write-protected media
Avail	Available for use
Susp	Designated as suspect
Mark	Marked media
Nontape	Nontape media associated with the policy class
Total	Total number of media associated with the policy class

The **NA** pool includes non-tape storage media such as SDISK and Object Storage, thus only the following categories apply to this pool: **Blank**, **Prot**, **Avail**, and **Susp**.

Long Report

The long report lists the total number of media in each policy class and the number of those media belonging to each of the categories shown below. Additionally, the media IDs are listed under each category. Note: Not all of these categories are mutually exclusive, which means a storage medium can be counted in more than one category.

In Drive	Residing in a drive
In Bin	Residing in a slot/bin
Exiting	Exiting the Tertiary Manager system
Out of StorNext	Checked out of the Tertiary Manager system
Marked for Check-Out	Temporarily checked out of the Tertiary Manager system
Marked for Export	Marked for export out of the Tertiary Manager system
Marked as Errored	Errored media
Marked	Marked media
Formatted Blank	Formatted blank media
Unformatted Blank	Unformatted blank media
Write Protected	Write-protected media

Imported	Media imported into the Tertiary Manager system
Suspect	Designated as suspect
Available	Available for use
Unavailable	Unavailable for use
Unavailable to Tertiary Manager	Unavailable or pending removal from the Tertiary Manager system
Nontape Media	Nontape media associated with the policy class
Total	Total number of media associated with the policy class

The **NA** pool includes non-tape storage media such as SDISK and Object Storage, thus only the following categories apply to this pool: **Formatted Blank**, **Unformatted Blank**, **Write Protected**, **Imported**, **Suspect**, **Available**, and **Unavailable**.

OPTIONS

-c *class...*

One or more policy classes for which the report is to be generated. Only media associated to these policy classes will be included in the report. To specify a report for nontape media such as SDISK and Object Storage, enter "NA" as the class name. If the **-c** option is not used, a media report will be generated for all policy classes, including media in the **genscratch** pool and media in the **NA** pool.

-g Generate a report on the media in the **genscratch** pool.

-l Generate a report in the long report format. If this option is not specified, a short report is generated. All categories are listed in the long report if no option modifiers are specified. To restrict the long report to specific categories, option modifiers may be specified immediately after the **-l** option without spaces. For example, **fsmedlist -lkbd** will produce a long report for media in the following categories: **Marked for Check-Out**, **Unformatted Blank**, and **In Drive**. All option modifiers and their corresponding categories are listed below.

Categories for Media Location

- **d** **In Drive**
- **h** **In Bin**
- **z** **Exiting**
- **o** **Out of StorNext**

Categories for Marked Media

- **k** **Marked for Check-Out**
- **e** **Marked for Export**
- **q** **Marked as Errored**
- **m** **Marked**

Categories for Media State

- **f** **Formatted Blank**
- **b** **Unformatted Blank**
- **p** **Write Protected**

i **Imported**

s **Suspect**

a **Available**

u **Unavailable**

n **Unavailable to Tertiary Manager**

x **Nontape Media**

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsmedinfo(1), **fsxsd**(1)

NAME

fsmedloc − Change or report the external location of media.

SYNOPSIS

fsmedloc *mediaID...* **-l** *location*

fsmedloc [*mediaID...*]

DESCRIPTION

The **fsmedloc**(1) command changes or reports the external location of one or more specified media. This command is valid only for media that are checked out or marked for check out. Media removed by use of the **fsmedout**(1) are valid candidates for **fsmedloc**(1). Without the **-l** option, the **fsmedloc**(1) command reports the location for the given *mediaID*. The **fsfsmedloc**(1) report indicates that a medium is checked out of the Quantum storage subsystem by placing a '+' sign next to the media location description. If the '+' does not appear, the medium is marked for removal.

OPTIONS

mediaID...

One or more media identifiers to have their external location description(s) changed or reported. Entry of at least one media identifier is required to make a change. Multiple media identifiers must be separated by spaces. The number of media identifiers that can be entered is limited by the local machine's command-line length. If no media identifiers are specified and the **-l** option is not used, a report is generated that provides the location of all media.

-l *location*

Text describing the new external location. The *location* parameter must be less than 255 characters. If the location is composed of more than one word, it must be surrounded by single quotes (') or double quotes (").

SEE ALSO

fsfileinfo(1), **fsmedinfo**(1), **fsmedout**(1)

NAME

fsmedout − Initiate removal of media from the Quantum storage subsystem.

SYNOPSIS

fsmedout [**-F** *type*] [**-q** *quantity*] [**-c** *class*] **-b**

fsmedout [**-F** *type*] [**-k**] [**-f**] *mediaID...*

fsmedout [**-F** *type*] [**-l** *location*] [**-f**] **-r** *mediaID...*

fsmedout [**-F** *type*] [**-l** *location*] [**-f**] **-m** *mediaID...*

DESCRIPTION

The **fsmedout**(1) command initiates removal of media from the Quantum storage subsystem. Media can be permanently removed or temporarily removed (checked out) from the system. This command only performs database operations on the Tertiary Manager system to initiate the removal process. Afterward, the media can be physically ejected from the Quantum storage subsystem with Media Manager commands or by using the Library Operator Interface in the StorNext GUI.

Permanent Removal

When media are permanently removed, all knowledge of that media is deleted from the Tertiary Manager system. For that reason, permanent removal is allowed only for media that do not contain file data. This applies to blank and error media.

Blank media are eligible to be permanently removed regardless of whether the media have been assigned to a policy class. Blank media can be processed individually or in bulk.

Error media are media that failed to format. Such media appear with an **Error** mark status in their **fsmedinfo**(1) report. Since error media have no file information tracked in the database, these media may be removed like blank media.

Check-Out

Media can be temporarily checked out with the intent of reentering the media at a later date. The Tertiary Manager retains knowledge of checked out media and any file data that resides on them. This allows the media to be reentered into the system via **fsmedin**(1) at a later date. Checked out media appear with a **Check-Out** mark status in their **fsmedinfo**(1) report. Blank media may also be checked out.

Media can also be merely marked for check-out. This does not initiate the media check-out operation but is a useful way to prevent further access to media. A subsequent **fsmedout**(1) command can be used to initiate the check-out operation, or the **fschmedstate**(1) command can be used to unmark the media.

Checking out media or merely marking media for check-out will render the files and/or file segments on the media temporarily inaccessible. However, if a file has additional copies available on other media, the file data can be accessed from those media.

Incomplete files are allowed to be checked out. This allows part of a file (a segment of a spanned file) residing on a specific medium to be removed and then reentered. The file segments that are not checked out can still be accessed using the partial retrieve feature of the **fsretrieve**(1) command.

Note that checking out media with the **fsmedout**(1) command differs from exporting media with the **fsexport**(1) command. Both commands can initiate the removal of media that contain file data. But while the Tertiary Manager retains knowledge of checked out media, it does not retain any knowledge of exported media.

OPTIONS

mediaID...

A list of one or more media IDs separated by spaces, up to a maximum of 99 media IDs. Without the **-r** or **-m** options, the listed media are assumed to be blank or error media and a permanent removal will be initiated. If the listed media were previously marked for check-out, specifying the media with no options will initiate the check-out operation.

-b Initiates permanent removal of blank media. The Tertiary Manager system chooses the blank media to be removed.

-q *quantity*
Used with the **-b** option to specify the number of blank media to be removed. The maximum number of media that can be specified is 10000. The default *quantity* value is defined by the VS_DEF_QUANTITY system parameter.

-c *class* Used with the **-b** option to specify the policy class from which the Tertiary Manager system should choose the blank media to be removed.

-k When permanently removing media, this option keeps the selected media in the library instead of directing the Media Manager to remove it.

-f Forces the check-out or permanent removal operation that is being initiated. This can be used if the media status appears as **Unavail** in the **fsmedinfo**(1) report and has already been physically removed from the Media Manager without using the **fsmedout**(1) command. The removed media should be reentered into the Media Manager system to correct this situation.

-r Initiates media check-out operation.

-m Marks media for check-out. Does not initiate actual media check-out. The **fsmedout**(1) command can be used again at a later time (with the **-r** option or with no options) to initiate the check-out operation. Alternatively, the **fschmedstate**(1) command can be used to unmark the media for check-out.

-l *location*
For media that are being checked out, *location* specifies the physical location where the media are intended to be moved. The *location* argument is limited to 255 characters. This is merely an aid for tracking the location of checked out media. The location is displayed in the "External Location" field of the **fsmedinfo**(1) report. If the physical location of checked out media changes at a later time, "External Location" can be updated with the **fsmedloc**(1) command.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

NOTES

This command is only valid for tape media.

SEE ALSO

fsmedinfo(1), **fsmoverpt**(1), **fsmedin**(1), **fsmedloc**(1), **fschmedstate**(1), **fsretrieve**(1), **fsrminfo**(1), **fsexport**(1)

NAME

fsmedread − Reads file from a media.

SYNOPSIS

fsmedread [**-c** *cdbn*] **-f** *fsn* **-l** *ldbn* **-d** *device dest*

fsmedread -u *URL* [**-n** *namespace* **-o** *objectname*] [**-v** *provider*] [**-t** *mediatype*]
[[**-U** *username* **-P** *password*] | [**-A** *CAP_agency* **-m** *CAP_mission*]] [**-a** *serverAuth*]
[**-e** *encryptiontype*] [**-S** *signingtype*] [**-Y** *authenticationtype*] [**-r** *role*] [**-O** *storageclass*]
[**-T** *restoretier*] [**-Z** *authentication_endpoint*] [**-H** *CAP_hostport*] [**-C** *certfile* | **-R** *certpath*]
[**-L** *clientcertfile*] [**-k** *clientkeyfile*] [**-p** *clientkeypass*] *dest*

fsmedread *source dest*

DESCRIPTION

This command will retrieve data from the specified location into the specified file for SNSM ANTF formatted tapes, storage disk media, or Object Storage media (AXR, AWS, Azure, S3, S3COMPAT, GOOGLES3). This does not support LTFS, AXF, DL64K or DL512K formatted tapes. For tape and storage disk media, it is recommended to use **fsmedscan**(1) first to determine the file's location on the media.

Before this command is run, tape media must be mounted in the tape drive which can be done using **fsmount**(1). There is no such a requirement for other media.

OPTIONS

-f *fsn* File Sequence Number (FSN/tape mark) where file is located.

-c *cdbn* Cumulative Data Block Number (CDBN) where file is located.

-l *ldbn* Logical Data Block Number (LDBN) where file is located. This is the byte offset from the previous FSN.

-d *device*
SCSI device path of tape drive where media is mounted.

-u *URL* Full URL to the object located on the Object Storage media.

-n *namespace*
Namespace name. Required if the Object Storage media is not using the AXR REST API.

-o *objectname*
Object name. Required if the Object Storage media is not using the AXR REST API.

-v *provider*
Identifies the provider of the appliance where the file is located. The appliance provider determines the format of provider-specific REST headers. The following provider types are supported by Tertiary Manager software:

AWS
AZURE
GOOGLE
LATTUS
S3COMPATIBLE

-t *mediatype*
The object storage media type. The following object storage media types are supported by Tertiary Manager software:

AWS
AXR
AZURE
GOOGLE
GOOGLES3
S3
S3COMPAT

The default is **AXR**.

-U *username* **-P** *password*

Username and password to access the URL if necessary. These are required if the Object Storage media is using S3 or Azure Blob REST API from the following providers:

AWS (standard and STS)
AZURE
GOOGLE
LATTUS
S3COMPATIBLE

-A *CAP_agency* **-m** *CAP_mission*

The agency and mission associated with the target C2S account. These options are valid only for the CAP authentication type.

-e *encryptiontype*

The encryption algorithm to be applied to the content of the file stored into an Object Storage system. This option is ignored for all other media types. Currently, the following encryption types are supported:

none None (No decryption of file's content on Client Host)

SSE_S3

SERVER_AES256 (File can be Server-side AES256 encrypted or not encrypted)
This encryption type is valid only if supported by the Object Storage system.

If this option is not specified, the encryption type will be set to None.

-a *serverAuth*

The server authorization setting. Default is 2. The valid options are:

0 none

1 verify peer only

2 verify peer and host

-S *signing_type*

Used to specify the signing type for the requests sent to the Object Storage and/or authentication server. The following signing types are supported:

V2
V4
AZURE

The default is **V4** for AWS and S3 compatible media, **V2** for GOOGLES3 and Lattus S3 media, and **AZURE** for Azure media. This option is not valid for AXR and Google media. When configuring V4, chunked mode is preferable and should be used if supported by the object storage system. For object storage which does not support chunked mode, such as AWS Snowball, you must specify the full payload mode signing option, **-J**, for the object storage bucket.

-Y *authentication_type*

An authentication type is required for all Object Storage media except for AXR and GOOGLE. The following authentication types are supported:

STANDARD
STS_PUBLIC
STS_GOVCLOUD
CAP

If this option is not specified, the authentication type will be set to **STANDARD**.

The **STANDARD** type applies to all media and authenticates with a user name and password for Object Storage access. For S3 compatible media the user name and password are the Access Key

Id and Secret Access Key. For Azure media, the user name and password are the Storage Account Name and Storage Access Key.

The **STS_PUBLIC** type uses the AWS Security Token Service (STS) to obtain temporary credentials for access to Object Storage in the AWS public cloud. This authentication type applies only to AWS media and requires a role, username and password be specified.

The **STS_GOVCLOUD** type uses the AWS Security Token Service (STS) to obtain temporary credentials for access to Object Storage in the AWS GovCloud. This authentication type applies only to AWS media and requires a role, username and password be specified.

The **CAP** type uses the AWS Commercial Cloud Services (C2S) Access Portal (CAP) to obtain temporary credentials for access to Object Storage in the AWS Private Cloud for the Federal Government. This authentication type applies only to AWS media and requires a role, CAP mission, and CAP agency be specified. The connection endpoint for the CAP server, the location of the client certificate, and either the file or directory path to the CA certificate files must also be specified.

-r *role* For **STS_PUBLIC** and **STS_GOVCLOUD** authentication, use the Amazon Resource Name (ARN) of the role to assume. For **CAP** authentication, use the Identity and Access Management (IAM) role associated with the target C2S account. A role is required by AWS STS and the CAP servers to obtain temporary credentials. This option is valid only with the **STS_PUBLIC**, **STS_GOVCLOUD**, and **CAP** authentication types.

-O *storageclass*

Defines the storage class for AWS, S3COMPAT, AZURE and GOOGLES3 media only. The following storage classes are supported:

azure_block_blob Azure Block Blob storage class. Blobs in this storage class are in either Cool or Hot tier. Cool or Hot tier is an attribute of the container, defined by the user during configuration of the container on Azure. Note: If the Azure lifecycle policies are configured to use the archive tier, then the **azure_block_blob_archive** storage class must be used instead of this one, otherwise files will not be retrievable from the archive tier.

azure_block_blob_archive Data written to this tier must be rehydrated before it can be read. This storage class utilizes the Azure archive tier. It should be used when Azure lifecycle policies are configured to use the archive tier. Note: this tier feature only applies to Azure Blob storage and General Purpose v2 (GPv2) accounts. General Purpose v1 (GPv1) accounts do not support tiering. For more information, check the Azure documentation on Blob storage tiers.

glacier S3 Glacier storage class using standard retrievals to access data from Glacier storage. When restoring from AWS S3 Glacier using standard retrievals, data can typically be accessed within 3-5 hours.

glacier_exp S3 Glacier storage class using expedited retrievals to access data from Glacier storage. When restoring from AWS S3 Glacier using expedited retrievals, data can typically be accessed within 5 minutes.

glacier_bulk S3 Glacier storage class using bulk retrievals to access data from Glacier storage. When restoring from AWS S3 Glacier using bulk retrievals, data can typically be accessed within 5-12 hours.

glacier_deep	S3 Glacier storage class using standard retrievals to access data from Glacier Deep Archive storage. When restoring from AWS S3 Glacier Deep Archive using standard retrievals, data can typically be accessed within 12 hours.
glacier_deep_exp	S3 Glacier storage class using the expedited retrieval option to access data in Glacier Deep Archive storage. This option is not supported by AWS.
glacier_deep_bulk	S3 Glacier storage class using the bulk retrieval option to access data from Glacier Deep Archive storage. When restoring from AWS S3 Glacier Deep Archive using bulk retrievals, data can typically be accessed within 48 hours.
standard	S3 Standard storage class
standard_ia	S3 Infrequent Access storage class

By default, the storage class is set to **standard** for AWS, S3COMPAT, and GOOGLES3 media, and set to **azure_block_blob** for Azure media. **standard** is the only supported storage class for GOOGLES3.

Note, the cost of retrieval from an AWS Glacier storage class will vary by retrieval time, with the quickest retrieval type having the highest cost.

Please refer to provider documentation for detailed information on supported storage classes, including cost and retrieval times.

-T *restoretier*

Specifies the restore tier when rehydrating an object from Azure Archive storage. The following tiers are supported:

hot Rehydrate the Azure Archive object to the *Hot* access tier.

cool Rehydrate the Azure Archive object to the *Cool* access tier.

-Z *authentication_endpoint*

Specifies an authentication endpoint which overrides the default endpoint for the public or GovCloud STS server. This option is valid only with the **STS_PUBLIC** and **STS_GOVCLOUD** authentication types. If the endpoint is newly supported by Amazon, support within the Quantum storage system can be enabled by adding this endpoint along with its region name to the AWS regions configuration file (*/usr/cvfs/config/awsregions.json*).

-C *certfile*

File name to Certificate Authority (CA) certificate(s).

-R *certpath*

Directory path that contains the individual CA certificate files.

-k *clientkeyfile*

Specifies the location of the client private key. This option is required for CAP authentication if the client private key is kept in a separate file rather than included as part of the X.509 client certificate file. This option is also required to authenticate requests to Google Cloud Storage. The specified Google private key file must be in JSON format.

-p *clientkeypass*

Client private key passphrase. This is required if the private key is protected by a passphrase.

-H *CAP_hostport*

Specifies the connection endpoint for the CAP server.

-L *clientcertfile*

The location of the X.509 client certificate installed on the system for the CAP server to authenticate. Note that the certificate should be in PEM format and is required for CAP authentication.

source The full pathname to a file copy on a storage disk whose data will be retrieved.

dest The name of the file where the data is to be written. The file must not exist, otherwise the command will fail.

EXIT STATUS

Exit codes for the **fsmedread**(1) command are:

0 Command completed successfully.

1 Command failed.

2 Command syntax error.

NOTE

The resulting file will end up with the ownership and group of the caller to the program and empty permissions. This is done because no permission, ownership, or group information for files is kept on the source media.

EXAMPLES

Read from Object Storage using the AWS Security Token Service to obtain access to the AWS media:

```
fsmedread -u https://namespace.s3.amazonaws.com/myFile -n namespace \
     -o myFile -t AWS -U username -P password -r role -Y STS_PUBLIC \
     -v AWS -S V4 localFile
```

Read from Object Storage using the AWS Commercial Cloud Services Access Portal to obtain access to the AWS media:

```
fsmedread -u https://namespace.s3.amazonaws.com/myFile -n namespace \
     -o myFile -t AWS -A agency -m mission -r role -Y CAP -v AWS \
     -R /opt/ssl -H capserver -S V4 -L /opt/ssl/client.pem localFile
```

Read from Object Storage using STANDARD authentication to obtain access to the GOOGLES3 media:

```
fsmedread -u https://namespace.storage.googleapis.com/myFile -n namespace \
     -o myFile -t GOOGLES3 -U username -P password \
     -v GOOGLE -S V2 localFile
```

Read from Object Storage using the credentials found in file /key/StorNext-1d5684e5cf83.json to obtain access to the GOOGLE media:

```
fsmedread -u https://storage.googleapis.com/namespace -n namespace -t GOOGLE \
     -o myFile -k /keys/StorNext-1d5684e5cf83.json -v GOOGLE localFile
```

SEE ALSO

fsmedscan(1), **fsmount**(1)

NAME

fsmedscan − Scan SNSM media and report found files.

SYNOPSIS

fsmedscan -h

fsmedscan [**-b**] [**-n** *num*] **-c** *cdbn devpath*

fsmedscan [**-b**] [**-n** *num*] [**-f** *fsn* [**-r** *recs*]] *devpath*

fsmedscan [**-n** *num*] **-L** *devpath*

fsmedscan [**-R** *recoverRoot*] [**-n** *num*] [**-f** *fsn*] *devpath*

fsmedscan [**-d** *dir*] *sdpath*

fsmedscan [**-k** *key* [**-v** *ver*] [**-s** *seg*]] *sdpath*

DESCRIPTION

This utility can be used to scan a Tertiary Manager ANTF formatted tape media or storage disk media. However, this utility is not applicable for Object Storage media and it only supports ANTF formatted tapes. For tape media, it will scan an entire media for session directories and then print the contents of the directory. Each session directory contains a listing of information about all files which were written to the media during the session. For storage disk media, it will scan all the files on the media and print the header label for each file.

This can also be used to perform a detailed scan by using the **-b** option. This form of the scan will search byte by byte looking for file labels that the Tertiary Manager writes to tape before each file. When a label is encountered, the file's position on tape is reported along with some other label information. All data between two tape marks is examined. The **-f** option can be used to indicate where the scan should begin.

This can also be used to read files found on tape by using the **-R** option. This form of the scan will read the tape, temporarily placing file contents in files under the recoverRoot named according to the key. If a matching directory entry is found, the temporary file is renamed to the original path appended to the recoverRoot. If **-n** is specified, it limits the recovery to *num* session directories. If **-f** is specified, it causes an initial forward skip *fsn* file marks. For files with multiple segments, the data for each segment is written to the appropriate offset in the recovered file.

For tape media, it must already be mounted in the drive before this command is run.

TAPE LAYOUT

A formatted media contains at minimum a volume label and a tape mark. It can then contain any number of sessions. The elements of a formatted media are:

Volume Label
Tape Mark
Sessions (0 or more)

A session is the amount of data written to a media by a single I/O process (i.e fs_fmover). Each session consists of a number of file records and ends with a directory. The directory is basically a table of contents which provides information about all the files written during the session. A session contains the following elements:

File Records (1 or more)
Tape Mark
Session Directory
Tape Mark

Each file written during the session is preceded with a label which allows the Tertiary Manager software to verify the file when it is retrieved from the media. A file record contains the following elements:

File Label
File Data

OPTIONS

-h Help. Shows usage.

-b Byte scan. This will search byte by byte for file header labels up to the next tape mark. If **-n** is used the scan will stop after finding the specified number of file header labels. This can be used to verify file position information.

-R *recoverRoot*
Recover scan. This will Recreate files found on tape to the directory tree beginning with the recoverRoot. Files are first created with the name of the file key, then renamed to the original path beneath the *recoverRoot* if a matching session directory is found. If **-n** is specified, it limits the recovery to *num* session directories.

-n *num* The number of session directories or file header labels to find. Default is all except when **-L** is used then it is 1. When scanning for directories, the entire tape is examined and tape marks have no impact. When scanning for file header labels (**-b** option), only data between two tape marks is examined, so the scan is terminated when either a tape mark is encountered or *num* file headers have been detected.

-f *fsn* Specifies File Sequence Number (FSN) to start at. The default is 0 which is the beginning of tape (BOT).

-c *cdbn* Specifies Cumulative Data Block Number (CDBN) to start at.

-r *recs* The number of records to skip from specified FSN. The default is 0. This allows the scanning to start anywhere between tape marks and is only valid with the **-f** option.

-L This will find session directories starting with the last one and working backwards. The default number to find is 1 unless a larger quantity is specified with the **-n** option. The results are displayed in reverse order, i.e. the last one, the one prior to that, etc. This option can be useful when trying to locate files from a StorNext backup prior to performing a **snrestore** operation.

-d *dir* Specifies the starting directory to scan a storage disk. *dir* can be set to any directory below the storage disk's root directory.

-k *key* Scan file copies that belong to a file whose file key equals to *key*.

-v *ver* Scan file copies that belong to the specific file version *ver*. This is only valid with the **-k** option.

-s *seg* Scan file copies whose segment number equals to *seg*. This is only valid with the **-k** option.

devpath
SCSI tape device path.

sdpath Storage Disk directory that is specified when adding a storage disk.

EXIT STATUS

Always 0

SEE ALSO

fsmedread(1), **fsdiskcfg**(1)

NAME

fsmedwrite − Reads a file from disk and writes it to a namespace in the Object Storage system.

SYNOPSIS

fsmedwrite -n *namespace* [**-o** *outfilename*] [**-N]** [**-V**] [**-e** *encryptiontype*] [**-M** *mkeyname*] *sourcefile*

fsmedwrite -u *URL* **-n** *namespace* [**-o** *outfilename*] [**-v** *provider*] [**-t** *mediatype*]
[[**-U** *username* **-P** *password*] | [**-A** *CAP_agency* **-m** *CAP_mission*]] [**-a** *serverAuth*]
[**-e** *encryptiontype*] [**-M** *mkeyname*] [**-S** *signingtype*] [**-Y** *authenticationtype*] [**-r** *role*]
[**-Z** *authentication_endpoint*] [**-H** *CAP_hostport*] [**-C** *certfile* | **-R** *certpath*] [**-L** *clientcertfile*]
[**-k** *clientkeyfile*] [**-p** *clientkeypass*] [**-V**] *sourcefile*

DESCRIPTION

This is a utility used internally by the StorNext backup system to write a file to the Object Storage system without tracking it in StorNext. This utility will take advantage of the Object Storage System configured using **fsobjcfg**(1). When the URL is not specified, the location of the namespace and all security options are located in the TSM database and used for writing the file to the Object Storage System. When the URL is specified, the location of the namespace and the security options are determined from the command options and used for writing the file to the Object Storage System. The namespace option is required.

OPTIONS

-V Used to print more verbose output.

-n *namespace*

The namespace on the Object Storage system to write to. This is a required option if Object Storage is accessed via TSM or the Object Storage media is using S3 REST API.

-o *outputfilename*

If a specific name is wanted on the Object Storage system, this can be specified. Otherwise the name will be that of the source file. This is required if the URL is specified and the Object Storage media is using S3 REST API.

-u *URL* The URL to the Object Storage System. A complete connection endpoint address including the protocol. Example https://192.168.21.111:444

-v *provider*

Identifies the provider of the appliance to which the file is to be written. The appliance provider determines the format of provider-specific REST headers. The following provider types are supported by Tertiary Manager software:

AWS
AZURE
GOOGLE
LATTUS
S3COMPATIBLE

-t *mediatype*

The object storage media type. The following object storage media types are supported by Tertiary Manager software:

AWS
AXR
AZURE
GOOGLE
GOOGLES3
S3
S3COMPAT

The default is **AXR**.

-a *serverAuth*

The server authorization setting. Default is 2. The valid options are:

0 none

1 verify peer only

2 verify peer and host.

-S *signing_type*

Used to specify the signing type for the requests sent to the Object Storage and/or authentication server. The following signing types are supported:

V2
V4
AZURE

The default is **V4** for AWS and S3 compatible media, **V2** for GOOGLES3 and Lattus S3 media, and **AZURE** for Azure media. This option is not valid for AXR and Google media. When configuring V4, chunked mode is preferable and should be used if supported by the object storage system. For object storage which does not support chunked mode, such as AWS Snowball, you must specify the full payload mode signing option, **-J**, for the object storage bucket.

-Y *authentication_type*

An authentication type is required for all Object Storage media except for AXR and GOOGLE. The following authentication types are supported:

STANDARD
STS_PUBLIC
STS_GOVCLOUD
CAP

If this option is not specified, the authentication type will be set to **STANDARD**.

The **STANDARD** type applies to all media and authenticates with a user name and password for Object Storage access. For S3 compatible media the user name and password are the Access Key Id and Secret Access Key. For Azure media, the user name and password are the Storage Account Name and Storage Access Key.

The **STS_PUBLIC** type uses the AWS Security Token Service (STS) to obtain temporary credentials for access to Object Storage in the AWS public cloud. This authentication type applies only to AWS media and requires a role, username and password be specified.

The **STS_GOVCLOUD** type uses the AWS Security Token Service (STS) to obtain temporary credentials for access to Object Storage in the AWS GovCloud. This authentication type applies only to AWS media and requires a role, username and password be specified.

The **CAP** type uses the AWS Commercial Cloud Services (C2S) Access Portal (CAP) to obtain temporary credentials for access to Object Storage in the AWS Private Cloud for the Federal Government. This authentication type applies only to AWS media and requires a role, CAP mission, and CAP agency be specified. The connection endpoint for the CAP server, the location of the client certificate, and either the file or directory path to the CA certificate files must also be specified.

-r *role* For **STS_PUBLIC** and **STS_GOVCLOUD** authentication, use the Amazon Resource Name (ARN) of the role to assume. For **CAP** authentication, use the Identity and Access Management (IAM) role associated with the target C2S account. A role is required by AWS STS and the CAP servers to obtain temporary credentials. This option is valid only with the **STS_PUBLIC**, **STS_GOVCLOUD**, and **CAP** authentication types.

-Z *authentication_endpoint*
Specifies an authentication endpoint which overrides the default endpoint for the public or GovCloud STS server. This option is valid only with the **STS_PUBLIC** and **STS_GOVCLOUD** authentication types. If the endpoint is newly supported by Amazon, support within the Quantum storage system can be enabled by adding this endpoint along with its region name to the AWS regions configuration file (*/usr/cvfs/config/awsregions.json*).

-N Disable tiering to Azure Archive storage. This option is ignored for media types other than Azure.

-C *certfile*
File name to Certificate Authority (CA) certificate(s).

-R *certpath*
Directory path that contains the individual CA certificate files.

-U *username* **-P** *password*
Username and password to access the URL if necessary. These are required if the Object Storage media is using S3 REST API.

-A *CAP_agency* **-m** *CAP_mission*
The agency and mission associated with the target C2S account. These options are valid only for the CAP authentication type.

-k *clientkeyfile*
Specifies the location of the client private key. This option is required for CAP authentication if the client private key is kept in a separate file rather than included as part of the X.509 client certificate file. This option is also required to authenticate requests to Google Cloud Storage. The specified Google private key file must be in JSON format.

-p *clientkeypass*
Client private key passphrase. This is required if the private key is protected by a passphrase.

-H *CAP_hostport*
Specifies the connection endpoint for the CAP server.

-L *clientcertfile*
The location of the X.509 client certificate installed on the system for the CAP server to authenticate. Note that the certificate should be in PEM format and is required for CAP authentication.

-e *encryptiontype*
The encryption algorithm to be applied to the content of files stored into an Object Storage system. This option is ignored for all other media types. Currently, the following encryption types are supported:

none	No encryption
SSE_S3	Server-side AES256 encryption with S3-managed keys
SSE_KMS	Server-side AES256 encryption using the AWS Key Management Service (KMS)

For the **SSE_KMS** encryption type, the default customer master key (CMK) can be used for encryption or a CMK can be created using the AWS Key Management Service and specified with the **-M** option.

If this option is not specified, the encryption type will be set to **none**.

-M *mkeyname*
The Master Key name that has been created using the AWS Key Management Service and requested for server-side encryption.

sourcefile
The file to write to the Object Storage system.

EXIT STATUS

Exit codes for the **fsmedwrite**(1) command are:

0 Command completed successfully.

2 Command failed.

3 Command syntax error.

EXAMPLES

Here are examples of typical usage:

Basic write using information from TSM database

```
fsmedwrite -n ns2 localFile
```

Basic write using information from TSM database. Specifying a different name to use in the Object Storage system

```
fsmedwrite -n ns2 -o myFile2 /tmp/localFile
```

Write using command line URL

```
fsmedwrite -u http://192.168.21.111:8080 -n ns2 -o myFile2 /tmp/localFile
```

Write using command line URL, https security, and specific Certificate Authority Cert file

```
fsmedwrite -u https://192.168.21.111:444 -n ns2 -o myFile2 \
     -a 2 -C /opt/ssl/certfile /tmp/localFile
```

Write using command line URL, https security, and Certificate Authority Cert directory:

```
fsmedwrite -u https://192.168.21.111:444 -n ns2 -o myFile2 \
     -a 1 -R /opt/ssl /tmp/localFile
```

Write using command line URL, https security, Certificate Authority Cert directory, and username/password:

```
fsmedwrite -u https://192.168.21.111:444 -n ns2 -o myFile2 \
     -a 1 -R /opt/ssl -U user -P password /tmp/localFile
```

Write to Object Storage via S3 REST API using command line URL:

```
fsmedwrite -u http://192.168.21.111:7070 -n bucket1 -o myFile2 \
     -t S3 -U user -P password /tmp/localFile
```

Write to Object Storage using the AWS Security Token Service to obtain access to the AWS media:

```
fsmedwrite -u https://namespace.s3.amazonaws.com -n namespace \
     -o myFile2 -t AWS -U username -P password -r role       \
     -Y STS_PUBLIC -v AWS -S V4 /tmp/localFile
```

Write to Object Storage using the AWS Commercial Cloud Services Access Portal to obtain access to the AWS media:

```
fsmedwrite -u https://namespace.s3.amazonaws.com -n namespace     \
     -o myFile2 -t AWS -A agency -M mission -r role -Y CAP -v AWS \
     -R /opt/ssl -H capserver -S V4 -L /opt/ssl/client.pem       \
     /tmp/localFile
```

Write using command line URL to Azure media:

```
fsmedwrite -u https://blob.core.windows.net -n namespace \
     -o myFile2 -t AZURE -v AZURE -U user -P password /tmp/localFile
```

Write using command line URL to GOOGLES3 media:

```
fsmedwrite -u https://namespace.storage.googleapis.com -n namespace \
     -o myFile2 -t GOOGLES3 -v GOOGLE -U user -P password /tmp/localFile
```

Write to Object Storage using the credentials found in file /key/StorNext-1d5684e5cf83.json to obtain access to GOOGLE media:

```
fsmedwrite -u https://storage.googleapis.com/namespace -n namespace    \
     -o myFile -t GOOGLE -v GOOGLE -k /keys/StorNext-1d5684e5cf83.json  \
     /tmp/localFile
```

SEE ALSO

fsobjcfg(1), **fskey**(1)

NAME

fsmodclass − Modify the processing parameters of a policy class.

SYNOPSIS

fsmodclass

[**-t** *mediatype*|**-C** *copy*] [**-T ANTF**|**LTFS**|**NONE**] [**-L** *drivelimit*]
[**-E** *expirationtime* [**-X**]] [**-P y**|**n**]
[**-s** *softlimit*] [**-h** *hardlimit*] [**-S** *stubsize*]
[**-x** *maxcopies*] [**-d** *defaultcopies*]
[**-m** *minstoretime*] [**-c** *mintrunctime*] [**-a** *affinity*...**]**
[**-i** *minreloctime*] [**-R** *affinity*] [**-v** *drivepool*] [**-k** *maxversions*]
[**-f i**|**p**] [**-W** *clnvertime*] [**-r c**|**s**] [**-p yes**|**no**]
[**-z** *minsetsize* -g *maxsetage***]**
[**-A y**|**n**] [**-G y**|**n**] [**-V y**|**n**] [**-D y**|**n**] [**-K y**|**n**] [**-H y**|**n**] [**-F** *type*]
[**-e** *encryptiontype*] [**-M** *mkeyname*]
[**-O** *retrieveorder*] [**-U** *putstreams*] [**-I** *getstreams*]
[**-Z** *multistreamsize*] [**-Q y**|**n**] *class*

DESCRIPTION

The **fsmodclass**(1) command allows the system administrator to modify the processing parameters of a particular policy class in the Tertiary Manager software. For each of the optional arguments that are not entered, those policy class processing parameters will be left unmodified.

This command accepts upper case or lower case input. However, most input is converted to lower case, except for the security level, and account number.

The administrator can view the current parameter settings by using the **fsclassinfo**(1) command.

NOTE: At least one option must be specified for the **fsmodclass**(1) command. The **-C** option requires one or more of the following options to be specified: **-E**, **-I**, **-L**, **-P**, **-Q**, **-T**, **-U**, or **-Z**.

OPTIONS

class policy class. A policy class name can have a maximum of 16 alphanumeric characters. The special "-", ".", and "_" characters are also permitted.

-a *affinity*...

A space-separated list of disk affinities that the files in this policy class will traverse throughout their life cycle. Valid entries include any of the affinities already configured in the managed file systems or the word *none*. The first affinity in this list will be considered the default disk affinity for this policy class, the affinity in which files initially will be created. When they become eligible candidates for relocation, the files will be moved to the next disk affinity in the list. The word *none* may be specified in place of the affinity list to indicate no automatic relocation is to occur for this policy class. NOTE: Presently a maximum of two affinities is supported, including the default disk affinity. WARNING: When removing an affinity from the list, bear in mind that files in this policy class may reside on that disk affinity. To avoid unwanted behavior, these files may be manually relocated to another disk affinity using **fsrelocate**(1) command.

-R *affinity*

The affinity to retrieve a truncated file to. This will override the default affinity. If *affinity* is set to *none* then a truncated file will be retrieved to the default affinity.

Note that Storage Manager must be stopped and restarted after using the **-R** option before the new affinity selection will take effect.

-d *defaultcopies*

The total number of copies that will be stored (including the primary copy) for each file in this policy class. The *defaultcopies* option can be set equal to, but not exceeding, the *maxcopies* setting. It cannot be set to less than one.

-f i|p The file retention policy for the policy class. The files can be truncated immediately (**i**) or at policy application time (**p**) once all file copies are stored on a medium.

-h *hardlimit*

The maximum number of media that are allowed in this policy class. When hard limit is reached, a warning message is sent to the *syslog* and to the user specified as the e-mail contact for this policy class. Files can still be stored in that policy class, as long as there is room on the media that are already been to store files in that policy class.

-c *mintrunctime*

The minimum time that a stored file must reside unaccessed on disk before being considered a candidate for truncation (the clearing of disk blocks). A file will not have its disk blocks truncated (by a truncation policy) until it has remained unaccessed on disk for this amount of time. After that time, a truncation policy will consider the file a valid candidate for truncation, but it may or may not actually be truncated. That will depend on the current file system fill level and the file system configuration parameters. NOTE: An "emergency" truncation policy ignores this time. The minimum value allowed for this time is 5 minutes. See SETTING CLASS TIMES below for more info on time format and usage.

-W *clnvertime*

Each class can have an *clnvertime* interval for the cleanup of old versions of files within the class by the **fsclean**(1) command. This is in addition to a system-wide cleanup interval that is described in the **fsschedule**(1) man page. The interval is up to five digits plus a single-letter interval factor for days (**d**), weeks (**w**), months (**m**), or years (**y**). Setting the interval to zero turns off per-class cleanup and leaves the files in the class subject to the system-wide cleanup interval.

-i *minreloctime*

The minimum time that a file must reside unaccessed on disk before being considered a candidate for relocation (the moving of data blocks from one disk affinity to another). A file will not have its data blocks relocated (by a relocation policy) until it has remained unaccessed on disk for this amount of time. After that time, a relocation policy will consider the file a valid candidate for relocation. The file may or may not actually be relocated at that time, depending on the current file system fill level and the file system configuration parameters. NOTE: An "emergency" relocation policy ignores this time. The minimum value allowed for this time is 5 minutes. See SETTING CLASS TIMES below for more info on time format and usage.

-m *minstoretime*

The minimum time that a file must reside unmodified on disk before being considered a candidate for storage on media. A file will not be stored (by a store policy) until it has remained unmodified on disk for this amount of time. After that time, the next policy run will attempt to store the file. The minimum value allowed for this time is 1 minute. See SETTING CLASS TIMES below for more info on time format and usage.

-S *stubsize*

The truncation stub size (in kilobytes). This value is used to determine the number of bytes to leave on disk when files are truncated. This value will be the minimum number of bytes left on disk (the value will be rounded up to a multiple of the file system block size). This value is not used when a stub size has been explicitly set for a file with **fschfiat**(1). If the **-S** option is not used, the default will be specified by the system parameter CLASS_TRUNCSIZE. **-r c|s** Media classification cleanup action. When all files are deleted from a medium, the medium can revert back to the policy class blank pool (**c**) or to the system blank pool (**s**).

-s *softlimit*

The warning limit for the number of media that can be allocated to this policy class. When the soft limit is reached, a warning message is sent to the *syslog* and to the user specified as the e-mail contact for this policy class. The value of *softlimit* must be less than or equal to the value of *hardlimit*.

-t *mediatype*

Defines the type of medium to be used. Depending on the type of platform used, the following media types are supported by Tertiary Manager software:

AWS
AZURE
LATTUS
GOOGLE
GOOGLES3
S3
S3COMPAT
LTO
LTOW
3592
T10K
SDISK

If this policy class is being modified so that it only supports disk-to-disk relocation, then **DISK** is also allowed. The **-a** option must also be specified unless the policy class is already configured with disk affinities.

Changing the media type will change the media format type to a supported value, if necessary, for each copy not overridden in the *filesize.config* file, according to the rules for determining a default value listed under the **-T** option. The **-t** option is not allowed with the **-C** option.

The **fsstore**(1) command can override this parameter with the **-t** option.

-v *drivepool*

The Media Manager drive pool group used to store or retrieve data for this policy class. This drive pool name must be defined within the Media Manager software before any media operations can occur for this policy class. The special "_" character is permitted to identify the drive pool group.

-x *maxcopies*

The maximum number of copies (including the primary copy) that are allowed for each file in this policy class. The *maxcopies* value cannot be less than one. NOTE: If the copy setting for a particular file is adjusted using the **fschfiat**(1) command, it cannot exceed the value defined by the *maxcopies* setting.

-k *maxversions*

This is the maximum number of inactive versions to keep for a file (the current version is active, all others are inactive). When a stored file is modified, the version number is immediately incremented for the file. Old versions are kept around until the **fsclean**(1) command is used to clean those versions from Tertiary Manager media. At the time a new version is stored, the oldest version will be deleted if maxversions has been reached. The **fsclean**(1) command will not be required to clean that version. If maxversions is defined as 0, only the active version will be retained and if removed from the file system, it will still be recoverable unless the **-D** option is set on the file.

When *maxversions* is changed for a class, it will not take effect for up to 60 minutes due to caching. This cache timeout can be adjusted using the CPYRESP_MAXVER_QRY_SECS sysparm.

-p yes|no

This option decides if we allow the policy engine to automatically store files for this policy class.

-z *minsetsize*

The minimum set size of the policy class. It will be specified in the form of [0-999][MB|GB]. When this option is specified with a non-zero value, the store candidates in the policy class have to add up to *minsetsize* before any of them can be stored by the policy engine. This option works in conjunction with **-g** option. Specifying zero value for this option will disable the check, which re-

quires **-g** be set to zero as well. **-g** option.

-g *maxsetage*

Candidate expiration time (in hours) of the policy class. This option works in conjunction with **-z** option so that files will not sit forever on the candidate list because the *minsetsize* has not been reached. As soon as any file on the store candidate list in the policy class is *maxsetage* old, all of the files should be stored. Specifying zero will disable the check, which requires **-z** be set to zero as well. The maximum value for this option is 720 hours, which equals to 30 days.

-C *copy*

Used to apply the copy options to a specific copy. The **-C** option is required for the **-E**, **-Q** or **-P** options. When the **-C** option is not specified for the **-I**, **-L**, **-T**, **-U**, or **-Z** options, those options apply to all copies. The **-C** option requires one or more of the options: **-E**, **-I**, **-L**, **-P**, **-Q**, **-T**, **-U**, or **-Z**. The **-C** option is not allowed with the **-t** option.

-T ANTF|LTFS|NONE

Specifies the media format type that will be used when formatting or selecting media. **ANTF** is the Quantum internal tape format. **LTFS** is the Linear Tape File System specification tape format. **NONE** is used for Object Storage media since no special formatting is done. A value of **ANTF** will format media with a single partition containing ANTF volume labels. This data partition will store StorNext file data. A value of **LTFS** will format media with two partitions containing LTFS volume labels. The index partition will store LTFS metadata and the data partition will store StorNext file data. The **LTFS** media format type is only applicable to LTO media generation 5 and beyond (e.g. LTO-5, LTO-6, etc). Any attempt to assign **LTFS** to non-LTO media types will result in an error. The backup policy class _adic_backup does not support a media format type of **LTFS**. Any attempt to assign **LTFS** to _adic_backup will result in an error.

If neither of the **-T** or **-t** options have been specified, no changes will be made to the existing media format type settings.

If the **-T** option is not specified and the **-t** option is specified, the default value will be determined as follows:

A media type of Object Storage will default to **NONE**.
A media type that does not support **LTFS** will default to **ANTF**.
A media type that supports **LTFS** will default to the system parameter CLASS_MEDIA_FORMAT.

Whenever the **-T**, **-t**, or **-x** options are specified, the *filesize.config* file will be validated for copies to ensure that at least one of the file size range media types supports the configured media format type for that specific copy. Any media types that do not support the configured media format type will be overridden to a supported media format type during file store processing.

-L *drivelimit*

The maximum number of drives to use when the policy is run. Specifying **none** will disable the drive limit. If the **-L** option is not specified, no changes will be made to the existing drive limit settings.

-E *expirationtime*

The minimum time per copy that a stored file must reside without being accessed before the copy becomes eligible for deletion. The **-C** option must be specified with this option. The default value of 0 (zero) indicates that the copy is not eligible for expiration. Expiration time can be set on any copy, but at least one copy must be configured as ineligible for expiration. The default unit for the copy expiration value is days. See SETTING CLASS TIMES below for more info on time format and usage.

Caution: When a copy expiration criteria is added to a policy class, it will not apply to files that have already been stored. To apply the copy expiration criteria to those files you will need to call the following command:

fspolicy -b -y *filesystem mount point* **-C**

-X The **-X** option can be used to override the restriction where the user is not allowed to configure a copy expiration time on all configured copies. The **-C** option must be specified with this option, and the **-E** option must also be specified with an expiration time greater than 0. When the last copy of a file expires and is removed, the file is automatically removed from the file-system name space and will no longer be recoverable. WARNING: If you select to expire all copies, and when the last copy of a file expires, the file and all stored copies are removed. The file is DELETED from the file system and it is DELETED from all storage destinations. This operation cannot be undone; you are no longer able to retrieve the file and the file is no longer accessible at all from any location.

-P y|n Indicates if copies are to be recreated when referenced after being expired. The default value **n** specifies that the copy is not to be recreated. The **-P** option is not allowed with the **-X** option or when the **-E** is being set to 0.

-O *retrieveorder*
A comma-separated list of copy numbers specifying the order in which copies will be selected when retrieving files. Specifying **none** will re-enable the default retrieve order. If the **-O** option is not specified, the default retrieve order will be used.

-G y|n Generate and maintain a checksum for each stored file. If this option is enabled, a checksum will be generated as each file is written (stored) to the associated protection tier. The checksum will be retained for use in the checksum-validation phase of subsequent retrieve operations. This option can be overridden by the FS_GENERATE_CHECKSUM parameter in *fs_sysparm*. See *fs_sysparm.README* file for details. The **fsfileinfo**(1) command can be used to determine if a file has a corresponding retained value.

-V y|n Verify the checksum of each retrieved file. If this option is enabled, a checksum will be generated as each file is read (retrieved) from the associated protection tier. The generated checksum will be compared to the corresponding (added when the file was stored) retained value. The retrieve will fail and a RAS ticket will be opened if the checksum does not match. No compare will occur if no retained value exists (checksum generation was not enabled when file was stored). This option can be overridden by the FS_VALIDATE_CHECKSUM parameter in *fs_sysparm*. See *fs_sysparm.README* file for details.

-D y|n Remove database information when a file is removed. If this option is enabled, then when a file is removed from the file system, the corresponding database information indicating where the file was stored will also be removed, and the file will NOT be recoverable through **fsrecover**(1). If this option is disabled, then the database entries will be retained and the file is recoverable through **fsrecover**(1).

-H y|n Enable (**y**) or disable (**n**) Dynamic Library Pooling for this policy class. Default value is **n**. If enabled, the Tertiary Manager will direct store requests to specific drive pools on a rotating basis. DLP drive pool sets are configurable for each copy number via the DLP_COPY*_DRIVEPOOL_SET sysparms. See *fs_sysparm.README* for details. If disabled, no drive pool rotation will be performed, and the default drive pool will be used for store requests.

-Q y|n When storing to an object storage system, enabling this option will cause metadata, including the copy number, last modification time, file offset, file key, path, segment number, and version, to be stored along with the object. File metadata can be added or updated using **fsobjmeta**(1). Note, the path will be URL encoded if it contains non-ASCII characters.

The **-C** option must be specified with this option.

-K y|n This indicates that default retry logic will not be peformed for retrieve failures. It can be useful for applications that have their own specialized retry logic based on the type of the data being stored (such as erasure coded data). **-A y|n** Enable Alternate Store Location support. If this option is enabled, files created under this policy class will be eligible for an additional remote copy to be made to the configured Alternate Store Location. Note: This is a licensed feature.

-e *encryptiontype*

The encryption algorithm to be applied to the content of files stored into an Object Storage system. This option is ignored for all other media types. Currently, the following encryption types are supported:

none	No encryption
SSE_S3	Server-side AES256 encryption with S3-managed keys
SSE_KMS	Server-side AES256 encryption using the AWS Key Management Service (KMS)

For the **SSE_KMS** encryption type, the default customer master key (CMK) can be used for encryption or a CMK can be created using the AWS Key Management Service and specified with the **-M** option.

If this option is not specified, the encryption type will be set to **none**.

-M *mkeyname*

The Master Key name that has been created using the AWS Key Management Service and requested for server-side encryption.

-U *putstreams*

Defines the number of streams to use for stores when a file in the request is equal to or larger than the size defined with option **-Z**. The value must be in the range of 0 and 64.

-I *getstreams*

Defines the number of streams to use for retrieves when a file in the request is equal to or larger than the size defined with option -Z. The value must be in the range of 0 and 64.

-Z *multistreamsize*

Defines the minimum file size for using multiple streams.The value of *multistreamsize* cannot be less than 20MiB. It can include one of the following suffixes:

B	bytes
KB	kilobytes (1000)
KiB	kibibytes (1024)
MB	megabytes (1000^2)
MiB	mebibytes (1024^2)
GB	gigabytes (1000^3)
GiB	gibibytes (1024^3)
TB	terabytes (1000^4)
TiB	tebibytes (1024^4)

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-inde-

pendent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SETTING CLASS TIMES

The policy time values associated with a class: *minstoretime*, *mintrunctime* and *minreloctime* can all be set in units of minutes, hours or days. To specify minutes put an '**m**' suffix on the value, to specify hours put an '**h**' suffix on the value and for days use a '**d**' suffix. Note that if the unit suffix is not specified the *minstoretime* value defaults to units of minutes while the others default to units of days. Some valid examples for policy times:

15m - 15 minutes
3h - 3 hours
7d - 7 days
10 - 10 minutes for *minstoretime* and 10 days for the other times

Store policy commands are kicked off automatically every minute or so as long as files are being created. Therefore setting a value for *minstoretime* to just a few minutes will usually result in a store policy command starting the store operation for the files in a class within the time requested. Note however if files are being created slowly the software may wait up to 5 minutes before kicking off a store policy in an attempt to get a larger number of files to store at one time. Also be aware that other factors like system load, media availability etc can affect the time the stores will actually occur.

Truncation and relocation policies are only run automatically once a day at midnight. (There are also policies that run as file system fill levels warrant but the time when these occur is not scheduled. See the **filesystems**(4) man page for more info on the space based policies.) If there is a real need to have file data truncated at a specific time after last access then two steps are necessary, the setting of the class time and the scheduling of the policy commands. As an example let us assume you want to truncate class data for class1 after being unaccessed after one hour. The first step would be to set the *mintrunctime* to **1h** (or **60m**). Next you must schedule via cron a truncation policy to run every half hour. (Note here that even with policies running every half hour; a file may wait up until the time between policies beyond the trunctime before truncation occurs. For example if policies run on the half hour, a file is created at 01:00:01am, it will not be truncated until 2:30am since at 2:00am it is 1 second short of being an hour old.)

When setting up the policy cron job it is probably easiest to set up a simple shell script to wrap the policy so that the processing environment is set up correctly. For example set up a script under TSM: */usr/adic/TSM/util/truncPolicy* The contents of the script may look like:

```
#!/bin/sh
#
. /usr/adic/.profile
/usr/adic/TSM/exec/fspolicy -t -c class1 -o 0
```

Note the last argument to the policy command '**-o 0**'. This tells the policy to keep truncating files until it runs out of candidates or the file system reaches 0 percent full. If you look at the **filesystems**(4) man page it indicates the automatic nightly policy only truncates to the Min Use percentage and then quits even if more valid candidates are present. If the desire is to truncate all candidates then the '**-o 0**' is needed. Truncating all files for a class should be done carefully as there will be an expense to retrieving those files back if needed.

Only one truncation policy is allowed to run at a time. This is due to potential conflicts in candidate management between policies and also for performance reasons. If it is desired

to run multiple policies 'at the same time' then just put multiple policies in the truncate script and have them run sequentially. Be sure and not place an ampersand after the commands or some will fail because they are locked out by the currently running policy. Also be sure when setting up the cron jobs not to have the scheduled scripts run too close together. The User's Guide contains more info on the scheduling of truncation policies. An example of a script with multiple scheduled policies would be:

```
#!/bin/sh
#
. /usr/adic/.profile
/usr/adic/TSM/exec/fspolicy -t -c class1 -o 0
/usr/adic/TSM/exec/fspolicy -t -c class2 -o 0
/usr/adic/TSM/exec/fspolicy -t -c class3 -o 0
```

The next step is to create the actual cron entry. Run **crontab -e** and set the entry to look something like this:

```
00,30 * * * * /usr/adic/TSM/util/truncPolicy
```

One last thing to note on scheduling 'extra' truncation or relocation policies: there is an expense to running these commands as they get and check their candidate lists, even if no files are actually truncated or relocated. This is especially true for sites where millions of files are resident on disk at one time. See the User's Guide for more recommendation info on the scheduling of these policies.

SEE ALSO

filesystems(4), **fsaddclass**(1), **fsrmclass**(1), **fsclassinfo**(1), **fsclassrnm**(1), **fsstore**(1), **fsversion**(1), **fsclean**(1), **fsschedule**(1), **fsfileinfo**(1), **fschfiat**(1), **fspolicy**(1), **fskey**(1), **fsobjmeta**(1),

NAME

fsmount − Mounts specified media

SYNOPSIS

fsmount [**-v** *drivepool*] *mediaID*

DESCRIPTION

The **fsmount**(1) will allow any media known to the Media Manager to be mounted. A media mounted using this command is under the control of the user or application which mounted it and will not be dismounted until the

fsdismount(1) command is issued or the Tertiary Manager software is restarted.

OPTIONS

-v *drivepool*

Media Manager drive pool group used to mount the media. The drive pool must be defined in Media Manager software. If the **-v** option is not used, the default drive pool group will be used.

mediaID The media identifier that is to be mounted.

NOTES

This command is not valid for storage disks.

SEE ALSO

fsdismount(1)

NAME

fsmoverpt − Generate a report on media that were removed from or introduced into Quantum storage subsystems.

SYNOPSIS

fsmoverpt [**-aboi**] [*mediaID...*] [**-f** *logfilename*] [**-t** *starttime* [*endtime*]]

DESCRIPTION

The **fsmoverpt**(1) command generates a report of media that were moved into or out of the Quantum system using the **fsmedin**(1) and **fsmedout**(1) commands. The historical data for the media status is maintained in the $FS_HOME/logs/history/hist_03 file. This is the default log file used, however a different log file can be specified using the **-f** option.

Any number of category options can be entered. If no options are entered, all media movement types are listed. The softcopy report is sent to *stdout* and can be redirected to a file or piped to a printer.

The time range option (**-t** *starttime*) extracts only information stored in the log over the time frame specified. If specified, the *starttime* and *endtime* must be before the current time, and the *starttime* must be before the *endtime*. If no *endtime* is specified, *endtime* defaults to current time.

OPTIONS

-a Generate an historical report listing blank media that were added.

-b Generate an historical report listing blank media that were removed.

-o Generate an historical report listing media that were checked out (removed with the information retained).

-i Generate an historical report listing media that were checked in.

mediaID...

One or more media identifiers. Multiple media identifiers must be separated by spaces. The number of media identifiers that can be entered is limited by the local machine's command-line length.

-t *starttime* [*endtime*]

Time range on which to report. If no end time is entered, the end time defaults to the current time. The format for the time parameter is MM:DD:hh:mm:ss, where:

MM = Numeric, two-digit month
(default:01 (January)),

The following are optional; defaults are shown:,
DD = Numeric, two-digit day
(range: 01-31, default:01),
hh = Numeric hour
(range: 00-23, default:00),
mm = Numeric minute
(range: 00-59, default:00),
ss = Numeric second
(range: 00-59, default:00)

-f *logfilename*

File name of a Tertiary Manager log file - The full file path name does not have to be entered. If preceded by a slash (/) in the command definition, the full path name, starting from the root directory, is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent.

SEE ALSO

fsmedout(1), **fsmedin**(1)

NAME

fsobjcfg − Configure or report all Object Storage components in Quantum storage system.

SYNOPSIS

fsobjcfg -a -i *host_address* [**-p** *port_number*] [**-e http**|**https**] [**-U** *username* **-P** *password*] [**-v** *provider*] [**-B**] *appliance_alias*

fsobjcfg -m [**-i** *host_address*] [**-p** *port_number*] [**-e http**|**https**] [**-U** *username* [**-P** *password*]] [**-B**] *appliance_alias*

fsobjcfg -d *appliance_alias*

fsobjcfg -a -n *controller_alias* [**-s** *streams[:readstreams]*] [**-B**] *appliance_alias*

fsobjcfg -m -s *streams[:readstreams]* [**-B**] **-n** *controller_alias*

fsobjcfg -d -n *controller_alias*

fsobjcfg -a -o *iopath_alias* **-i** *connection_endpoint* [**-e http**|**https**] [**-t** *mediatype*] [**-u PATH**|**VHOST**] [**-B**] **-n** *controller_alias*

fsobjcfg -m -o *iopath_alias* [**-i** *connection_endpoint*] [**-e http**|**https**] [**-t** *mediatype*] [**-u PATH**|**VHOST**] [**-B**] **-n** *controller_alias*

fsobjcfg -d -o *iopath_alias* **-n** *controller_alias*

fsobjcfg -a -b *namespace* [**-c** *copy*] [**-C** *class*] [**-f** *mediaid*] [**-t** *mediatype*] [**-U** *username* **-P** *password*] [**-K** *keyfile*] [**-I** *uuid*] [**-S** *signing_type*] [**-N** *object_prefix*] [**-O** *storageclass*] [**-D** *duration*] [**-Y** *authentication_type*] [**-G y**|**n**] [**-J y**|**n**] [**-B**] *appliance_alias*

fsobjcfg -a -b *namespace* **-t AWS -Y STS_PUBLIC**|**STS_GOVCLOUD -R** *role* **-U** *username* **-P** *password* [**-c** *copy*] [**-C** *class*] [**-f** *mediaid*] [**-D** *duration*] [**-N** *object_prefix*] [**-O** *storageclass*] [**-S** *signing_type*] [**-Z** *authentication_endpoint*] [**-G y**|**n**] [**-J y**|**n**] [**-B**] *appliance_alias*

fsobjcfg -a -b *namespace* **-t AWS -Y CAP -R** *role* **-A** *CAP_agency* **-M** *CAP_mission* [**-c** *copy*] [**-C** *class*] [**-f** *mediaid*] [**-D** *duration*] [**-N** *object_prefix*] [**-O** *storageclass*] [**-S** *signing_type*] [**-G y**|**n**] [**-J y**|**n**] [**-B**] *appliance_alias*

fsobjcfg -m -f *mediaid* [**-b** *namespace*] [**-c** *copy*] [**-C** *class*] [**-t** *mediatype*] [**-U** *username*] [**-P** *password*] [**-K** *keyfile*] [**-D** *duration*] [**-N** *object_prefix*] [**-O** *storageclass*] [**-S** *signing_type*] [**-Y** *authentication_type*] [**-G y**|**n**] [**-J y**|**n**] [**-B**] [**-X**] [appliance_alias]

fsobjcfg -m -f *mediaid* **[-t AWS] [-Y STS_PUBLIC**|**STS_GOVCLOUD]** [**-b** *namespace*] [**-c** *copy*] [**-C** *class*] [**-U** *username*] [**-P** *password*] [**-R** *role*] [**-D** *duration*] [**-N** *object_prefix*] [**-O** *storageclass*] [**-S** *signing_type*] [**-Z** *authentication_endpoint*] [**-G y**|**n**] [**-J y**|**n**] [**-B**] [appliance_alias]

fsobjcfg -m -f *mediaid* **[-t AWS] [-Y CAP]** [**-b** *namespace*] [**-c** *copy*] [**-C** *class*] [**-A** *CAP_agency*] [**-M** *CAP_mission*] [**-R** *role*] [**-D** *duration*] [**-N** *object_prefix*] [**-O** *storageclass*] [**-S** *signing_type*] [**-G y**|**n**] [**-J y**|**n**] [**-B**] [appliance_alias]

fsobjcfg -d -f *mediaid*

fsobjcfg -r -f *mediaid*

fsobjcfg -q -f *mediaid*

fsobjcfg -V *awsregionsfile*

fsobjcfg [**-l**] [**-F** *type*]

DESCRIPTION

The **fsobjcfg**(1) command adds, modifies, deletes, and reports configuration settings for Object Storage components in the Quantum storage system. Object Storage components are: Appliances, Controllers, IO Paths, and Namespaces. These components and their attributes provide the addressing information required to form the URL to store and retrieve objects from the Object Storage.

The URL to the Objects in the Object Storage consists of an Internet Address and optional TCP/IP Port Number, and the Namespace. The IO Path is a complete TCP/IP connection endpoint. The Namespace component provides the location within Object Storage where the data is stored. A URL looks like :

AXR http://10.64.68.22:8090/namespace/bucketname/objectid

S3 http://10.64.68.22:7070/bucketname/objectid

Object Storage Namespaces are used by the Tertiary Manager software as secondary storage similar to tape or Storage Disk media.

For any **fsobjcfg**(1) command option, the Tertiary Manager software can be active or inactive.

Submitting the **fsobjcfg**(1) command with no options or **-l** option generates a report showing all Quantum Object Storage components that are currently configured.

The **fschstate**(1) command can be used to change the state of an Object Storage component once it has been configured.

OPTIONS

-a Add a new Object Storage component. If the command is invoked and any of the required parameters are omitted, an error message is returned. The error message identifies the field that was not entered.

-m Modify an Object Storage component. The component alias cannot be modified.

-d Delete an Object Storage component. A component cannot be deleted if it has children components still configured. A namespace component can not be deleted if it contains any stored copies. The **fsrminfo**(1) command can be used to purge any data prior to removing a namespace component from the configuration.

-r Refresh the space available capacity for the specified media. It is typically used to reenable storage of data after an out-of-space condition has been encountered with a media and actions have already been taken to address the out-of-space issue on the appliance.

-q Run a connectivity test to the specified media to ensure that the media is reachable and that the authentication, connection, and signing information configured for the media is valid. This test will also return the Object Storage system's server header string, which usually includes the server name and version, if available.

appliance_alias

Appliance Alias. This is a variable string of up to 256 characters and is case sensitive. Duplicates are not allowed. This command is normally the name you have given to your Object Storage device. The *appliance_alias* is used to uniquely identify a single system.

-i *connection_endpoint*

Connection_endpoint of either the management GUI or IO Data Path of the Object Storage device.

The *connection_endpoint* given in the appliance configuration is used to access the appliance management GUI. For the appliance, the port number is given as a separate option.

The *connection_endpoint* given in the IO Path configuration is used to access the appliance namespace and its objects. For the iopath, the port number is a part of the connection endpoint if it is needed. *connection_endpoint* can either be DNS hostname e.g. host.abc.com or an Internet Protocol address e.g. 10.65.166.123, or an Internet Protocol address and port number e.g. 10.65.166.123:8080. If the port number is not given in the connection endpoint, port 80 is the default.

For the AWS media type, the *connection_endpoint* must specify an AWS region endpoint. If the endpoint is newly supported by Amazon, support within the Quantum storage system can be enabled by adding this endpoint along with its region name to the AWS regions configuration file (*/usr/cvfs/config/awsregions.json*) and then restarting the Quantum storage system.

Duplicate connection endpoints are not allowed.

-u PATH | **VHOST**

There are 2 ways to format a URL, PATH style and VHOST style. A PATH style URL looks like:

http://*ip-address***:***port***/***namespace-name***/***object-id*

A VHOST style URL looks like:

http://*namespace-name***.***ip-address***:***port***/***object-id*

Using a VHOST style URL requires additional address resolution setup on the calling host and target Object Storage system. For more information on VHOST URLs, please refer to the following link:
http://docs.aws.amazon.com/AmazonS3/latest/dev/VirtualHosting.html.

Please refer to the S3 setup instructions provided by the specific vendor.

The default is **PATH** style URL. **VHOST** style URLs are supported for S3 compatible media only and is not supported for AXR and Azure media.

-p *port_numbers*

This is the TCP/IP port number that you have configured for the management GUI. Specify the singular VIP(Virtual IP) for the management GUI.

-e http | **https**

Network protocol to use when accessing the management GUI or Client Daemons on Object Storage systems. Management GUI port can either be http or https. Each iopath references a Client Daemon. Each iopath can either be http or https. Verify with your Administrator the configured http/https port numbers.

Https is a layering of http on top of SSL/TLS. SNSM, via libcurl, uses an OpenSSL implementation of the SSL and TLS protocols.

Libcurl is configured at build time to set the correct default location for Certificate Authority (CA) Root certificates for OpenSSL use. This default varies depending on the OS distribution. You can override the default directory that holds the individual certificates with FS_OBJSTORAGE_CAPATH. You can also override the default certificate file bundle with FS_OBJSTORAGE_CACERT. Note that /usr/bin/c_rehash must be run on the overriding directory that holds the individual certificates to create the necessary symlinks to the individual certificate files for OpenSSL use. This action is not needed when overriding the certificate bundle file.

The FS_OBJSTORAGE_SSL_VERIFY_PEERHOST indicates the type of verification to be used: either peer or both peer and host verification. The default is peer and host verification. Note that wildcards in the hostname used in the Common Name (CN) in the certificate are incompatible with host verification. You must change the host verification to peer if wildcards are used.

If you overrode the default CA Root certificate(s) location set by libcurl, make sure that each DDM host is also re-configured accordingly.

-n *controller_alias*

This is normally the name you have given to your Object Storage controllers. Duplicates are not allowed. *controller_alias* is a variable string of up to 256 characters and is case sensitive.

-s *streams[:readstreams]*

streams specifies the maximum number of I/O streams per controller. This allows the number of concurrent I/O operations per controller to be adjusted. The maximum stream count must be greater than zero.

readstreams can be specified to reserve a subset of *streams* for file retrieves. The number of retrieves can exceed this count but the reserved streams cannot be used for non-retrieve requests. If the *readstreams* count is equal to the *streams* count, streams will not be made available for store requests.

If the **-s** option is not specified, the current default is used and streams are not reserved specifically for retrieves. Use the display option to see the default configured values.

-o *iopath_alias*

The I/O path alias. Duplicates are not allowed. *iopath_alias* is a string of up to 256 characters and is case sensitive.

-b *namespace*

This corresponds to the namespace that you have created on your Object Storage system. Duplicate namespaces within the same appliance are not allowed. *namespace* is a string of up to 256 characters and is case sensitive. The namespace is persistent and cannot be modified once created.

-v *provider*

This identifies the provider of the appliance. The appliance provider determines the format of provider-specific REST headers. The following provider types are supported by Tertiary Manager software:

AWS
AZURE
GOOGLE
LATTUS
S3COMPATIBLE

-t *mediatype*

The object storage media type. This is assigned to a namespace and an iopath. Namespaces and iopaths are associated with a specific Object Storage API. The following object storage media types are supported by Tertiary Manager software:

AWS
AXR
AZURE
GOOGLE
GOOGLES3
S3
S3COMPAT

If **-t** is not specified, the media type will default to **AXR**. Connectivity tests are only performed for **AXR** iopaths.

-S *signing_type*

Used to specify the signing type for the requests sent to the Object Storage and/or authentication server. The following signing types are supported:

V2
V4
AZURE

The default is **V4** for AWS and S3 compatible media, **V2** for GOOGLES3 and Lattus S3 media, and **AZURE** for Azure media. This option is not valid for AXR and Google media. When configuring V4, chunked mode is preferable and should be used if supported by the object storage system. For object storage which does not support chunked mode, such as AWS Snowball, you must specify the full payload mode signing option, **-J**, for the object storage bucket.

-U *username*

If permissions have been configured on the appliance system which require username/password authentication, the same credentials must be specified here. If no permissions have been configured on the appliance, they should not be configured here either. To remove the username/password combination during modification specify **none** for the username. For media types **AWS** (not using **CAP** authentication), **S3COMPAT**, **AZURE** and **GOOGLES3**, the username is required and must be defined in the bucket (namespace) configuration. The *username* can be a maximum of 256 characters and consist of any ASCII character. The username is also known as the Access

Key Id for S3 compatible media, or the Storage Account Name for Azure media.

-P *password*

If permissions have been configured on the appliance system which require username/password authentication, the same credentials must be specified here. If no permissions have been configured on the appliance, they should not be configured here either. For media types **AWS** (not using **CAP** authentication), **S3COMPAT**, **AZURE** and **GOOGLES3**, the password is required and must be defined in the bucket (namespace) configuration. The *password* can be a maximum of 256 characters and consist of any ASCII character. The password is also known as the Secret Access Key for S3 compatible media, or the Storage Access Key for Azure media.

-f *mediaid*

Media identifier is a name that uniquely identifies a specific namespace amongst multiple Object Storage systems. Media identifier is a variable string of 16 characters.

-I *uuid* Uuid uniquely identifies a specific namespace amongst multiple Object Storage systems. Uuid, if given must conform to RFC 4122 - A Universally Unique IDentifier (UUID) URN Namespace.

-N *object_prefix*

Specifies a prefix to be applied to all object IDs created on the media. *object_prefix* can be a maximum of 255 characters and is case sensitive. When modifying a namespace, the object prefix can be changed only if the media is blank. The object prefix can be disabled for blank media by specifying an empty string with this option.

-K *keyfile*

Specifies the location of the private key file containing the credentials for the service account which manages your media. The private key file must be created and downloaded from the Google Cloud service account associated with the specified media and must be in JSON format. This option is valid only for GOOGLE media.

-Y *authentication_type*

An authentication type is required for all Object Storage media except for AXR and GOOGLE. The following authentication types are supported:

STANDARD
STS_PUBLIC
STS_GOVCLOUD
CAP

If this option is not specified, the authentication type will be set to **STANDARD**.

The **STANDARD** type applies to all media and authenticates with a user name and password for Object Storage access. For S3 compatible media the user name and password are the Access Key Id and Secret Access Key. For Azure media, the user name and password are the Storage Account Name and Storage Access Key.

The **STS_PUBLIC** type uses the AWS Security Token Service (STS) to obtain temporary credentials for access to Object Storage in the AWS public cloud. This authentication type applies only to AWS media and requires a role, username and password be specified.

The **STS_GOVCLOUD** type uses the AWS Security Token Service (STS) to obtain temporary credentials for access to Object Storage in the AWS GovCloud. This authentication type applies only to AWS media and requires a role, username and password be specified.

The **CAP** type uses the AWS Commercial Cloud Services (C2S) Access Portal (CAP) to obtain temporary credentials for access to Object Storage in the AWS Private Cloud for the Federal Government. This authentication type applies only to AWS media and requires a role, CAP mission, and CAP agency be specified. Furthermore, it requires the specification of the following sysparms in */usr/adic/TSM/config/fs_sysparm_override*:

FS_OBJSTORAGE_CAPATH which sets the directory where the issuer's certificate authority (CA) can be found, if it is not already included in the operating system's default trusted root cer-

tificate file. Note, the certificate should be in PEM format. For example, the certificate can be copied to */usr/cvfs/config/ssl* and configured as follows:

FS_OBJSTORAGE_CAPATH=/usr/cvfs/config/ssl;

Note, if your customized CA PEM file contains more than one certificate, we recommend that you append the content of your customized CA PEM file to your operating system's default CA bundle and to **NOT** use sysparm FS_OBJSTORAGE_CAPATH to set the location of your customized CA PEM file. Otherwise, you could split your CA PEM file into multiple CA PEM files, each of which contains a single CA certificate, and use sysparm FS_OBJSTORAGE_CAPATH to set the location of your newly split single certificate CA PEM files.

FS_OBJSTORAGE_CLIENTCERT which sets the location of the X.509 client certificate installed on the system for the CAP server to authenticate. Note that the certificate should be in PEM format. For example, the client certificate can be copied to */usr/cvfs/config/ssl/client-cert-filepath*, and configured as follows:

FS_OBJSTORAGE_CLIENTCERT=/usr/cvfs/config/ssl/client-cert-filepath;

FS_OBJSTORAGE_C2S_CAP_HOSTPORT which sets the connection endpoint for the CAP server and can be configured as follows:

FS_OBJSTORAGE_C2S_CAP_HOSTPORT=cap-portal:port;

FS_OBJSTORAGE_CLIENTKEY which sets the location of the client private key if the client private key is kept separately from (i.e. not included in) the client certificate file. This parameter can be configured as follows:

FS_OBJSTORAGE_CLIENTKEY=/usr/cvfs/config/ssl/client-key-filename;

FS_OBJSTORAGE_CLIENTKEY_PASS which specifies the passphrase used to protect the client private key and can be configured as follows:

FS_OBJSTORAGE_CLIENTKEY_PASS=passphrase;

Make sure to run the following command to generate the hash for your certificates: */usr/bin/c_rehash /usr/cvfs/config/ssl*. Restart TSM to allow the system parameter changes to take effect.

-Z *authentication_endpoint*

Specifies an authentication endpoint which overrides the default endpoint for the public or GovCloud STS server. This option is valid only with the **STS_PUBLIC** and **STS_GOVCLOUD** authentication types. If the endpoint is newly supported by Amazon, support within the Quantum storage system can be enabled by adding this endpoint along with its region name to the AWS regions configuration file (*/usr/cvfs/config/awsregions.json*).

-O *storageclass*

Defines the storage class for AWS, S3COMPAT, AZURE and GOOGLES3 media only. The following storage classes are supported:

azure_block_blob — Azure Block Blob storage class. Blobs in this storage class are in either Cool or Hot tier. Cool or Hot tier is an attribute of the container, defined by the user during configuration of the container on Azure. Note: If the Azure lifecycle policies are configured to use the archive tier, then the **azure_block_blob_archive** storage class must be used instead of this one, otherwise files will not be retrievable from the archive tier.

azure_block_blob_archive — Data written to this tier must be rehydrated before it can be read. This storage class utilizes the Azure archive tier. It should be used when Azure lifecycle policies are configured to use the archive tier. Note: this tier feature only applies to Azure Blob storage and General Purpose v2 (GPv2) accounts. General Purpose v1 (GPv1) accounts do not support tiering.

For more information, check the Azure documentation on Blob storage tiers.

glacier S3 Glacier storage class using standard retrievals to access data from Glacier storage. When restoring from AWS S3 Glacier using standard retrievals, data can typically be accessed within 3-5 hours.

glacier_exp S3 Glacier storage class using expedited retrievals to access data from Glacier storage. When restoring from AWS S3 Glacier using expedited retrievals, data can typically be accessed within 5 minutes.

glacier_bulk S3 Glacier storage class using bulk retrievals to access data from Glacier storage. When restoring from AWS S3 Glacier using bulk retrievals, data can typically be accessed within 5-12 hours.

glacier_deep S3 Glacier storage class using standard retrievals to access data from Glacier Deep Archive storage. When restoring from AWS S3 Glacier Deep Archive using standard retrievals, data can typically be accessed within 12 hours.

glacier_deep_exp S3 Glacier storage class using the expedited retrieval option to access data in Glacier Deep Archive storage. This option is not supported by AWS.

glacier_deep_bulk S3 Glacier storage class using the bulk retrieval option to access data from Glacier Deep Archive storage. When restoring from AWS S3 Glacier Deep Archive using bulk retrievals, data can typically be accessed within 48 hours.

standard S3 Standard storage class

standard_ia S3 Infrequent Access storage class

By default, the storage class is set to **standard** for AWS, S3COMPAT, and GOOGLES3 media, and set to **azure_block_blob** for Azure media. **standard** is the only supported storage class for GOOGLES3.

Note, the cost of retrieval from an AWS Glacier storage class will vary by retrieval time, with the quickest retrieval type having the highest cost.

Please refer to provider documentation for detailed information on supported storage classes, including cost and retrieval times.

-R *role* For **STS_PUBLIC** and **STS_GOVCLOUD** authentication, use the Amazon Resource Name (ARN) of the role to assume. For **CAP** authentication, use the Identity and Access Management (IAM) role associated with the target C2S account. A role is required by AWS STS and the CAP servers to obtain temporary credentials. This option is valid only with the **STS_PUBLIC**, **STS_GOVCLOUD**, and **CAP** authentication types.

-D *duration*

The duration, in seconds, of the role session or OAuth credentials before expiration. The value must be in the range 900 to 3600. A default value of 3600 seconds is used if a duration is not specified. This option is valid only for the **STS_PUBLIC**, **STS_GOVCLOUD**, and **CAP** authentication types and for GOOGLE media.

-A *CAP_agency*

The agency associated with the target C2S account. This option is valid only for the CAP authentication type.

-M *CAP_mission*
The mission associated with the target C2S account. This option is valid only for the CAP authentication type.

-c *copy* Used to specify the copy number for the namespace or media being configured. If not specified when adding a namespace, the copy number will be set to 1.

-C *class*
Used to specify the policy class for the namespace or media being configured. If not specified when adding a namespace, no policy class association will be set for the media, and the media can be used by any and multiple policy classes. When modifying a namespace, the associated policy class can be changed as long as the media is blank. To remove the policy class association for a media so that it can be used by multiple policy classes, use the **fschmedstate**(1) command with the **-b** option.

-l As part of the report, show the configured username for all configured appliances and namespaces.

-G y | n
Specify whether the Object Storage server supports batch delete operation.

-J y | n Enable the AWS V4 full payload signing mode for store operation. If this option is not specified or 'n' is specified, the chunked transfer mode will be used. For performance reasons, full payload transfers with V4 signing should only be used for those storage systems, such as AWS Snowball, which do not support chunked mode V4 signing.

-B During addition and modification, the path to the component is verified for connectivity. If connectivity cannot be made, a warning message will be given and the command will fail. By specifying the **-B** option, the connectivity verification will still occur, but the command will not fail if the connectivity cannot be verified.

-X For conversion from AXR to S3, if the media has active objects, this option is required to force the change of the media name and/or type.

-V *awsregionsfile*
Validates that the specified AWS region configuration file is correctly formatted.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

RESTRICTIONS

The host address, port numbers, and namespace must be accessible when performing add, modify, or delete operations. The command automatically performs connectivity tests. If using https, the host address must match one of the names in the connecting server's SSL Certificate during

SSL/TLS authentication.

EXAMPLES

Configure a Webscale Object Storage destination, followed by a namespace corresponding to bucket web-bucket, specifying media type S3COMPAT, V4 signing, and standard authentication :

```
fsobjcfg -a -i webscale-server-hostname -v S3COMPATIBLE -p 443 \
     -e https webscale
fsobjcfg -a -n websctl -s 48 webscale
fsobjcfg -a -o webspath -i webscale-server-hostname:8082 -e https \
     -u PATH -t S3COMPAT -n websctl
fsobjcfg -a -b web-bucket -t S3COMPAT -c 1 -f WS0001 -U username \
     -P password -S V4 -Y STANDARD webscale
```

Configure an Azure Object Storage destination, followed by a namespace corresponding to Azure container az-container, specifying media type AZURE, Azure signing, standard authentication, and the Azure Block Blob storage class. The default is Azure Block Blob:

```
fsobjcfg -a -i blob.core.windows.net -v AZURE -p 443 -e https azure
fsobjcfg -a -n azurectl -s 48 azure
fsobjcfg -a -o azurepath -i blob.core.windows.net -e https -u PATH \
     -t AZURE -n azurectl
fsobjcfg -a -b az-container -t AZURE -c 1 -f AZ0001 -U username \
     -P password -O azure_block_blob -S AZURE -Y STANDARD azure
fsobjcfg -a -b az-container -t AZURE -c 2 -f AZ0002 -U username \
     -P password -O azure_block_blob_archive -S AZURE -Y STANDARD azure
```

Configure an AWS Object Storage destination, followed by a namespace corresponding to AWS bucket, bucket-1, requesting V4 signing, standard authentication, and the standard storage class :

```
fsobjcfg -a -i s3.amazonaws.com -v AWS -p 443 -e https aws
fsobjcfg -a -n awsctl -s 48 aws
fsobjcfg -a -o awspath -i s3-us-west-2.amazonaws.com -e https \
      -u VHOST -t AWS -n awsctl
fsobjcfg -a -b bucket-1 -t AWS -c 1 -f AWS0001 -U username \
      -P password -O standard -S V4 -Y STANDARD aws
```

Configure an AWS GovCloud Object Storage destination, followed by a namespace corresponding to AWS bucket govbucket-1, requesting V4 signing, AWS STS GovCloud authentication, and the standard storage class :

```
fsobjcfg -a -i s3-us-gov-west-1.amazonaws.com -v AWS -p 443 \
     -e https awsgov
fsobjcfg -a -n awsctl -s 48 awsgov
fsobjcfg -a -o awspath -i s3-us-gov-west-1.amazonaws.com -e https \
     -u VHOST -t AWS -n awsctl
fsobjcfg -a -b govbucket-1 -t AWS -c 1 -f AWS0000 -U username \
     -P password -O standard -S V4 -Y STS_GOVCLOUD -D 3600 \
     -R arn:aws-us-gov:iam::123456789012:role/gov-role awsgov
```

Configure a Commercial Cloud Services Object Storage destination, followed by a namespace corresponding to AWS bucket c2sbucket-1, requesting V4 signing, CAP authentication and the AWS Glacier storage class :

```
showsysparm FS_OBJSTORAGE_C2S_CAP_HOSTPORT
FS_OBJSTORAGE_C2S_CAP_HOSTPORT=ec2-12-34-567-89.us-west-2.compute.amazonaws.com:4
```

```
showsysparm FS_OBJSTORAGE_CLIENTCERT
FS_OBJSTORAGE_CLIENTCERT=/root/ec2-client.pem

fsobjcfg -a -i s3.amazonaws.com -v AWS -p 443 -e https awsc2s
fsobjcfg -a -n awsctl -s 48 awsc2s
fsobjcfg -a -o awspath -i c2sbucket-1_region_endpoint -e https \
       -u VHOST -t AWS -n awsctl
fsobjcfg -a -b c2sbucket-1 -t AWS -c 1 -f C2S0001 -O glacier -S V4 \
       -Y CAP -R myrole -D 3600 -M mission -A agency awsc2s
```

Configure a GOOGLES3 Object Storage destination, followed by a namespace corresponding to GOOGLE bucket googbucket-1 :

```
fsobjcfg -a -i storage.googleapis.com -v GOOGLE -p 443 \
       -e https google
fsobjcfg -a -n googlectl google
fsobjcfg -a -o googlepath -i storage.googleapis.com -e https \
       -u VHOST -t GOOGLES3 -n googlectl
fsobjcfg -a -b googbucket-1 -t GOOGLES3 -c 1 \
       -U username -P password google
```

Configure a GOOGLE Object Storage destination, followed by a namespace corresponding to GOOGLE bucket google-0 :

```
fsobjcfg -a -i storage.googleapis.com -v GOOGLE -p 443 \
       -e https google
fsobjcfg -a -n googlectl google
fsobjcfg -a -o googlepath -i storage.googleapis.com -e https \
       -u VHOST -t GOOGLE -n googlectl
fsobjcfg -a -b google-0 -t GOOGLE -f GGL0000 -c 1 -D 3600 \
       -K /keys/StorNext-1d5684e5cf83.json google
```

FILES

/usr/cvfs/config/awsregions.json
/usr/adic/TSM/config/fs_sysparm.README
/usr/adic/TSM/config/fs_sysparm

SEE ALSO

fschstate(1), **fschmedstate**(1), **fsrminfo**(1), **fsazure**(1)

NAME

fsobjimport − import object storage media and ingest content into the Tertiary Manager system

SYNOPSIS

fsobjimport [**-c** *class*] [**-d** *directory*] [**-D** *delimiter*] [**-f** *manifest-file*] [**-g** *group*] **-m** *mediaid* [**-M** *mode*] [**-p** *obj-prefix*] [**-P** *path-prefix*] [**-Q**] [**-t** *files*|*list*|*media*] [**-T** *time*] [**-u** *user*] [**-U**] [**-V** *numvers*] [**-W**]

fsobjimport -C [**-m** *mediaid*]

fsobjimport -R -m *mediaid*

fsobjimport -s [**-F** *type*] [**-l**] [**-m** *mediaid*]

fsobjimport -S -i *requestid*|**-m** *mediaid*

fsobjimport -h

DESCRIPTION

fsobjimport(1) is used to import object storage media and to ingest content into the Tertiary Manager system. It also provides administrative operations for scanning and listing the contents of object storage media and for reporting the status of import requests.

The media must be known to the Tertiary Manager system before the import can be requested and must be in the **protect** state in order for it to be used for import.

For file imports, the contents of the files found on the media will be copied into a destination directory. If the file already exists in the destination directory, a numeric suffix will be appended to the name of the copied file, increasing from zero, until a file with a unique name can be created. Otherwise, option -W can be specified to request that the file be overwritten if it already exists. Once the import has completed, no references to the media are retained and the media can be removed from the Tertiary Manager system unless needed for other purposes.

For media imports, media is retained in the Tertiary Manager system, and the metadata required to map the contents of an object storage bucket into a StorNext file system will be created for each file found on the media. The created metadata allows the data to be retrieved through the Tertiary Manager system. Once the import is complete, the media state must be changed from the **protect** state to allow access to the files.

Specifying the list import type will generate a list of the files on an object storage medium. The file list will be created in JSON format. The contents of the file can be modified and then used for file or media import.

fsobjimport(1) will log file error information for the current operation to */usr/adic/TSM/logs/reports/import.mediaid*. The status information for an import operation can be displayed by specifying the **-s** option of the **fsobjimport**(1) command.

If an import request terminates before successful completion, it can be resumed by specifying the **-R** option with the **fsobjimport**(1) command. In this case, the **-m** option must also be specified to identify the failed import request. The import request will be resumed from the failure point.

If an import request has been terminated and is no longer needed, the **-U** option can be used to undo the import. If neither an undo nor resume of a terminated import is required, the checkpoint metadata for this media should be cleared with the **-C** option of the **fsobjimport**(1) command. The **-m** option must also be specified to identify the import request to clear.

An import request can be tuned by modifying system parameters which define the number of threads used on the request. System parameters IMP_META_NUM_THREADS and IMP_DBASE_NUM_THREADS can be modified to optimize the media import processing for your system. IMP_META_NUM_THREADS defines the number of threads used to create the file metadata and IMP_DBASE_NUM_THREADS the

number of threads used to create the database metadata. Parameter IMP_COPY_NUM_THREADS defines the number of threads used to copy data on file imports and IMP_REMOVE_NUM_THREADS defines the number of threads used to remove the imported files or metadata for an undo request. For both media and file imports, IMP_OBJMETA_NUM_THREADS defines the number of threads used to retrieve metadata for the imported objects. By default, fifteen threads are used for the object metadata threads and five threads are used for each of the other thread types. For further descriptions of each system parameter, see the *fs_sysparm.README* file.

OPTIONS

-c *class* For media imports, associate the destination directory with the specified policy class.

-C Clear the import request information for the specified medium or for all terminated import requests. After using this option, the import request can no longer be resumed.

-d *directory*

Ingest data into the specified destination directory *directory*. The destination directory *must* be associated with a relation point when ingesting media. The destination directory *may* be associated with a relation point if ingesting files. Note that if file ingest is chosen and the destination directory is not under a relation point, the files will be ingested but will not be managed by the Tertiary Manager system.

If a path prefix is specified with option **-P** and is absolute, the prefix will be stripped from the metadata path before the file is created in the destination directory.

If option **-P** is not specified and path metadata exists for the object, each file will be created using the path specified in the metadata.

-D *delimiter*

Specifies the character representing a directory delimiter in the object storage names.

-f *manifest-file*

If specified with **-t** *list*, output a list of the files on an object storage medium to this file. The file list will be output in JSON format and will include file attributes. If specified with **-t** *media* or *files*, ingest the files listed in file *manifest-file*. The file attribute values can be modified from their defaults before using the file list for ingest.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

-g *group*
Set ingested files to group *group* where *group* is the group name or numeric group ID. Defaults to 0.

-h Print a usage message and exit.

-i Specify the request ID of the import request to suspend. The **fsobjimport**(1) command, specified with options **-s** and **-l**, can be used to display the request IDs of pending imports.

-l Long status report. For the specified import request, this option provides all of the import option values, the state of the import, import identifiers, and the progress of the import.

-m *mediaid*
Operate on the medium *mediaid*. The medium must be known to the Tertiary Manager system.

-M *mode*
Set the permission bits for ingested files to *mode*. If this option is not specified, for file imports, the mode will default to 0440, and for media imports, the mode will be inherited from the destination directory, with the execute bits set to zero.

-p *obj-prefix*
Import only those files with object IDs beginning with the specified prefix.

-P *path-prefix*
Import only those files having object metadata paths beginning with the specified prefix. If the path prefix is absolute and this option is specified with the -d option, the prefix will be stripped from the metadata path before the file is created in the destination directory.

-Q Query each object found on the media for user-defined metadata. If supported object metadata is detected, it will be used on import. Supported metadata includes Storage Manager metadata and the AWS POSIX metadata.

-R Restart the media or file import processing from the failure point using the checkpoint metadata. The request is identified by the media ID specified with the **-m** option. Restarts of failed list requests are not supported.

-s Display the status of the import request identified by the media ID specified with the **-m** option, or, in the case where the **-m** option is omitted, display all pending or failed import requests.

-S Suspend an import request. Either the **-m** option or the **-i** option must be used to identify the import request to suspend. A suspended import can be restarted by specifying the **-R** option with the media ID. If an import is no longer needed, the **-U** option can be specified to undo the partially completed import. If neither an import resume nor undo is needed, the checkpoint metadata should be cleared from the Storage Manager by specifying the **-C** option with the media ID of the suspended request.

-t *type* Identify the import request type. Supported types are:

files Copy contents of medium to destination directory

list Generate a list of files residing on medium

media Create metadata for each file residing on medium, which will allow file content to be retrieved

If the *type* is **files** or **media**, then the **-d** option must also be specified. If the *type* is **list**, an output file must be specified with option **-f**.

-T *time* Import only those files created after the specified time. Times are expressed in local time and must be specified in format **YYYY-MM-DD HH:MM:SS**.

-u *user* Set ingested files to owner *user* where *user* is the user name or numeric user ID. Defaults to 0.

-U Undo a media or file import request. An undo of a media import will delete the metadata and database information associated with the files found on the imported media. An undo of a file import will remove all files copied into the destination directory during the import. The options specified with an undo must match those used on the import to return to the pre-import state. If this is an undo of a terminated import, the checkpoint metadata will be used to undo the request.

-V *numvers*
On file imports, the maximum number of file versions to import. If this option is not specified, all versions of a file are imported.

-W On file imports, overwrite a file if it already exists.

EXIT VALUES

Exit codes for the **fsobjimport**(1) command are:

EEXIST
Import file or manifest file already exists.

EFAULT
Bad address specified in file system request.

EINTR The import request was interrupted or suspended.

EINVAL
Invalid command arguments, manifest file, or object storage data.

EIO An I/O error occurred while transferring object storage data or when accessing the manifest file or database.

ENAMETOOLONG
Object ID or path length exceeds the system maximum.

ENODIR
Destination path is not a directory.

ENOENT
The specified media has not been configured, a path to the media has not been configured, the specified import request is not pending, or the destination directory does not exist.

ENOMEM
Insufficient memory to process command.

ENOSPC
No space on file system.

ENOSYS
: Object storage import not supported by provider.

ENOTCONN
: Cannot connect to object storage system or database.

EPERM
: The media is write-protected, an import request is already active for the media, or you do not have permission to access the object storage system or to access a file.

The **fsobjimport**(1) command may also fail and return *errno* for the errors documented in **fopen**(3)

EXAMPLES

Example 1

Import files from medium AWS0000, copying the files to directory /stornext/snfs1/import1. Create the files with an owner ID of 101, a group ID of 120, and with file permissions of 0660. Pre-existing files in the destination directory which have the same name as an imported file will be overwritten:

```
fsobjimport -m AWS0000 -t files -d /stornext/snfs1/import1 -u 101 \
            -g 120 -M 0660 -W
```

Example 2

Undo the import request from Example 1, removing all imported files from directory /stornext/snfs1/import1:

```
fsobjimport -m AWS0000 -t files -d /stornext/snfs1/import1 -u 101 \
            -g 120 -M 0660 -W -U
```

Example 3

Import medium STS0001 into the Tertiary Manager system, creating the metadata which allows access to the file contents. Import only those files which were created after 7:30 P.M., August 10, 2017:

```
fsobjimport -m STS0001 -t media -d /stornext/snfs1/import2 \
            -T "2017-08-10 19:30:00"
```

Example 4

Generate a list of files found on medium AWS0000, which are prefixed with *accounts*, but do not ingest the contents. Replace the object storage delimiter ':' with a forward slash ('/') in the destination file path:

```
fsobjimport -m AWS0000 -t list -f manifest-file -p accounts -D :
```

The file created with the list option can then be modified and used to create a manifest file that is used for ingesting content:

```
fsobjimport -m AWS0000 -t files -d /stornext/snfs1/import3 \
            -f manifest-file
```

Example 5

Check the status of all running **fsobjimport**(1) processes:

```
fsobjimport -s
```

Example 6

Suspend the import of medium AWS0000:

```
fsobjimport -m AWS0000 -S
```

Example 7

Restart a suspended import of medium AWS0000:

```
fsobjimport -m AWS0000 -R
```

Example 8

Undo a suspended import of medium AWS0000:

```
fsobjimport -m AWS0000 -U
```

FILES

/usr/adic/TSM/logs/reports/import.<mediaid>
objimport.json

SEE ALSO

fsaddclass(1) **fschmedstate**(1) **fsobjcfg**(1), **fsobjmeta**(1) **objimport.json**(5)

NAME

fsobjlist − Generate a list of object ID to path mappings.

SYNOPSIS

fsobjlist [**-a**] [**-C** *class*] [**-m** *mediaid* | **-n** *namespace*] [**-F** *type*]

fsobjlist -h

DESCRIPTION

fsobjlist(1) is used to generate a report of object ID to path mappings for the specified media, namespace or class. Each entry represents a file segment.

Specifying fsobjlist(1) without options will display object ID to path mappings for all configured object storage media.

REPORT STATUS

The file segment entries listed by the **fsobjlist**(1) command are as follows:

Object ID	The object id of the segment archived to object storage media.
Path	The name of the file at store time.
Segment	The number of the segment stored for a file.
Total Segments	The total number of segments stored for a file.
Version	The version number of the file.

OPTIONS

-a Generate a list of active object IDs. By default, all objects (active and inactive) are listed.

-C *class*
Generate a list of object IDs for objects associated with the specified class.

-m *mediaid*
Generate a list of Object IDs for the specified media ID.

-n *namespace*
Generate a list of Object IDs for the specified namespace.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

EXIT VALUES

Exit codes for the **fsobjlist**(1) command are:

EBUSY
Database too busy to process request.

EINVAL
Invalid command arguments.

EIO An I/O error occurred when accessing the database.

ENOENT
: The specified media, namespace or class has not been configured.

ENOMEM
: Insufficient memory to process command.

ENOTCONN
: Cannot connect to database.

SEE ALSO

fsobjcfg(1) **fsclassinfo**(1)

NAME

fsobjlocate − Locates files based on object IDs.

SYNOPSIS

fsobjlocate [**-F** *type*] *objectid ...*

fsobjlocate -h

DESCRIPTION

The **fsobjlocate**(1) command displays the file path for each object ID specified.

OPTIONS

objectid...

Report the pathname of each object ID listed. At least one object ID is required.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

EXIT VALUES

Exit codes for the **fsobjlocate**(1) command are:

EINVAL
Invalid command arguments.

EIO An I/O error occurred when accessing the database.

ENOENT
An object ID or path was not found.

ENOMEM
Insufficient memory to process command.

SEE ALSO

fsfileinfo(1), **fsmedinfo**(1)

NAME

fsobjmeta − Set metadata on objects in Object Storage.

SYNOPSIS

fsobjmeta [**-a**] [**-c** *copynum*] [**-C** *class*] [**-m** *mediaid* | **-n** *namespace*]

fsobjmeta [**-a**] [**-c** *copynum*] [**-B** *batchfile*]

fsobjmeta [**-a**] [**-c** *copynum*] *filename ...*

fsobjmeta -h

DESCRIPTION

fsobjmeta(1) is used to set metadata on objects in Object Storage. The objects affected may be specified either through options or by providing a file list.

In order to set metadata on objects in Azure Archive Storage, the objects must first be rehydrated. Similarly, objects in S3 Glacier Storage must be restored before metadata can be set on them.

Specifying fsobjmeta(1) without options will set metadata on all objects in Object Storage.

OPTIONS

filename...
: For each file listed, metadata will be set on the archive copy in object storage for that file.

-a
: Set metadata only on active object IDs. By default, metadata is set on all objects (active and inactive).

-B *batchfile*
: Specify a batch file containing a list of files for which metadata will be set on the archive copy in object storage for each file.

-c *copynum*
: Set metadata on a specific copy for objects associated with a specified class or file.

-C *class*
: Set metadata on objects associated with the specified class.

-m *mediaid*
: Set metadata on objects archived to the specified media.

-n *namespace*
: Set metadata on objects archived to the specified object storage namespace.

EXIT VALUES

Exit codes for the **fsobjmeta**(1) command are:

EBUSY
: Database or object storage system busy.

EINVAL
: Invalid command arguments.

EIO
: An I/O error occurred while setting object storage metadata or when accessing the database.

ENAMETOOLONG
: Path length exceeds the system maximum.

ENODEV
: Cannot find device ID for specified file.

ENOENT
: The specified media, namespace or class has not been configured or the specified file does not exist.

ENOMEM
: Insufficient memory to process command.

ENOSYS
: Object storage system does not support setting of metadata.

ENOTCONN
: Cannot connect to object storage system or database.

EPERM
: Error authenticating object storage request.

EREMOTE
: Archive copy of file does not exist.

The **fsobjmeta**(1) command may also fail and return *errno* for the errors documented in **fopen**(3)

FILES

/usr/adic/TSM/logs/reports/objmeta.<requestid>

SEE ALSO

fsobjcfg(1) **fsclassinfo**(1) **fsobjlist**(1)

NAME

fsobjratelimit − Set the object storage upload and/or download rate limit

SYNOPSIS

fsobjratelimit [**-r** *receive-rate*] [**-s** *send-rate*] [**-F** *type*]

fsobjratelimit -h

DESCRIPTION

fsobjratelimit(1) is used to rate limit data transferred to or from object storage systems, to control the maximum bandwidth used by the Storage Manager. The values specified with this command will override parameter settings FS_OBJSTORAGE_SEND_RATELIMIT and FS_OBJSTORAGE_RECV_RATELIMIT. Note that the limits are applied globally; that is, a rate limit set using this command will limit the total bandwidth used across all Storage Manager hosts.

Storage Manager uses libcurl to maintain the configured rates. libcurl will ensure a rate lower than the requested value but cannot guarantee that the configured aggregate rate will be achieved across all connections.

Specifying fsobjratelimit(1) without options will display the current rate limit settings, the number of object storage streams allocated, and the rate per stream for both uploads and downloads.

OPTIONS

-r *receive-rate*

Limit the total object storage download rate, specified in bytes per second, to *receive-rate*. The cumulative rate of all Storage Manager streams used to download data from an object storage system will not exceed this value. The receive rate may include one of the following suffixes:

b	bytes
k	kibibytes (1024)
M	mebibytes (1024^2)
G	gibibytes (1024^3)

-s *send-rate*

Limit the total object storage upload rate, specified in bytes per second, to *send-rate*. The cumulative rate of all Storage Manager streams used to upload data to object storage systems will not exceed this value. The send rate may include one of the following suffixes:

b	bytes
k	kibibytes (1024)
M	mebibytes (1024^2)
G	gibibytes (1024^3)

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

SEE ALSO

fsobjcfg(1) **fsschedule**(1) **fsschedlock**(1)

NAME

fsobjrestore − Restores and reports the state of an object in Object Storage.

SYNOPSIS

fsobjrestore -h

fsobjrestore [**-c** *copynum*] [**OPTIONS**] *filename ...*

fsobjrestore [**-c** *copynum*] **-B** *batchfilename* | **-R** *directory* | **-D** *directory* [**OPTIONS**]

fsobjrestore [**-a** *serverauth*] -n *namespace* -o *objectname* -u **URL** [**-U** *username* **-P** *password*] [**-C** *certfile* | **-p** *certpath*] [**OPTIONS**]

OPTIONS

[**-F** *type*] [**-O** *restoretier*] [**-r** | **-s** | -w] [**-T** *restoretype*] [**-y**]

DESCRIPTION

This command will report the state of an object or restore an object asynchronously or synchronously from S3 Glacier Object Storage or from the Azure Archive tier. If the stored data is known to the Tertiary Manager system, file names can be listed on the command line or in a file or requested by directory. If the stored data is not known to the Tertiary Manager system, the object storage components and security information must be provided. These attributes provide the addressing information and credentials required to access the object.

If a recursive restore is requested using the **-R** option, all files that reside in an archive tier in the directory specified and all files that reside in an archive tier in all subdirectories will be restored. The **-D** option will process files in the directory without recursion.

If a batch restore is requested using the **-B** option, all files that reside in an archive tier, listed in file *batchfilename*, will be restored.

When restoring from Azure Archive, by default the tier state will be changed from the Archive tier to the Hot tier. Option **-O cool** can be specified to restore to the Cool tier.

Note, Azure object restores only apply to Azure Blob storage and General Purpose v2 (GPv2) accounts. General Purpose v1 (GPv1) accounts do not support tiering.

OPTIONS

filename ...

The full path name(s) of the file(s) to restore. Each file path must specify a file in a migration directory.

-a *serverAuth*

The server authorization setting. Default is 2. The valid options are:

0 none

1 verify peer only

2 verify peer and host

-B *batchfilename*

The batch file contains a list of files to be restored. File names should be listed one per line. Each line in the file is read as a string of characters, without interpretation, up to the line terminator. You should not enclose the file name in quotes, use escape characters or any character not in the actual file name.

-c *copynum*

Restore or report the object status of a specific copy of a file.

-C *certfile*

File containing the Certificate Authority (CA) certificate(s).

-D *directory*
Restore files in the specified directory. This is not recursive.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

-n *namespace*
Name of the bucket containing the object to restore.

-o *objectname*
Object ID of the object to restore.

-O *restoretier*
Specifies the restore tier when rehydrating an object from Azure Archive storage. The following tiers are supported:

hot Rehydrate the Azure Archive object to the *Hot* access tier.

cool Rehydrate the Azure Archive object to the *Cool* access tier.

-p *certpath*
Directory path that contains the individual CA certificate files.

-r Restore the object from S3 Glacier Object Storage or from the Azure Archive tier. Do not wait for completion.

-R *directory*
The directory from which to start the recursive restore. All files from the specified directory and any subdirectories will be restored. Depending upon the number of files in the directory and subdirectories, running this option may incur a large cost.

-s Report the state of the object in Object Storage.

-T *restoretype*
Specifies the type of retrieval from S3 Glacier, S3 Glacier Deep Archive or Azure Archive storage. For retrievals from Amazon Web Services (AWS) S3 Glacier storage, the types vary by cost and retrieval time, with the quickest retrieval type having the highest cost. Please refer to provider documentation for detailed information on retrieval times and cost. The following types are supported:

standard S3 Glacier or Azure Archive standard retrieval. Allows for data access within 3-5 hours from AWS S3 Glacier, within 12 hours from AWS Glacier Deep Archive storage, or within 15 hours from Azure Archive storage.

expedited S3 Glacier expedited retrieval allowing data access within 1-5 minutes. This option is not supported for data in AWS S3 Glacier Deep Archive storage or for data in Azure Archive storage.

bulk S3 Glacier bulk retrieval. Allows for data access within 5-12 hours from AWS S3 Glacier storage or within 48 hours for data in AWS S3 Glacier Deep Archive storage. This option is not supported for data in Azure Archive storage.

This option is ignored if unsupported by the media or if the file is not in S3 Glacier, S3 Glacier Deep Archive, or Azure Archive storage.

-u *URL* URL that includes the bucket name of the media.

-U *username* **-P** *password*
Username and password to access the URL.

-w Restore the object from S3 Glacier Object Storage or from the Azure Archive tier. Wait for completion.

-y Assume a *yes* response for the restore and do not prompt the user.

EXIT STATUS

Exit codes for the **fsobjrestore**(1) command are:

0 Command completed successfully.

1 Command failed.

2 Command syntax error.

NOTE

Objects are restored to temporary cache space on the Object Storage system and not into Tertiary Manager primary disk space.

EXAMPLES

Example 1

Restore the object storage archive copy for all files in directory /stornext/snfs1/movies and in all subdirectories of /stornext/snfs1/movies. Wait for the restore to complete.

```
fsobjrestore -R /stornext/snfs1/movies
```

Example 2

Report the state of all archive copies for the files listed in batch file /tmp/batchlist.

```
fsobjrestore -s -B /tmp/batchlist
```

Example 3

Restore an object not known to the Tertiary Manager system from Glacier storage. Do not wait for the response.

```
fsobjrestore -u https://bucket.s3-us-west-1.amazonaws.com \
      -o objectid -n bucket -U username \
      -P password -r
```

SEE ALSO

fsazure(1), fsobjcfg(1), fsretrieve(1)

NAME

fspolicy − Command used for managing disk data and disk space on a policy class or file system basis.

SYNOPSIS

fspolicy -b -y *filesystemmountpoint* [**-A**] [**-C**[**-S**]]

fspolicy -s -y *filesystemmountpoint* [**-e**] [**-m** *minstoretime*] [**-v** *drivepool*]

fspolicy -r -y *filesystemmountpoint* [**-e**] [**-m** *minreloctime*] [**-o** *goal*] [**-z** *minsize*] **-a** *affinity*

fspolicy -t -y *filesystemmountpoint* [**-e**] [**-m** *mintrunctime*] [**-o** *goal*] [**-z** *minsize*] [**-a** *affinity*]

fspolicy -s -c *classname* [**-m** *minstoretime*] [**-v** *drivepool*]

fspolicy -r -c *classname* [**-m** *minreloctime*] [**-z** *minsize*]

fspolicy -t -c *classname* [**-m** *mintrunctime*] [**-o** *goal*] [**-z** *minsize*]

DESCRIPTION

The **fspolicy**(1) command manages files on file systems controlled by the Tertiary Manager software. It applies the storage, relocation, truncation, and copy expiration rules defined in each policy class to the files in the file system. Depending on which type of policy is invoked, the **filesystems**(4) configuration parameters may also be applied.

The Tertiary Manager tracks file system activity and maintains lists of files that are candidates to be stored, relocated, truncated, or expired. When the **fspolicy**(1) command is invoked, these candidate lists are accessed and the files that meet the policy class criteria and/or the criteria specified with the command will be stored, relocated, or truncated as needed.

A set of Tertiary Manager daemons automatically kick off **fspolicy**(1) commands as needed for storing, relocating and truncating files. It is generally not necessary for a user/administrator to run the **fspolicy**(1) command.

Store Policy

The **-s** option invokes a store policy, which stores files to storage media. Files that have remained on disk longer than *minstoretime* and have not been excluded from stores via **fschfiat**(1) or **exclusions**(4) are considered candidates to be stored. Any additional copies of files that were not stored since the last run of a store policy are also stored at this time. If a file is modified after being stored, all existing copies on media will become invalid, and the file will again become a candidate for storage after the *minstoretime* has elapsed. The storage policy uses the drive pool and media type configured in the policy class. The drive pool and media must be available, or no files will be stored. The **-s** option can be specified with either a policy class or with a file system. A file system should only be specified in emergency situations (see the **Emergency Policies** section below).

The Tertiary Manager **fs_store** daemon constantly monitors store candidates and launches store policies as needed using the **fspolicy -s -c** command signature. Additionally, the **fs_eventd** daemon constantly monitors file system used space and may automatically launch an emergency store policy using the **fspolicy -s -y -e** command signature when it determines that a particular file system has reached its configured High Use threshold for used space. These daemons effectively automate the application of store policies in the Tertiary Manager system. Under normal circumstances, a user/administrator should not have to run the **fspolicy**(1) command to invoke a store policy.

Relocation Policy

The **-r** option invokes a relocation policy, which relocates files on disk from one disk affinity to another. Files that have not been accessed in *minreloctime* and have not been excluded from relocation via **fschfiat**(1) are considered candidates for relocation. The **-r** option may be specified with either a policy class or with a file system affinity.

If a policy class is specified, all relocation candidates will be relocated from their source affinity to their target affinity, as dictated by the policy class configuration. This type of policy is automatically launched by the Tertiary Manager **fs_tierman** daemon once daily after midnight using the **fspolicy -r -c** command signature.

Alternatively, if a particular file system and affinity are specified, relocation candidates will be relocated to their target affinity only until enough space has been freed on the source affinity to reach a particular used space goal. The affinity's used space goal is dependent on the **filesystems**(4) configuration parameters and whether the **-o** option was specified with the command. This type of policy is automatically launched by the Tertiary Manager **fs_eventd** daemon using the **fspolicy -r -y -a** command signature when it determines that a particular file system affinity has reached its configured High Use threshold for used space.

As stated above, the Tertiary Manager **fs_tierman** and **fs_eventd** daemons are responsible for launching relocation policies as needed to relocate files from one disk affinity to another. These daemons effectively automate the application of relocation policies in the Tertiary Manager system. Under normal circumstances, a user/administrator should not have to run the **fspolicy**(1) command to invoke a relocation policy.

Truncation Policy

The **-t** option invokes a truncation policy, which truncates files on disk. Files that have all required copies stored on storage media, have not been accessed in *mintrunctime*, and are not excluded from truncation via **fschfiat**(1) or **exclusions**(4) are considered candidates for truncation. The **-t** option can be specified with either a policy class or with a file system. Truncation candidates will be truncated only until enough space has been freed on the file system to reach a particular used space goal. The file system's used space goal is dependent on the **filesystems**(4) configuration parameters and whether the **-o** or **-m** options were specified with the command.

The Tertiary Manager **fs_tierman** and **fs_eventd** daemons are responsible for launching truncation policies as needed. The **fs_tierman** daemon automatically launches a truncation policy once daily after midnight using the **fspolicy -t -c** command signature. The **fs_eventd** daemon constantly monitors file system used space and automatically launches a truncation policy using the **fspolicy -t -y** command signature when it determines that a particular file system has reached its configured High Use threshold for used space. These daemons effectively automate the application of truncation policies in the Tertiary Manager system. Under normal circumstances, a user/administrator should not have to run the **fspolicy**(1) command to invoke a truncation policy.

Rebuild Policy

The **-b** option invokes the candidate rebuild functionality. This repopulates/rebuilds candidate lists (store, alternate-store-location, relocation, expiration, and truncation) per file system. This by default is done by searching the file system mdarchive contents. The purpose of this functionality is to verify candidate list integrity. This policy is automatically run on a regular basis by one of the Tertiary Manager daemons (see **fsschedule**(1)). It can also be run by hand if desired.

When searching the mdarchive, the rebuild policy by default does not perform an exhaustive search of the file system contents but queries time frames of interest based on last known results and class candidate times. If the candidate lists have become completely lost/corrupted and need to be rebuilt from the ground up, the default functionality will be insufficient. See the **-C** and **-S** options for performing a complete rebuild operation.

Emergency Policies

The **-e** option changes the priority of the command. For store policies, this means that instead of the store command being placed at the end of the queue waiting for resources, the command is given a priority value of 1 so it will be placed at the head of the queue. For relocation policies, this means the *minreloctime* for a file will be ignored, and any file that can be relocated will be a valid candidate, regardless of how recently it has been accessed. For truncation policies, this means the *mintrunctime* for a file will be ignored, and any file that can be truncated (all copies stored) will be a valid candidate, regardless of how recently it has been accessed.

Min Size

The **-z** option allows file size to be considered when determining which files are to be relocated or truncated. Files larger than or equal to the specified *minsize* will be candidates for relocation or truncation. Files less than the minsize will not be relocated or truncated during the command execution.

Used Space Goal Override

The **-o** option can be specified to override the file system or affinity used space goal specified in the **filesystems**(4) configuration parameters. For relocation policies that use the **-y** option and for all truncation policies, the used space goal specifies the used space of the affinity or file system at which point the policy will stop relocating or truncating files.

Min Time Override

The **-m** option is used to override the *minstoretime*, *minreloctime*, and *mintrunctime* values configured in the policy class.

When used with a store policy, this option overrides the *minstoretime*, the amount of time that a file must reside on disk before becoming a store candidate. The store policy will not store a file to storage media unless it has resided on disk for this specified time.

When used with a relocation policy, this option overrides the *minreloctime*, the amount of time that a file must remain unaccessed on disk before becoming a relocation candidate. The relocation policy will not relocate a file's disk blocks unless it has remained unaccessed on disk for this specified time.

When used with at truncation policy, this option overrides the *mintrunctime*, the amount of time that a stored file must remain unaccessed on disk before becoming a truncation candidate. The truncation policy will not truncate a file's disk blocks unless it has remained unaccessed on disk for this specified time.

Drive Pools

The **-v** option will store files using only drives associated with the specified *drivepool*. This allows the number of drives used for store policies to be throttled by only allocating a subset of the drives to the specified drivepool.

OPTIONS

-a *affinity*
Relocate files that presently reside on the specified file system *affinity*.

-c *classname*
The policy class associated with the data to be stored, relocated or truncated.

-e Designates the policy as an emergency policy. For a storage policy, this bypasses the policy class "Store Max Set Age", "Store Min Set Size" and "Store Automatically" settings and sets a high priority for immediate action on the storage of the files. For a relocation policy, this disregards the policy class *minreloctime* value. For a truncation policy, this disregards the policy class *mintrunctime* value.

-s Invoke a storage policy.

-r Invoke a relocation policy.

-t Invoke a truncation policy.

-y *filesystemmountpoint*
File system to which policy will be applied. The file system name must be the mount directory for the file system.

-m *minstoretime,minreloctime,mintrunctime*
Minimum time that a file must reside on disk before being considered a candidate for storage (*minstoretime*), or the minimum time a file must remain unaccessed on disk before being considered a candidate for relocation (*minreloctime*), or the minimum time a file must remain unaccessed on disk before being considered a candidate for truncation (*mintrunctime*). If the **-m** option is not used, the default *minstoretime*, *minreloctime*, and *mintrunctime* values will be those specified by the policy class definition. See the MINIMUM TIME FORMAT section below for more info on the format and usage of the minimum time value specified with the **-m** option.

-o *goal* The percentage of used disk space at which a relocation or truncation policy ceases to be applied. It will override the target specified by the **filesystems**(4) configuration parameters.

-v *drivepool*
The *drivepool* from which drives are allocated when storing files.

-b Searches the file system and rebuilds the candidate lists for files needing to be stored, relocated, truncated, and/or stored to an alternate store location.

-A When running a rebuild policy, also look for any failed **fsaddrelation**(1) operations and complete them. Note that this is done after regular rebuild processing has completed.

-C When running a rebuild policy, completely rebuild the candidates lists. As noted before, the default functionality is to search the mdarchive for only the most obvious candidates based on class times etc. Use of this option will result in a complete scan of the file system to verify that all candidates are identified.

-S In certain situations, the store candidate database tables may contain candidates that cannot be removed with a complete rebuild using just the **-C** option. This option will remove all entries in the store candidate database tables before performing a complete rebuild.

-z *minsize*
Minimum file size in bytes for a file to qualify as a relocation or truncation candidate. Only files larger than or equal to the specified *minsize* will be processed.

USAGE

A set of Tertiary Manager daemons will automatically kick off the **fspolicy**(1) command as needed for storing, relocating and truncating files. It is generally not necessary for a user/administrator to run the command.

EXIT STATUS

Exit codes for the **fspolicy**(1) command are:

0 Policy command completed successfully.

1 A policy of this type is already running.

2 Generic policy error.

3 Command syntax error.

4 Specified file system not mounted.

5 Truncation goal not met.

6 Scanning of file system failed.

7 License failure for one or more features.

11 Currently no files to store/relocate/truncate.

MINIMUM TIME FORMAT

The **-m** option for *minstoretime*, *minreloctime* and *mintrunctime* can be specified in units of minutes, hours or days. Specify a value with either an '**m**' suffix for minutes, an '**h**' suffix for hours, or a '**d**' suffix for days. If a unit suffix is not specified, the *minstoretime* value defaults to units of minutes while the others default to units of days. Please be very careful when using the **-m** option, as you may get unintended results. Some valid examples for times are shown below:

15m - 15 minutes
3h - 3 hours
7d - 7 days
10 - 10 minutes *minstoretime*, 10 days *minreloctime*, or 10 days *mintrunctime*

Below is an example of invoking an emergency truncation policy specifying a *mintrunctime* value of 6000 minutes.

fspolicy -t -y /stornext/snfs -e -m 6000m

SEE ALSO

filesystems(4), **fsclassinfo**(1), **fschfiat**(1), **exclusions**(4), **fsaffdf**(1), **fs_mapper**(1),

NAME

fspostrestore − Resync the disk and database after a restore operation.

SYNOPSIS

fspostrestore -s *sTimeString* [**-e** *eTimeString*] [**-d** *directory*] [**-t y**|**n**] [**-o** *outFile*] *mountPoint*

DESCRIPTION

The **fspostrestore**(1) command is used to resync the disk and database after a restore operation. If a managed file system or any part of the database was lost, then this should be executed as the last step of the restore process. If multiple file systems were lost, or the database was lost, this will need to be executed against each managed file system.

This will only resync files located under relation points.

The start time provided to the command is the beginning of the time period to be resynchronized. Only files that have been modified/created since the start time provided will be examined for issues, older files will be assumed to be in-sync. (Note too that some SNSM commands, including media based commands, may affect the time used to select files for processing.) The start time provided should be the last time it was known that the managed file system and the mysql database were in-sync. In general the time stamp of the backup used for the recovery operation is a good choice. If there is reason to believe there were synchronization issues in the backup used for the recovery an older start time can be provided. If unsure about the last time everything was in-sync it is better to err on the side of providing a start time that is much older. Providing an older start time will just result in the command running longer since more files will be scanned and processed. If there is a desire to check all files a start time prior to the initial installation of SNSM can be used.

OPTIONS

mountPoint

The mount point of the file system to resync with the database.

-s *sTimeString*

Indicates the start time of the time range to check during operation. The format for the time parameter is YYYY:MM:DD:hh:mm:ss where:
YYYY = Numeric, year

The following are optional; defaults are shown:,
MM = Numeric, two-digit month
(default:01 (January)),
DD = Numeric, two-digit day
(range: 01-31, default:01),
hh = Numeric hour
(range: 00-23, default:00),
mm = Numeric minute
(range: 00-59, default:00),
ss = Numeric second
(range: 00-59, default:00)

-e *eTimeString*

Indicates the end time of the time range to check during operation. This defaults to the current time and should not normally be set without consulting Quantum technical assistance. The format for the time parameter is YYYY:MM:DD:hh:mm:ss where:
YYYY = Numeric, year

The following are optional; defaults are shown:,
MM = Numeric, two-digit month
(default:01 (January)),
DD = Numeric, two-digit day
(range: 01-31, default:01),
hh = Numeric hour

(range: 00-23, default:00),
mm = Numeric minute
(range: 00-59, default:00),
ss = Numeric second
(range: 00-59, default:00)

-o *outFile*

The output file for the report of the command. The default is */usr/adic/TSM/logs/reports/fspostrestore.out*

-t y|n Indicates if a list of files that are excluded from truncation should be generated. The file created is named *fspostrestore.<file system name>.no_trunc*. If postrestore exclusions are specified (see exclusions man page) then two lists will be generated. The second file name is identical to the first with an additional _filtered suffix. The default is **y** which generates the list.

-d *directory*

This specifies the directory where the list of files excluded from truncation will be generated. The default is /usr/adic/TSM/logs/reports/

NOTES

One of the tasks of this command is to find files that are on disk but not in the Tertiary Manager database. (This can happen for example if an older version of the database backup had to be used for some reason.) Files that are found in this state are removed from disk since no data is left on disk after a file system is restored. One side effect of this behavior is that if a user has any 100% sparse files on disk, those would be removed as part of this processing. (The Tertiary Manager software adds nothing to the database for these files.) A listing of the files removed by the processing can be found in the saved output of the command.

This command runs in two passes. The first pass is where the bulk of the work is done. (Files only on disk are located and removed, version discrepancies are fixed etc.) In the second pass files that are in the database, but not on disk in the location they were when the file was stored are processed. This step of the command basically scans the dump to determine if the files do still exist on disk in some new location. If the files are not anywhere on disk they are made recoverable in the database. At that point they can be brought back to disk by the **fsrecover**(1) command if desired. If there are lots of files in this state (because for example the database restored was more up to date than the disk) and the dump contains many files, then this step can take a long time.

During the first pass of the command no activity should be attempted on the file system being processed. During the second pass activity on the file system can be resumed if it is decided that you do not want to wait for the pass to complete. It is recommended however that both passes are allowed to complete before the file system is put back in use.

WARNINGS

This utility should be used carefully and only under the guidance of Quantum technical assistance.

SEE ALSO

snbkpreport(1), **exclusions**(4)

NAME

fsprobe − Probe and report on configuration, usage and licensing information

SYNOPSIS

fsprobe [-c] [-a]

fsprobe [-c] [-C type**...] [-M** type**...] [-F** type**...]**
[-P type**...] [-L** type**...]**

DESCRIPTION

The **fsprobe**(8) utility will probe a managed system looking for the requested data, then generate a JSON report detailing what was found. The utility basically mirrors and combines the functionality of other Tertiary Manager commands into one report. Some of these commands include: fsconfig, fsdiskcfg, fsddmconfig, fsobjcfg, fsstate, fsclassinfo, fsmedlist and others...

The output for this utility is designed for use by other programs. This utility is not intended for an administrator/user to run directly.

OPTIONS

-c Display the json output in compact form which minimizes its size. This can be used when only machine readable output is needed.

Note that for any of the options below a valid value for the *type* argument is 'all'. This indicates that all of the valid types should be included in the report.

Also, for any of the options below, multiple types may be provided with the option. So for example with the **-C**, the 'arch' and 'sdisk' types may both be specified together. When two or more are specified they are to be separated by spaces.

-C *type ...*

Report on configuration information. Valid values for the *type* arg:

arch - configured archives
sdisk - configured storage disks
obj - configured object storage
ddm - configured distributed data movers

-M *type ...*

Report on media counts. Valid values for the *type* arg:

total - total number of media
used - media currently in use
blank - blank media
avail - media available for storing
suspect - media marked as suspect
protect - media marked as protected

-F *type ...*

Report on managed file systems. Valid values for the *type* arg:

name - the name of the managed file system
fill - block counts, used space percentage
(per tier/affinity)
cfg - target low/high water marks
ffc - is a foreign file system conversion being done?

-P *type ...*

Report on policy classes. Valid values for the *type* arg:

cfg - general configuration info
tiers - info on storage tiers
ops - info on operations (store, trunc and reloc)

-L *type* ...

Report on licensing items. Valid values for the *type* arg:

features - list the available features and provide status indicating whether or not they are authorized and in use
(if no options are specified to the command this is the default report)

capacity - list only the capacity based features (storage tiers and ddms) and in the listing include the capacity info for each of those features

SAMPLE USAGE

Report on all information:

```
fsprobe -a
```

Report the name and fill level of all managed file systems:

```
fsprobe -F name fill
```

Report on configured storage disks and ddms, also all media and policy class information:

```
fsprobe -C sdisk ddm -M all -P all
```

NAME

fsqueue − View subsystem resource requests.

SYNOPSIS

fsqueue [**-r** *requestID*] [**-F** *type*]

fsqueue -m [**-r** *requestID*] [**-F** *type*]

fsqueue -f [[**-r** *requestID*] | [*filename*...]] [**-F** *type*]

fsqueue -a [**-v**] [**-F** *type*]

fsqueue -s [**-q**] [**-i**] [**-c** *time*] [**-r** *requestID*] [**-F** *type*]

DESCRIPTION

The **fsqueue**(1) command displays a report on requests awaiting drive and media resources. Alternatively, it displays data mover host and data mover request reports.

Resource Allocation Report: By default, the **fsqueue**(1) command generates a full report on all drive and media allocation requests currently in the system. If a request ID is specified, it generates a report on drive and media allocation requests associated with that request ID.

Media Operations Report: The **-m** option generates a report on all media operations occurring in the system. If a request ID is specified, it generates a report on the media operations associated with that request ID.

File Processing Report: The **-f** option generates a report on the current state of each file being processed in the system. This report obtains the active request ID associated with a store or retrieve if a filename is known. However, because of potential dependencies, the original request ID for a **fsstore**(1) or **fsretrieve**(1) command issued by a particular user may not be discernible. If a request ID or a list of filenames is specified, it restricts the report to the files associated with that request ID or to the files in the file list.

Mover Host Report and **Mover Request Report:** The **-a** option generates a report on the hosts being used by distributed data movers. When the -v option is added, the report displays detailed information on the state of each data mover.

Status Summary Report The **-s** option generates a summary report on completed requests, queued requests waiting for resources, and requests being processed by data movers. The report is a consolidation of the **fsqueue** and **fsqueue -a -v** reports.

RESOURCE REPORT STATUS

Common Report Elements

Pos The position of the request in the queue. Note, this is only an approximation of the order in which requests will be processed. It is not a guarantee of actual processing order.

Request ID
The request ID.

Request Type
Type of request. Values are:

CPY	Copy files or duplicate media
ENT	Enter Media
EJE	Eject Media
FMT	Format
N/A	Not Available
RTV	Retrieve
STR	Store
UNK	Unknown/unavailable

State The state of the request. Values are:

ALLOCATE	Resource allocated
CANCEL	Being cancel

COMPLETE	Request done
COPY	Start copying
FORMAT	Formatting
MOUNT	Mounting
PROCESS	Being processed
QUEUE	Request in queue
READY	Ready to process
VERIFY	Verify label
QVWISSUE	Waiting to issue restore
QVWRESTR	Waiting for restore complete

Resource Allocation Report

Media Manager ID — The Media Manager request ID associated with the request

Priority C:M:A — The request's current priority level (*C*), maximum priority level (*M*), and age timer (*A*) in minutes for the request. When the age timer expires and the request has not been processed, the priority level of the request will be decremented to the next value. The age timer will be reset and the process will continue until it is processed or the maximum priority field is reached. If the request reaches the maximum priority value, the request will stay at this level until it is processed. Top priority requests specified with **fsretrieve -p** will display the letter 'P' for the current priority level instead of a numeric value.

Drive ID — Component alias drive ID

Media ID — Media ID associated with the request

Submitted Time — Time request was submitted

File Processing Report

File Size — The size of the file in bytes associated with the request

File Name — The pathname and file name associated with the file

Media Operations Report

Media ID — Media ID associated with the request

MOVER REPORT STATUS

Mover Host Report

Host — The name of the host configured for distributed data movers.

State — The state of the host for running data movers. Values are: **Enabled**, **Disabled**.

Active Data Movers — The number of data movers that are currently running on host.

Mover Request Report

Host — The name of the host configured for distributed data movers.

Request ID — The request ID.

Device Alias — The component alias as defined by **fsconfig**(1) for tape devices, the alias as defined by **fsdiskcfg**(1) for storage disks, the alias as defined by **fsobjcfg**(1) for object storage controller devices, or *disk2disk* for relocation operations. For media or file copy operations, the alias for the second device is also displayed. A non-tape device with streams will follow the convention of DevcieAlias.#, where the period followed by a number represents the StorNext I/O stream of the device being used. An object storage device performing a multipart upload operation will use the same convention and additionally display the number of parts concurrently being uploaded in parenthesis such as DeviceAlias.#(#).

Run Time — The length of time the data mover has been running. The format for the time is hh:mm:ss, where:
hh = Number of hours (range: 00-99)

mm = Number of minutes (range: 00-59)
ss = Number of seconds (range: 00-59)

Total Files The number of files to be copied.

Files Copied The number of files successfully copied.

Files Failed The number of files that failed to be copied.

Total Data The number of bytes to be copied.

Data Copied The number of bytes copied.

SUMMARY REPORT STATUS

Summary Status Report

Pos The position of the request in the queue. Note, this is only an approximation of the order in which requests will be processed. It is not a guarantee of actual processing order.

Request ID The request ID.

Request State The state of the request. Reference "Common Report Elements" section for values.

Request Type Type of request. Reference "Common Report Elements" section for values.

Run Time The length of time the request has been processing since it was received up until the time that it completes. The format for the time is hh:mm:ss, where:
hh = Number of hours (range: 00-99)
mm = Number of minutes (range: 00-59)
ss = Number of seconds (range: 00-59)

Total Files The number of files to be copied.

Files Copied The number of files successfully copied.

Files Failed The number of files that failed to be copied.

Total Data The number of bytes to be copied.

Data Copied The number of bytes copied.

Media ID The media allocated to the request

Device Alias The component alias allocated to the request.

Host The distributed data mover host allocated to the request.

OPTIONS

-f Displays all files in the queue or specific files if a request ID or file list is specified.

-m Displays all media operations (**ENT**,**EJE**,**FMT**,**CPY**) in the queue or media operations for a specific request ID.

filename
The *filename* for which the associated resource requests will be reported. A fully qualified pathname should be used unless the file is located in current working directory.

-r *requestID*
The request ID for which the associated resource requests will be reported.

-a Displays summary information on distributed data mover hosts.

-v Displays detailed status on active distributed data movers. When data movers have a small amount of work to perform they may not ever appear in this report. If that occurs, then try showing recently completed activity with:
fsqueue -s -i -c 30s

-s Displays status summary information on requests. By default this will show information on requests that have data movers working on them (**-i**) and are queued waiting for resources to be allocated (**-q**). The output can be adjusted by specifying any combination of the **-i**, **-q**, and **-c** options.

-i This option is only valid with **-s** and will limit the output to include requests that have data movers which are working on requests. When data movers have a small amount of work to perform they may not ever appear in this report. If that occurs, then try showing recently completed activity by also specifying **-c**.

-q This option is only valid with **-s** and will limit the output to include requests that are queued waiting on resource allocation.

-c *time* This option is only valid with **-s** and will limit the output to include requests that have completed within the specified amount of *time*. The time value is specified with a numeric value followed by an optional single character suffix indicating the units. The time value can be set in units of seconds, minutes, hours or days using the following suffix values:
s - seconds (default if no suffix specified)
m - minutes
h - hours
d - days

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fscancel(1), **fsddmconfig**(1), **fsxsd**(1)

NAME

fsrecover − Report or recover Recoverable files on media.

SYNOPSIS

fsrecover *filename...* [**-t** *starttime* [*endtime*]] [**-F** *type*]

fsrecover *dirname...* **-d** [**-r**] [**-a** [**-t** *dirtime*]] [**-F** *type*]

fsrecover [*RM_time::*] *filepathname...* **-u** [**-v**] [**-F** *type*]

fsrecover *dirpathname...* **-u -d** [**-r**] [**-v**] [**-a** [**-t** *dirtime* [**-n** *destination_dir*]]] [**-F** *type*]

DESCRIPTION

The **fsrecover**(1) command can be used to report on Recoverable files. It can also be used to recover those files back to disk. A Recoverable file is a file for which there is still knowledge of its location on managed media, but the file no longer resides on disk (it has been removed.)

A Recoverable file can be recovered up until the time the managed media has been cleaned via the **fsclean**(1) command. At that point the files cleaned are completely gone. When a file is recovered, the version of the file that was active at remove time is made the active version.

The following conditions must exist for a file to be recovered: - The Recoverable file must be stored on media. - The user must have read access to the file and write access to the directory from which the file was stored.

The following list shows the ways to use **fsrecover**(1): - To report Recoverable files and directories that contain them. - To recover these Recoverable files back to disk.

Reporting

The report generated by **fsrecover**(1) lists all Recoverable files and the directories that contain them. Names and wildcards can be used to limit what list is reported.

Note that the report wildcard is '%' so that it will not conflict with the shell wildcard.

The time range option (**-t** *starttime*) extracts only information over the time period specified. The *starttime* and *endtime* must be before the current time, and the *starttime* must be before the *endtime*. If no *endtime* is specified, *endtime* defaults to current time.

If the directory time option (**-t** *dirtime*) is used when reporting on a directory that indicates that only files that were active in the directory at that time will be reported. Note that those files may have been removed since then, or updated and now have a new version. Likewise files that were active before that time, then were removed, will not be included in the report.

Restoring Deleted Files and Directories

Although a report can be generated with a file name or a path name, execution of **fsrecover -u** to recover a file or a directory requires that a full path be indicated. The user cannot change the placement of the recovered file or directory. All files are recovered with the same name and directory in which the files had at the time when removed. After recovering a file or directory, it can then be renamed as usual. (Note that recovering a directory instance with the **-a**, **-t** and **-n** options is an exception to this behavior and will be covered more below.)

When reporting Recoverable files the complete path names are reported along with the time of the remove in the form of (RM_time::PATH_NAME) The expected format for the remove time string "RM_time" is shown in this example:

```
2005:05:31:06:36:44::/stornext/snfs1/dir1/dir2/file.a
```

If there are multiple files with the same path that can be recovered then by default the one with the most recent remove time is recovered. If it is desired to recover an older instance of the file with that name, then the remove time string must be provided with the file.

Files that are recovered are considered stored on media and truncated from disk. The file is not copied to disk, but appears in a listing of the directory contents (UNIX **ls** command). An attempt to read or edit the file results in a copy of the file from a medium to disk, as in all other stored and truncated Tertiary Man-

ager-controlled files. Additionally, if the file is configured to have a stub file, then the stub will not be present until the file is retrieved again.

If a list of files or directories are specified, **fsrecover**(1) processes each file and directory individually, failing invalid entries but continuing to recover what it can.

The **fsrecover -d -u** command, when used with the directory path and without the **-a**, **-t** and **-n** options, recovers the directory and its child files. The **fsrecover -d -u** command will recover the latest instance of all files in those directories. If the recurse option is specified, the Tertiary Manager software attempts to recover all child files and recursively recover subdirectories and subdirectory child files.

Recovering an Instance of a Directory

The **fsrecover -d** command, when used with the **-a**, **-t** and **-n** options, can be used to recover an instance of a directory. This flavor of the command accepts the recursive **-r** option and works like any directory recover with the following exceptions: - Files that are recovered from the directory are recovered to a new location (not where they were located when removed, if indeed they were removed). - Because the files are recovered to a new location they end up as new files with only a limited relation to the existing files. If the original files are still active then any actions such as removal or modification of those files will not affect the newly recovered directory. (The one relation the files do have is they share the same media copies. So if the media that contained the original is lost or the info is removed via **fsrminfo**(1) then both sets of files are gone.)

When recovering a directory instance it can be thought of as recovering the files that were active in a directory at the specified time. Hence the use of the **-a** and **-t** options. Here is sample usage for recovering an instance of directory /stornext/snfs1/relp1/subdir1 from Apr 10, 2010 at 4:30pm: - fsrecover /stornext/snfs1/relp1/subdir1 -duat 2010:04:10:16:30 -n /stornext/snfs1/relp1/newdir

Note that the destination directory does not have to exist but at least its parent directory must exist. Also the destination directory must be on the same file system and be of the same policy class as the original directory.

Note that file versions which have been removed and then recovered, via a normal **fsrecover**(1), will be counted as active from the time the version was first created until the final endtime. This is true regardless of how many times it was updated/removed and re-recovered during that time. Any recovery of the directory instance between those times will result in that file being recovered to the new destination directory.

Lastly note that recovering an instance of a directory should always be done to a new directory. (If a recover operation is stopped for whatever reason it can be restarted to the same destination directory.) The reason for this recommendation is that if an instance is recovered to an existing directory, for any name collisions those files will not be recovered. This is true even if the collisions are with removed files in the destination directory. To ensure getting every file from an instance in time always use a new destination directory.

> For example: if you have recovered an instance of a directory, then accidentally remove file(s) from that directory; you can use the command options not related to recovering an instance of a directory for reporting or recovering those file(s). Don't attempt to re-run the instance recovery again.
>
> Another example: if you have recovered an instance, and you realize that for some of the files you want them from a different instance in time; you will have to recover that instance to a new directory. (You can then move any of those newly recovered files to the desired location.)

OPTIONS

filename...

> The file name(s) of the file(s) to report. The name can be a file name, partial path name, or full path name. The full path name of each file matching the name is reported.

dirname...

> The dir name(s) to report. The name can be a dir name, partial path name, or full path name.

[RM_time::]filepathname...

> Full path name of each file to recover. If the command is being run in the directory desired a full path can be indicated by specifying: ./filename A timestamp of the form returned in the report can

be provided with the file path.

dirpathname...
Full path name of each dir to recover. If the command is being run in the directory desired a full path can be indicated by specifying: ./dirname

-u Indicates recovery processing requested. (Undo the remove.) Required option to recover files or directories. If the option is not specified, a report is produced. Wildcards cannot be used with this option.

-d Indicates that directory processing is requested.

-p DEPRECATED: Previously this option was required to report files that the user had permissions to access verses just files that the user owned. Now the report mode will show all files the user has permission to recover, which is the same behavior as recovery mode.

-r Indicates recursive processing is requested. Recursive processing is available for either a directory report or for recovering a directory.

-t *starttime* [*endtime*]
Indicates a time range to restrict the report length displayed to the user. If an entry to be displayed is a file to which the user has access and is within the time range specified by the user, it is displayed. If specifying a time range, the user must enter a *starttime*, but the *starttime* can not be greater than the current time. If the *endtime* is not specified, the current time is used. If the user specifies an *endtime*, it must be greater than the *starttime*. The format for the time parameter is YYYY:MM:DD:hh:mm:ss where:
YYYY = Numeric, year

The following are optional; defaults are shown:,
MM = Numeric, two-digit month
(default:01 (January)),
DD = Numeric, two-digit day
(range: 01-31, default:01),
hh = Numeric hour
(range: 00-23, default:00),
mm = Numeric minute
(range: 00-59, default:00),
ss = Numeric second
(range: 00-59, default:00)

-t *dirtime*
This indicates the point in time a directory instance should be reported/recovered.

-n *destination_dir*
This indicates the destination directory when recovering a directory instance.

-a When reporting or recovering a directory use active files instead of Recoverable files. Wildcards cannot be used with this option.

-v Verbose mode during a recover. Will report on files recovered.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

RENAME TRACKING

Here is a description of the renameTracking feature wich can be enabled on a managed file system.

Some Microsoft applications end up making temporary copies of operational files while they are being updated. For example while a file myDocument.doc is being updated with Word it may be renamed to a random sequence of characters like ABC123DYXab. When Word completes the updates it will save the new myDocument.doc file and remove ABC123DYXab. When Storage Manager recognizes that file ABC123DYXab has been removed it makes it recoverable but with the current temp name. This makes tracking old instances of myDocument.doc difficult if not impossible to accomplish. To get around this scenario the renameTracking feature is provided. This can be turned on in the file system (see the snfs_config(5) man page) and makes tracking of old instances of these temporary files possible.

When renameTracking is enabled, as a file is renamed it is made recoverable at that time. As an example: in the scenario described above as Word renames myDocument.doc to ABC123DYXab, Storage Manager will make myDocument.doc recoverable with its original name before the rename proceeds. (That renamed instance becomes inactive.) Then after Word is done, the new myDocument.doc will be the new instance of the file and the previous instance will be available for recovery via fsrecover(1) with the correct name.

Note that renameTracking is NOT a general purpose feature for keeping the Storage Manager database up to date with rename activity. Turning it on results in files that are being renamed being re-stored. This is not in general a desirable behavior except in the specific case described.

SEE ALSO

fsclean(1), **snfs_config**(5)

NAME

fsrelocate − Relocate a managed file from one disk affinity to another or change the affinity association of a truncated file.

SYNOPSIS

fsrelocate *filename...* **-a** *affinity* [**-F** *type*]

fsrelocate [-H|-L]-R *directory* **-a** *affinity* [**-F** *type*] [**-V**]

fsrelocate -D *directory* **-a** *affinity* [**-F** *type*] [**-V**] [**-A**]

fsrelocate -B *batchfilename* **-a** *affinity* [**-F** *type*] [**-V**]

DESCRIPTION

The **fsrelocate**(1) command allows a user with write-to-file permission to relocate the specified file(s) from one disk affinity to another, regardless of whether the file has been copied (stored) to secondary media. This command only works on files residing in managed directories.

NOTE: Before using the **fsrelocate**(1) command, the file system must be configured with two affinities, and a policy class must be created using those two affinities (see **fsaddclass**(1) or **fsmodclass**(1)). Refer to the system administrator guide for more details on configuring a system for disk-to-disk relocation.

The nominal method of relocating files is through the execution of periodic relocation policies. However, the **fsrelocate**(1) command allows the user to relocate files at will.

If the specified file has not been truncated from disk, its data blocks will be moved to the disks associated to the specified *affinity*.

If the specified file has been truncated from disk, no data blocks will be moved, but the file's affinity association will be changed to the specified *affinity*. This has the effect of designating the affinity to which the file will eventually be retrieved. This overrides the default behavior, which is to retrieve a file to the first affinity specified by the file's policy class.

The maximum number of concurrent relocation I/O requests that will be executed can be overridden by the **MAX_MOVERS_PER_RELOCATION_REQUEST** parameter in *fs_sysparm*. See *fs_sysparm.README* file for details.

OPTIONS

filename...

One or more files to be relocated. The files must reside in a managed directory. If multiple files are entered, the files must be separated by spaces.

-a *affinity*

The destination affinity. This affinity must be defined for the file system in which the file resides. The file will be relocated to this affinity if it has not been truncated. If the file has been truncated, only the file's affinity association will change.

-A This option can only be used with the **-D** option. It will allocate disk blocks to the files being retrieved in alphabetic order in an attempt to provide sequential block allocation to all the files in the directory.

-B *batchfilename*

The batch file contains the list of entries to be processed. Each entry must be listed one per line. Do not enclose the entry in quotes, use escape characters or any character not in the actual name. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

-D *directory*
Only entries in the specified directory will be processed. This is not recursive.

-R *directory*
The directory from which to start the recursive operation. All entries from the specified directory and any subdirectories will be processed. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-H This mirrors the functionality provided by the 'find' command and indicates that symbolic links should not be followed. It is only valid with the "**-R**" option. (Ex. ... "**-HR** *directory*" ...)

-L This mirrors the functionality provided by the 'find' command and indicates that symbolic links should be followed. It is only valid with the "**-R**" option. (Ex. ... "**-LR** *directory*" ...) This is the default behavior for all forms of this command which includes cases when **-R** is not specified.

-V When running in recursive (**-R**), directory (**-D**) or batch (**-B**) mode, this option will enable more verbose output.

SEE ALSO

fsaddclass(1), **fsmodclass**(1), **fsaddrelation**(1), **fspolicy**(1), **fsfileinfo**(1), **fsstore**(1), **fsrmcopy**(1)

NAME

fsretrieve − Retrieve files from media and place on disk.

SYNOPSIS

fsretrieve [**-c** *copy*] [**-x y**|**n**] [**-t** *affinity*] [**OPTIONS**] *filename...*

fsretrieve *filename* **-n** *newfilename* [**-b** *startbyte endbyte*] [**OPTIONS**]

fsretrieve *filename* **-n** *newfilename* [**-c** *copy*] [**OPTIONS**]

fsretrieve -R *directory*|**-D** *directory*|**-B** *batchfilename* [**-V**] [**-t** *affinity*] [**OPTIONS**]

OPTIONS

[**-a**] [**-g** *restore-type*] [**-O** *restore-tier*] [**-p**] [**-S** *numstreams*] [**-Z** *multistreamsize*] [**-F** *type*]
[**-A**] (only valid with -D)

DESCRIPTION

The **fsretrieve**(1) command retrieves files from media and places the files on disk. If the **-n** *newfilename* parameter is omitted, the files are retrieved and stored on disk using the original path and file name. The user can then transfer the files across the network to the client disk. If the **-n** *newfilename* option is specified, the files are retrieved into new file names on disk in a previously created directory.

The **fsretrieve**(1) command normally retrieves the first copy listed in a configured retrieve order. If the retrieve order is not configured, the primary copy of the file will be retrieved unless the **-c** option is used. If the primary copy is inaccessible or corrupted, an attempt is made to retrieve another copy of the file. A message will be displayed indicating that another copy was retrieved (File copy #2 used for retrieval). The system parameter MAX_RETRIEVE_RETRY_COUNT controls how many retry attempts are made for additional copies of a file in the event one or more copies are corrupted. This value can be changed to meet the needs of each site.

If a file cannot be retrieved using the copies mentioned above, and if the Alternate Retrieval Location feature is enabled, the file will be retrieved from a specified directory on an alternate machine (node). The **-x** option can be used to change the **fsretrieve**(1) behavior when the Alternate Retrieval Location feature is enabled.

If a recursive retrieve is requested using the **-R** option, all files that reside on media in the directory specified and all files that reside on media in all subdirectories will be retrieved and placed on disk. The **-D** option will just process files in the directory without recursion.

If a batch retrieve is requested using the **-B** option, all files specified in the *batchfilename* that reside on Tertiary Manager media will be retrieved and placed on disk.

The **-b** option causes a partial file retrieval. Given a *newfilename*, and a *startbyte* and *endbyte*, the part of the file specified in the byte range is copied onto disk in *newfilename* in the current working directory.

The **-c** option retrieves a specific copy of a file and places it in *newfilename*.

The **-a** option will update the access time of the file upon retrieval of the file. If a requested file already resides on disk, the access time will also be updated.

The **-p** option specifies top priority and will cause all files for the retrieve request to be placed at the top of the retrieve queue. Excessive use of this option could undermine the built-in retrieve starvation prevention mechanisms, which could lead to lengthy retrieve times for certain requests.

OPTIONS

filename...

The required path and file name(s) of the file(s) to retrieve. Each file path must specify a file in a migration directory.

-c *copy* Used to retrieve a specific copy of *filename* if one exists.

-t *affinty*

Retrieve file(s) to a specific disk *affinity* on the file system. The *affinity* must be defined for the file system in which the file(s) reside. An *affinity* denotes a specific disk or set of disks (a stripe group) on which the file system resides.

-n *newfilename*
The new path and file name into which to retrieve the file. NOTE: For systems configured with distributed data movers, using this option will result in the operation being performed on the primary MDC host only.

-R *directory*
The directory from which to do start the recursive retrieve. All files from the specified directory and any subdirectories will be retrieved. Depending upon the number of files in the directory and subdirectories, running this option may use extensive Tertiary Manager resources.

-D *directory*
Only entries in the specified directory will be processed. This is not recursive.

-B *batchfilename*
The batch file contains a list of files to be retrieved. File names should be listed one per line. Each line in the file is read as a string of characters, without interpretation, up to the line terminator. You should not enclose the file name in quotes, use escape characters or any character not in the actual file name.

Depending upon the number of files in the batch file, running this option may use extensive Tertiary Manager resources. This option can be used with the batch file generated by **fspostrestore**(1), which contains files that have been configured to be excluded from truncation. This provides a capability to re-stage those files back on disk.

-b *startbyte endbyte*
Retrieve the portion of a file specified by the *startbyte* and *endbyte* parameters into the new file specified by the **-n** *newfilename* option. The first byte of a file is numbered zero and the last byte of a file is the file length minus one.

So, the following example retrieves the first ten bytes of a file:

```
-b 0 9
```

The following example retrieves the last one byte of a 1,000 byte file:

```
-b 999 999
```

The *startbyte* must be positive and less than or equal to the *endbyte*. The *endbyte* must be less than or equal to the file size minus one.

-a Updates the access time of the requested files.

-g *restore-type*
Specifies the type of retrieval from S3 Glacier, S3 Glacier Deep Archive or Azure Archive storage. For retrievals from Amazon Web Services (AWS) S3 Glacier storage, the types vary by cost and retrieval time, with the quickest retrieval type having the highest cost. Please refer to provider documentation for detailed information on retrieval times and cost. The following types are supported:

standard S3 Glacier or Azure Archive standard retrieval. Allows for data access within 3-5 hours from AWS S3 Glacier, within 12 hours from AWS Glacier Deep Archive storage, or within 15 hours from Azure Archive storage.

expedited S3 Glacier expedited retrieval allowing data access within 1-5 minutes. This option is not supported for data in AWS S3 Glacier Deep Archive storage or for data in Azure Archive storage.

bulk S3 Glacier bulk retrieval. Allows for data access within 5-12 hours from AWS S3 Glacier storage or within 48 hours for data in AWS S3 Glacier Deep Archive storage. This option is not supported for data in Azure Ar-

chive storage.

This option is ignored if unsupported by the media or if the file is not in S3 Glacier, S3 Glacier Deep Archive, or Azure Archive storage.

If this option is not specified, the default retrieval type for the storage media will be used when retrieving from archive storage. See the man page for **fsobjcfg**(1).

-O *restore-tier*
Specifies the restore tier when rehydrating an object from Azure Archive storage. The following tiers are supported:

hot Rehydrate the Azure Archive object to the *Hot* access tier.

cool Rehydrate the Azure Archive object to the *Cool* access tier.

This option is ignored if unsupported by the media.

-p Specifies top priority and will cause all files for the retrieve request to be placed at the top of the retrieve queue.

-x y|n Force change in alternate retrieval location behavior. Using **-x y** forces the alternate location to be used immediately, bypassing the standard copies. Using **-x n** forces the alternate location to be ignored. The command will fail if a file cannot be retrieved from a standard copy.

-V When running in recursive (**-R**), directory (**-D**) or batch (**-B**) mode, this option will enable more verbose output.

-A This option can only be used with the **-D** option. It will allocate disk blocks to the files being retrieved in alphabetic order in an attempt to provide sequential block allocation to all the files in the directory.

-S *numstreams*
Defines the number of streams to use for retrieves when a file in the request is equal to or larger than the size defined with option **-Z**. The value must be in the range of 1 and 64.

-Z *multistreamsize*
Defines the minimum file size for using multiple streams.The value of *multistreamsize* cannot be less than 20MiB. It can include one of the following suffixes:

B	bytes
KB	kilobytes (1000)
KiB	kibibytes (1024)
MB	megabytes (1000^2)
MiB	mebibytes (1024^2)
GB	gigabytes (1000^3)
GiB	gibibytes (1024^3)
TB	terabytes (1000^4)
TiB	tebibytes (1024^4)

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming lan-

guage. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

NOTE

There are 2 phases to a recursive retrieve. Phase #1, called the mapping phase, determines which files in a directory should be retrieved. Phase #2, called the retrieval phase, retrieves the data from the tertiary location. Phase #1 is an exclusive phase and no other recursive retrieves can run at the same time. Because of this, all other recursive retrieves that are submitted will fail until phase #1 on the original recursive retrieve completes. An error similar to âAnother retrieve is busy mapping the file systemâ will be seen if this occurs. Multiple recursive retrieves can be in phase #2 simultaneously. Phase #2 is more time-consuming than phase #1.

SEE ALSO

fsstore(1), **fsaltnode**(1), **fsobjcfg**(1), **fspostrestore**(1), **fsxsd**(1)

NAME

fsretrievestats − Generate a report on successful and failed file retrieves and on checksum errors.

SYNOPSIS

fsretrievestats [-startdate *date*] [-enddate *date*] [-manageddir *dir...*] [-taclogdir *dir*] [-genonly]

DESCRIPTION

The **fsretrievestats**(1) tool parses the Tertiary Manager tac logs and generates a report on successful retrieves, failed retrieves, and checksum validation errors for a specific date range.

In addition to sending the report to stdout, the tool also sends a copy of the report via an SN admin alert email. Anyone signed up to receive admin alerts will receive an email with the report attached.

This tool generates an intermediate raw data file that must be kept up to date with the current Tertiary Manager tac log files. Additionally, the system administrator may want the report tool to run automatically on a periodic basic. To facilitate both of those objectives, edit the crontab for the tdlm user (crontab -e -u tdlm) and add lines similar to the following:

```
# The following entry is the daily fsretrievestats raw data gathering schedule.
0 0 * * * /usr/adic/gui/bin/cmdwrap -NO_END_OF_FILE /usr/adic/TSM/util/fsretrieve

# The following entry is the monthly fsretrievestats report generation schedule.
0 1 1 * * /usr/adic/gui/bin/cmdwrap -NO_END_OF_FILE /usr/adic/TSM/util/fsretrieve
```

The above entries define two cron jobs, each preceded by a comment line. The first job keeps the raw data file up to date. The second job generates the report on the first day of every month at 01:00. If a different report schedule is desired, modify the second job with the desired schedule.

REPORT STATUS

Header

The header displays the title and the date range that the report encompasses.

Summary

The summary displays the total number of Retrieve Successes, Retrieve Failures, and Checksum Errors for each directory.

File Retrieve Failures

This section displays details for each retrieve failure.

Failure Date
: The date the retrieve failure occurred.

Directory
: The directory that contains the file that failed to retrieve. The files are grouped by this directory column, so the directory name is listed only once at the beginning of each group.

File
: The name of the file that failed to retrieve. This will on occasion include the subdirectory path if the file resides within a subdirectory of the listed directory.

Checksum Validation Errors

This section displays details for each checksum validation error. Note: Not every checksum validation error results in a file retrieve failure.

Error Date
: The date the checksum error occurred.

Copy
: The copy number of the file that experienced the checksum error.

Ver
: The version number of the file that experienced the checksum error.

Copy Replacement Date
: The date when the copy was replaced. If the copy has yet to be replaced, "NA" will appear in this field.

Directory
The directory that contains the file that experienced the checksum error. The files are grouped by this directory column, so the directory name is listed only once at the beginning of each group.

File
The name of the file that experienced the checksum error. This will on occasion include the subdirectory path if the file resides within a subdirectory of the listed directory.

OPTIONS

-startdate **date**
The start date (default: first day of previous month at time 00:00:00). This date cannot be over a year away or after the end date. The format for the date parameter is YYYY:MM:DD:hh:mm:ss, where:
YYYY = numeric year
MM = numeric month (range: 01-12)
DD = numeric day (range: 01-31)
hh = numeric hour (range: 00-23)
mm = numeric minute (range: 00-59)
ss = numeric second (range: 00-59)

-enddate **date**
The end date (default: last day of previous month at time 23:59:59). The format for the date parameter is YYYY:MM:DD:hh:mm:ss, where:
YYYY = numeric year
MM = numeric month (range: 01-12)
DD = numeric day (range: 01-31)
hh = numeric hour (range: 00-23)
mm = numeric minute (range: 00-59)
ss = numeric second (range: 00-59)

-manageddir **dir...**
The list of managed directories to report on (default: all relation points, except the StorNext backup directory)

-taclogdir **dir**
The directory containing the Tertiary Manager tac logs (default: /usr/adic/TSM/logs/tac)

-genonly
Generate raw data file only. Do not print the report. This option is meant to be used by a cron job to keep the raw data file updated. The raw data file is an intermediate file used to generate the actual report.

RETURN VALUE

0 Success

>0 An error occurred. Check output for details.

NAME

fsrmclass − Remove a policy class

SYNOPSIS

fsrmclass *class*

DESCRIPTION

The **fsrmclass**(1) command removes a policy class and its relationships from the Tertiary Manager software. The policy class will not be removed when there are media associated with it. The **fsmedlist -c** report can be run to determine if any media are still associated with the policy class.

If non-empty directories are related to the policy class, the policy class will not be removed. The related directories can be displayed using the **fsclassinfo -l** report. A status message is displayed to show which directory relationships were successfully removed and not removed.

This command accepts upper case or lower case policy class names, although the parameter is converted to lower case.

OPTIONS

class Policy class to be removed from the Tertiary Manager software. A policy class name can have a maximum of 16 alphanumeric characters. The special "-", ".", and "_" characters are also permitted.

SEE ALSO

fsaddclass(1), **fsmodclass**(1), **fsclassinfo**(1)

NAME

fsrmcopy − Truncate copy/copies of a file from disk or remove copy/copies of a file from media

SYNOPSIS

fsrmcopy [**-c** *copynum*|**-a**] [**-F** *type*] [**-r]** *filename...*

fsrmcopy [**-c** *copynum*|**-a**] [**-F** *type*] [**-r] -R** *directory* **[-H|-L]**-R *directory*

fsrmcopy [**-c** *copynum*|**-a**] [**-F** *type*] [**-r] -D** *directory*

fsrmcopy [**-c** *copynum*|**-a**] [**-F** *type*] [**-r] -B** *batchfilename*

DESCRIPTION

The **fsrmcopy**(1) command allows a user with the UNIX write-to-file permission to truncate the copy/copies of the specified file(s) from disk after the file is copied to media, or remove copy/copies of the specfied file(s) from media if the file resides on disk.

When only *filename* is specified, then the disk copy of the file is truncated if a valid tape copy exists. If the **-c** option is specified, then the copy of the file indicated by *copynum* will be deleted from the medium where it is stored. The **-a** option may be used to remove all copies of a file from the media where it is stored. A valid copy of the file must reside on disk for these operations to be performed. Additionally, the **-c** and **-a** options are not allowed for Write Once Read Multiple (WORM) media types.

Suggested uses for the **fsrmcopy**(1) command include the following:
- After viewing a migrated file. Viewing the file caused it to be retrieved to disk. If the file is not modified, the disk copy can be truncated.
- After storing the file to medium with the **fsstore**(1) command without using the option to immediately truncate the file from disk.
- Between the application of the storage and truncation policies by the system administrator.

OPTIONS

filename...

One or more files to truncate from disk or remove from media. The file paths must be in a migration directory. The entire path name need not be entered. If preceded by a slash (/) in the command definition, the full path name, starting from the root directory, is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent. If multiple files are entered, the files must be separated by spaces.

-R *directory*

The directory from which to start the recursive operation. All entries from the specified directory and any subdirectories will be processed. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-H This mirrors the functionality provided by the ’find’ command and indicates that symbolic links should not be followed. It is only valid with the "**-R**" option. (Ex. ... "**-HR** *directory*" ...)

-L This mirrors the functionality provided by the ’find’ command and indicates that symbolic links should be followed. It is only valid with the "**-R**" option. (Ex. ... "**-LR** *directory*" ...) This is the default behavior for all forms of this command which includes cases when **-R** is not specified.

-D *directory*

Only entries in the specified directory will be processed. This is not recursive.

-B *batchfilename*

The batch file contains the list of entries to be processed. Each entry must be listed one per line. Do not enclose the entry in quotes, use escape characters or any character not in the actual name. Success for individual items (files, directories, etc) will not be reported. Instead an overall

progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-c *copynum*
The *copynum* of *filename* to delete from a medium. A copy of the file must exist on disk before the specified copy is deleted.

-a All copies of *filename* are deleted from media if a copy of the file exists on disk.

-r The remote copy of *filename* will be invalidated if it exists.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsstore(1), **fspolicy**(1), **fsrmdiskcopy**(1), **fsxsd**(1)

NAME

fsrmdiskcopy − Truncate the disk blocks of a file from disk after the file was stored to a medium.

SYNOPSIS

fsrmdiskcopy [**-F** *type*] *filename...*

fsrmdiskcopy [**-F** *type*] **[-H|-L]**-R *directory*

fsrmdiskcopy [**-F** *type*] **-D** *directory*

fsrmdiskcopy [**-F** *type*] **-B** *batchfilename*

DESCRIPTION

The **fsrmdiskcopy**(1) command allows a user with the UNIX write-to-file permission to truncate the disk blocks of the specified file(s) from disk after the file is copied to media. The cleanup policy is the system administration method of routinely freeing disk space by removing files after being stored on media. The **fsrmdiskcopy**(1) command is used by users to maintain a desired level of disk space by truncating individual files. Files specified for removal from disk with the **fsrmdiskcopy**(1) command must have an exact copy on media.

Suggested uses for the **fsrmdiskcopy**(1) command include the following:
- After viewing a migrated file. Viewing the file caused it to be retrieved to disk. If the file is not modified, the disk copy can be truncated.
- After storing the file to medium with the **fsstore**(1) command without using the option to immediately truncate the file from disk.
- Between the application of the storage and truncation policies by the system administrator.

OPTIONS

filename...

One or more files to truncate from disk. The file paths must be in a migration directory. The entire path name need not be entered. If preceded by a slash (/) in the command definition, the full path name, starting from the root directory, is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent. If multiple files are entered, the files must be separated by spaces.

-R *directory*

The directory from which to start the recursive operation. All entries from the specified directory and any subdirectories will be processed. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-H This mirrors the functionality provided by the ’find’ command and indicates that symbolic links should not be followed. It is only valid with the "**-R**" option. (Ex. ... "**-HR** *directory*" ...)

-L This mirrors the functionality provided by the ’find’ command and indicates that symbolic links should be followed. It is only valid with the "**-R**" option. (Ex. ... "**-LR** *directory*" ...) This is the default behavior for all forms of this command which includes cases when **-R** is not specified.

The directory specified on the -R option must be a managed directory. If it is not managed then a failure similar to the following is reported:

```
# fsrmdiskcopy -R /stornext
FS0058 05 0000004572 fsrmdiskcopy interim: Directory not archival. </s
FS0390 05 0000004572 fsrmdiskcopy failed: 0 out of 0 disk copy removes
```

-D *directory* Only entries in the specified directory will be processed. This is not recursive.

-B *batchfilename*

The batch file contains the list of entries to be processed. Each entry must be listed one per line. Do not enclose the entry in quotes, use escape characters or any character not in the actual name. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsstore(1), **fspolicy**(1), **fsrmcopy**(1), **fsxsd**(1)

NAME

fsrminfo − Marks the media as logically blank and invalidates all copies of files that were stored on the media in the Tertiary Manager database.

SYNOPSIS

fsrminfo *mediaID...* [**-y**]

DESCRIPTION

After the **fsrminfo**(1) command is run on a media, the media is logically blank and the files on the media are no longer accessible. (A user cannot even use the **fsrecover**(1) command to bring a file back.) Truncated files, who have no other copies on another media, will be removed from disk by the follow-on scheduled background processing (see below).

The **-y** option is used to cancel prompting. Since this command cannot be undone, by default a prompt is given for each media to perform processing. This option indicates a positive response to all prompts.

The command itself completes very quickly, but the scheduled background processing (**fsclean -r**) is performed by the system sometime after the command completes. This processing does the removal of truncated files from disk for the provided media (unless other media copies are available), and removes the information for all files on the media from the Tertiary Manager database. The user can manually run the **fsclean -r**; however, care should be used since this processing can be quite time consuming depending on the number of files on the media, and the overall number of files in the system.

During the time after the command is run, and before the scheduled background processing completes, there will be files on disk that no longer have their media data available. If an attempt is made to retrieve one of these files, an error will be reported indicating the media information for the file was removed.

Note that after the command has completed, the media is available for re-use immediately. Regardless of whether or not the background processing has completed.

OPTIONS

-y The command normally prompts before performing the rminfo processing on a media. This command will cause the prompt to be left off and a yes response will be assumed.

SEE ALSO

fsclean(1)

NAME

fsrmrelation − Remove a directory-to-policy class relationship.

SYNOPSIS

fsrmrelation *directory*

DESCRIPTION

The **fsrmrelation**(1) command removes a directory-to-policy class relationship from the Tertiary Manager system. The directory specified in *directory* must be an upper-level boundary of the policy class. It cannot be subordinate to any directory that retains a directory-to-policy class relationship. Before issuing the **fsrmrelation**(1) command, all files and subdirectories in the highest level directory of that particular directory hierarchy associated with the policy class must be deleted so that the *directory* is empty. The *directory* specified in the command remains in the file system unassociated with any policy class.

Removing a policy class relationship from a directory causes it to be considered a "nonmigration" directory by the Tertiary Manager software. Therefore, files that are stored by users in this *directory* after it is disassociated from the policy class are not migrated to media. The relationship can be restored by using the **fsaddrelation**(1) command. The *directory* will again become a migration directory.

OPTIONS

directory

The directory path to be disassociated from the policy class. The *directory* must be less than 256 characters long. The entire path name need not be entered. If preceded by a slash (/) in the command definition, the full path name starting from the root directory is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent.

SEE ALSO

fsaddrelation(1), **fsclassinfo**(1)

NAME

fsrmtickets − Remove service tickets (RAS) and admin alert entries.

SYNOPSIS

fsrmtickets [**-tickets**] [**-admin**] [**-all**] [**-help**]

DESCRIPTION

This utility will remove the closed service tickets (RAS messages), all the service tickets or all admin alerts from the StorNext database.

OPTIONS

Option specifiers need to be complete enough for uniqueness. Therefore **-admin** is equivalent to **-adm** and **-ad**.

The following options are supported:

-tickets Delete the closed service tickets from theStorNext database.

-admin Delete all the admin alerts from the StorNext database.

-all Delete all the service tickets (open and closed) from the StorNext database.

-help Print a usage giving a short description of the fsrmtickets operations.

EXAMPLES

fsrmtickets -tickets

Delete the closed service tickets from the StorNext database.

fsrmtickets -admin

Delete all the admin alerts from the StorNext database.

fsrmtickets -all

Delete all service tickets from the StorNext database.

NAME

fsrmversions − Selectively remove Tertiary Manager inactive file-copy versions

SYNOPSIS

fsrmversions -r [**-o** *report_file*] [**-l**] [**-F** *text|csv*] *FILTERS* [*QUALIFIERS*]

fsrmversions -x *FILTERS* [*QUALIFIERS*]

fsrmversions -C [*request_id*] [**-W**]

FILTERS:

[**-q** *quota*] [**-u** *user*] [**-g** *group*] [**-p** *project*]
[**-m** *media_id*] [**-T** *media_type*] [**-s** *fs_mount_point*]
[**-c** *class*] [**-t** [*end_time*]] [**-D** *directory* [**-R**]]
[**-f** *batch_file* | **-i** *report_file* | *file...*]

QUALIFIERS:

[**-b** *bytes_to_remove* | **-k** *percent_to_keep*]
[**-K** *number_of_versions_to_keep* | **-V** *list_of_versions_to_remove*]
[**-d**] [**-P**] [**-I** | **-B**]

DESCRIPTION

The **fsrmversions**(1) administrative command selectively removes inactive copy versions of files, and will not remove the active (current) copy version of files. This is similar to the function of the **fsclean**(1) command, but with a finer degree of control.

Control over the selection of copy versions to remove comes in two forms: *filters* and *qualifiers*. Filters apply to attributes of copy versions such as user, media type, directory, etc. Qualifiers provide another dimension of control over the actions performed on the copy versions selected by the filters, such as limiting the number of bytes removed, specifying a number of copy versions to keep, and removing the association to an object-store copy without deleting the copy from the object store. Running the command with the **-r** option produces a report that shows the effect of the selection-filtering criteria without removing any copy versions. Running the command with the **-x** option and the same selection criteria will then perform the desired removals.

FILTERS, QUALIFIERS, and OPTIONS

-B

Qualifier to run slower functionality in the background after command exit. Removing copy versions from an Object Storage involves: 1) cleaning out affected database records, and 2) sending delete requests to the Object Storage. When removing copy versions without the **-B** qualifier, the **fsrmversions** command runs until both tasks have been completed. When removing copy versions with the **-B** qualifier, the **fsrmversions** command completes when the copy records have been removed from the Tertiary Manager database. Meanwhile, the sending of delete requests to the Object Storage continues in the background. Background processing can run well past completion of the database operations. This qualifier has no effect when the media is not Object Storage type.

-b *bytes_to_remove*

Qualifier to end processing when the cumulative sum of removed-copy sizes reaches the *bytes_to_remove* threshold. The value can include one of the following suffixes:

B	bytes
KB	kilobytes (1000)
KiB	kibibytes (1024)
MB	megabytes (1000^2)
MiB	mebibytes (1024^2)
GB	gigabytes (1000^3)
GiB	gibibytes (1024^3)
TB	terabytes (1000^4)
TiB	tebibytes (1024^4)

PB petabytes (1000^5)
PiB pebibytes (1024^5)

-C *[request_id]*
Option to check and report the progress of the background task initiated with the **-B** qualifier to send copy-removal requests in the background. It reports the percentage of completion for all running background-removal tasks unless a request id is used to limit the report to one task.

-c *class*
Filter to restrict copy-removal selections to the named *class*.

-D *directory*
Filter to restrict copy-removal selections to the named *directory*.

-d
This qualifier will only process deleted files.

-F *text|csv*
Option to set the format of the **-o** output file to either text or comma-separated-value (CSV). The default is *text*. The format is applied to the rows produced with the **-l** long listing. Refer to the **-o** option description for information about the contents of its file.

-f *batch_file*
A list of absolute file pathnames, one per line, to identify the copy versions to be processed. This cannot be used with the **-i** filter or with a list of files on the command line.

files... Filter to use a list of absolute file pathnames, separated by spaces on the command line, to identify the copy versions to be processed. This cannot be used with the **-i** or **-f** filters.

-g *group*
Filter to use a group name as a criterion for selecting copy versions to process. Groups can be found in the */etc/group* file.

-I Qualifier to ignore removing copy versions from an Object Storage. The database cleanup removes references to the Object Storage, so after this action, there is no method within the Tertiary Manager for deleting the unreferenced copy versions from the Object Storage.

-i *report_file*
Filter to use the **-o** output file from a previous run of *fsrmversions* to identify the copy versions to be processed. Both text and csv formats are usable. This cannot be used with the **-f** filter or with a list of files on the command line.

-K *number_of_versions_to_keep*
Qualifier to exclude this number of versions per file from consideration for removal. The exclusions are in order of most recent end-time value, which may be different than ordering by highest version number.

-k *percent_to_keep*
Qualifier to end processing when the sum of inactive-copy sizes drops to or below the specified percentage. This can be used with the following filters: **-q**, **-u**, **-g**, or **-p**. When used with the **-q** filter, the percentage is relative to the quota definition's hard limit. When used with any other filter, the percentage is relative to the sum of inactive-copy sizes at the start of the **fsrmversions** command's processing. When used with the **-q**, **-u**, **-g**, or **-p**, the default value is 80%. When more than one of those filters is used, the order of precedence is as follows: quota, user, group, and project.

-l
Option to produce the long report output, which lists the selected file copy versions, one per line, in CSV or text format. When used with the **-o** option to create a report file in long format, that file is suitable for use with the **-i** filter.

-m *media_id*

Filter to use a Media ID value as a criterion for selecting copy versions to process.

-o *report_file*

Option to create a report file containing a preamble plus the information that would have printed to standard output. A generation date, format version, and format type are included in the preamble. When used with the **-l** option to create a report in long format, that file is suitable for use with the **-i** filter.

-P

Qualifier to cause a purge of the database name space to be performed. This is a time-consuming process that can be avoided when a quick response from the command is desired.

-p *project*

Filter to use a project name as a criterion for selecting copy versions to process. Project names can be listed with the **smproject**(1) command.

-q *quota*

Filter to use a quota name as a criterion for selecting copy versions to process. Quota names can be listed with the **smquota**(1) command.

-R

Option for the **-D** filter to repetitively search in subdirectories of its *directory* argument, to the bottom of that directory hierarchy, for selecting copy versions to process.

-r

Option to report the results of selection without performing any other actions. By itself, the option produces a summary report. When used with the **-l** option, the report includes the list of selected copy versions.

-s *fs_mount_point*

Filter to limit the search to the file system associated with the directory specified by the *fs_mount_point* argument.

-T *media_type*

Filter to limit the search to the media type specified by the *media_type* argument. Following is the set of possible media types:

AWS
AZURE
LATTUS
GOOGLE
GOOGLES3
S3
S3COMPAT
LTO
LTOW
3592
T10K
SDISK

-t *[end_time]*

Filter to use the time at which a copy became inactive as a criterion for selection. The *end_time* argument value should be the current time or older. When the time specified is in the future, it is changed to be the current time. When the time is not specified, it defaults to the current time.

The format for the time parameter is YYYY:MM:DD:hh:mm:ss where:
YYYY = Numeric, year

The following are optional; defaults are shown:,
MM = Numeric, two-digit month

(default:01 (January)),
DD = Numeric, two-digit day
(range: 01-31, default:01),
hh = Numeric hour
(range: 00-23, default:00),
mm = Numeric minute
(range: 00-59, default:00),
ss = Numeric second
(range: 00-59, default:00)

-u *user*

Filter to use a user name as a criterion for selecting copy versions to process. Users can be found in the */etc/passwd* file.

-V *list_of_versions_to_remove*

Filter to limit the search to a set of inactive-copy version numbers specified as a set of comma-separated values in the *list* option argument. Existing versions of a file can be displayed with the **fsversion**(1) command.

-W Option to be used in combination with the **-C** option to wait on background copy-removal requests that were set up by using the **-B** qualifier.

-x

Option to execute the copy removals according to the provided filters and qualifiers.

INVALID FILTER AND QUALIFIER COMBINATIONS

Most of the filters can be used in any combination. The filters that cannot be used together are: **-f** *batchfile*, **-i** *report_file*, and *file*...

The **-R** argument is not a filter. Rather, it is a qualifier that applies only to the **-D** filter.

The synopsis shows the combinations of options that are valid in the three forms of running this command.

EXAMPLES

Example 1

Operate on files owned by root

```
fsrmversions -r -u root      #Report
fsrmversions -r -u root -l   #Long Report
fsrmversions -x -u root -B   #Execute
```

Example 2

Operate on files in quota 'Quota3'. Remove inactive copy versions from files subject to the quota definition until 80% (default) of the quota hard limit remains.

```
fsrmversions -r -q Quota3 -l
fsrmversions -x -q Quota3 -B
```

Example 3

Operate on files in quota 'Quota3' until 40% of the quota hard limit remains.

```
fsrmversions -r -q Quota3 -k 40 -l
fsrmversions -x -q Quota3 -k 40 -B
```

SEE ALSO

fsclean(1), **fsversion**(1), **smproject**(1), **smquota**(1), **smusage**(1)

NAME

fsschedlock − Command used for locking/unlocking some automated features.

SYNOPSIS

fsschedlock [**-r**]

fsschedlock <**-a**|**-d**> <**-t** *type*> <**-f**|<**-s** *time* <**-e** *time*|**-p** *hrs*>>> <*day*> [...]

DESCRIPTION

The **fsschedlock**(1) command is used for managing the locking of some automated StorNext features. The command can be used to report the lock schedule, report the current lock status, lock out a feature for some defined time period, or to unlock a previously locked feature.

When run with no options or args, the current locking schedule is reported (see SCHEDULE REPORT STATUS). If no features are currently locked, the output from the command will be empty. With the -r option, the current lock status of all features is reported (see LOCK REPORT STATUS).

The **-a** option is used when a lock is to be added for a feature. Note that locking a feature does NOT prohibit the feature from being run by hand. It only stops the feature from being started AUTOMATICALLY. The **-d** option is used when a lock is to be deleted for a feature.

SCHEDULE REPORT STATUS

This is the default report which shows the locking schedule. It only reports on features which have scheduled locks. There will be an entry that corresponds to every lock created for a feature. Features that do not appear in this report will appear in the lock status report with a status of "unlocked"

Feature The name of the feature.

Stime The time of day to start the lock out. A "++++" value indicates the feature has been locked out for the full day.

Etime The time of day to end the lock out. A "++++" value indicates the feature has been locked out for the full day.

Day The day(s) of the week the feature is locked or "all".

LOCK REPORT STATUS

The **-r** option is used to generate this report which shows the current lock status of all automated features.

Feature The name of the feature.

Status This indicates if the feature is currently locked or unlocked.

OPTIONS

-a Add a lock to the feature(s) indicated by the other arguments.

-d Delete a lock from the feature(s) indicated by the other arguments. Typically, locks are removed with the same options that they were added with. It is easy to overlook when the **-f** option should be used to delete a lock. The schedule report is useful to determine what options are needed to remove locks. The following schedule report shows a daily store lock created with **-f** and a daily reloc lock created without it.

```
Feature            Stime       Etime       Day
-------            -----       -----       ---
store          ++++        ++++       all
reloc          0000        2400       all
```

When times of "++++" are shown in the report then the **-f** option is required to delete the lock. See EXAMPLES section for some specific cases.

-r Report the current lock status for all features.

-t *type* The type of feature(s) to have their lock added/deleted. The valid values (and their corresponding feature) are:

store Store Policies (non-scheduled automatic store policies that are run as the system recognizes there are files available for storing)

altstore
Alternate Store Location Remote Copy Transfers

reloc Relocate Policies

foreign Foreign Migration

rebuild Rebuild Policies (Scheduled Feature)

lospace
Low Space Truncation Policies

mintime
Class Mintime Truncation Policies

clninfo Cleanup of marked fsrminfo records (Scheduled Feature)

clnver Cleanup of old inactive file versions (Scheduled Feature)

defrag Defragment media (Scheduled Feature)

p_backup
Partial backup (Scheduled Feature)

f_backup
Full backup (Scheduled Feature)

healthck
Health Check Framework (Scheduled Feature)

spolicy Store policy (Scheduled Feature, a class store policy that runs at the scheduled time regardless of when the last policy was run and whether or not there are currently any files to store for the class)

fsys_upgrade
File system upgrade. This processing is typically run after an overall system upgrade. If there are per file attribute updates required that is handled by this processing. When required this processing will run continuously until the file system upgrades are complete. When locked out that processing will pause until the lock period ends. (Note that an active thread will complete before the lockout takes place. This may take a few minutes.)

activevault
Activevault policy (Scheduled Feature)

archive_cmp
Archive compare utility (Scheduled Feature)

expcopy
Copy Expiration utility (Scheduled Feature)

quota_report
Quota Full Report (Scheduled Feature)

objratelimit
Object Rate Limit (Scheduled Feature)

all All of the above

-f Lock out the feature for a full day.

-s *time* The time of day to start the lock out. The format for the time parameter is hh:mm on a 24-hour clock, where:
hh = Numeric hour
(range: 00-24, midnight is 00 or 24),
mm = Numeric minute

(range: 00-59),
For example: 1:15 AM is 115, 11:30 PM is 2330. For a midnight start time, use 0000. When **-s**, **-f** and **-p** options are not specified then the start time will default to 0000.

-e *time* The time of day to end the lock out. The format for the time parameter is hh:mm on a 24-hour clock, where:
hh = Numeric hour
(range: 00-24, midnight is 00 or 24),
mm = Numeric minute
(range: 00-59),
For example: 1:15 AM is 115, 11:30 PM is 2330. For a midnight end time, use 2400. When **-e**, **-f** and **-p** options are not specified then the end time will default to 2400.

-p *hrs* The size of the lock out period in hours. (Use this for time frames spanning days. Time frames specified with **-s** and **-e** cannot cross day boundaries. The **-p** cannot be used with multiple day args. The day given is the start day. The hours specified cannot be greater than a week.)

day [...] The day(s) of the week to lock the feature. Valid values for the arg are: sun, mon, tue, wed, thu, fri, sat or all.

WARNING

There is a functional dependency that the **defrag** feature has on the **clnver** feature. These two features work together in managing out of date media contents. The **clnver** feature will clean up the database information for old inactive file segments; and the **defrag** feature is used to replace media which become fragmented due to these segments being cleaned. (See the **fsdefrag** man page for more info.) If the **clnver** feature is locked out, but the **defrag** feature is left active; this will result in a waste of resources since no media will become fragmented if the **clnver** feature is not running.

NOTES

If a lock of a feature already exists, another lock cannot be added in the same time frame. For example: if store policies are locked on Mondays from noon to 3PM, another lock cannot be added from 2PM to 4PM. (One can however be added from 3PM to 4PM.)

An exception is made to the above rule for full day locks. A full day lock can be added even if there is already an existing lock. So for the case above, a full day lock can be added for store policies on Monday. (A full day lock cannot be added on top of another full day lock.)

Lock items that are added are treated as single entities. So for example if a lock is added from noon to 4PM, a delete specifying a start time of noon and an end time of 2PM (or a period of 2 hrs) will fail. If that is the desired behavior, the current item will have to be deleted, and a new one from 2PM to 4PM will have to be added.

In the report a start/end time of "++++" indicates the feature has been locked out for the full day.

A "*" before the start/end time in the schedule report, indicates the feature lock has been overridden by full day lockout.

EXAMPLES

Add full day lock of all features every day and then remove it:

```
fsschedlock -a -t all -f all
fsschedlock -d -t all -f all
```

Add a lock of store processing from 1100AM to 400PM on Tuesdays and then remove it:

```
fsschedlock -a -t store -s 1100 -e 1600 tue
fsschedlock fsschedlock
        Feature         Stime           Etime           Day
        -------         -----           -----           ---
```

```
        store            1100            1600           tue
fsschedlock -d -t store -s 1100 -e 1600 tue
```

Add a lock of reloc processing from 900AM Saturdays to 900PM on Sundays. It will not be obvious how this lock was created when looking at the schedule report. The original options can be used to delete the lock or it can be removed in pieces.

```
fsschedlock -a -t reloc -s 0900 -p 36 sat
fsschedlock
        Feature         Stime       Etime       Day
        -------         -----       -----       ---
        reloc            0000        2100       sun
        reloc            0900        2400       sat
fsschedlock -d -t reloc -s 0900 -p 36 sat

Alternatives to delete the lock:
fsschedlock -d -t reloc -s 0900 -e 2400 sat
fsschedlock -d -t reloc -s 0000 -e 2100 sun
```

Add a store lock with specific start time and a default end time. Then remove the lock.

```
fsschedlock -a -t store -s 1800 mon
fsschedlock
        Feature         Stime           Etime           Day
        -------         -----           -----           ---
        store            1800            2400           mon
fsschedlock -d -t store -s 1800 mon

Alternative command to delete the lock:
fsschedlock -d -t store -s 1800 -e 2400 mon
```

Add a store full day lock for every day. Since that the lock is created with the **-f** option, it must also be used to delete the lock.

```
fsschedlock -a -t store  -f all
fsschedlock
        Feature         Stime           Etime           Day
        -------         -----           -----           ---
        store            ++++            ++++           all
fsschedlock -d -t store  -f all
```

Another way to add a store full day lock for every day. By not specifying **-f**, explicit start/end times are used. Unlike the previous example, the **-f** should not be used to delete the lock.

```
fsschedlock -a -t store  all
fsschedlock
        Feature         Stime           Etime           Day
        -------         -----           -----           ---
        store            0000            2400           all
fsschedlock -d -t store  all

Alternatives to delete the lock:
fsschedlock -d -t store -s 0000 -e 2400 all
fsschedlock -d -t store        -e 2400 all
fsschedlock -d -t store -s 0000        all
```

Add a store full day lock for every day with default start and end times. Locks that are created for every

day can also be removed by individual days. When a specific day is deleted from an everyday lock, then the remaining days will be expanded into individual day locks. From that point on the use of "all" to delete the lock will not work. This works the same way if the f option was used.

```
fsschedlock -a -t store all
fsschedlock
        Feature         Stime           Etime           Day
        -------         -----           -----           ---
        store            0000            2400           all
fsschedlock -d -t store sun
fsschedlock -d -t store -s 0000 -e 2400 mon
fsschedlock
        Feature         Stime           Etime           Day
        -------         -----           -----           ---
        store            0000            2400           tue
        store            0000            2400           wed
        store            0000            2400           thu
        store            0000            2400           fri
        store            0000            2400           sat
fsschedlock -d -t store all
FS0693 20 0000001074 fsschedlock failed: Unsuccessful delete operation. Entry not
```

SEE ALSO

fsschedule(1), **fspolicy**(1), **fsrminfo**(1), **fsdefrag**(1) **fsobjratelimit**(1)

NAME

fsschedule − Insert, modify, delete, reset, or report all maintenance features in the Quantum storage subsystem.

SYNOPSIS

fsschedule [**-F** *type*] [**-f** *feature*|**-n** *name*] [**-l**]

fsschedule -a [**-F** *type*] **-n** *name* **-f** *feature* **-p** *period* [**-e** *weekday*|**-y** *monthday*] **-t** *runtime* [**-w** *window*] [**-o** *option*] [**--** *activevault_option ...*] [**--** *archive_cmp_option ...*] [**--** *objratelimit_option ...*]

fsschedule -m [**-F** *type*] **-n** *name* [**-p** *period* [**-e** *weekday*|**-y** *monthday*]] [**-t** *runtime*] [**-w** *window*] [**-o** *option*] [**--** *activevault_option ...*] [**--** *objratelimit_option ...*]

fsschedule -d [**-F** *type*] **-n** *name*

fsschedule -r [**-F** *type*] **-f** *feature*

DESCRIPTION

The **fsschedule**(1) command reports, inserts, modifies, deletes, or resets scheduled features for the Quantum storage system. By default, **fsschedule**(1) will generate a report that indicates when the automated features are scheduled along with the status of the last run for each feature. To see further information about the scheduled features, the **-l** option is available for reporting the next run time and last good run time.

Option **-a** and its associated arguments are used to add a schedule for the specified feature. The name specified with the **-n** option must be unique across all scheduled entries.

Option **-d** is used to delete a schedule. A schedule name is required for this option.

Option **-r** is used to reset schedules of a feature. A feature is required for this option. When a reset is performed, all of the existing schedules are deleted. A default schedule for that feature will be added.

Option **-m** and its associated arguments are used to modify an existing schedule. This option requires a schedule name. The modify option allows the user to change the run time, the window, the period, the weekday (or the monthday), and the policy or rate limit options of a schedule. If the period is changed, the weekday (or the monthday) must be provided. If the weekday (or the monthday) is changed, the period must be provided.

The **fsschedule**(1) command can be run with TSM software active or inactive, but requires the database to operate.

REPORT STATUS

The columns reported by the **fsschedule**(1) command are as follows:

Name	Name of the scheduled feature
Feature	Type of feature. Values are: **activevault**, **archive_cmp**, **clninfo**, **clnver**, **defrag**, **expcopy**, **f_backup**, **healthck**, **objratelimit**, **p_backup**, **quota_report**, **rebuild**, **spolicy**,
Period	Period of the schedule. Values are: **daily**, **weekly**, **monthly**
Day	Day(s) of the week or month. If the period is **daily**, this field is displayed in "XXXXXXX" format, where each X represents a weekday starting from Sunday. The value of X will either be 'Y' (enabled) or 'N' (disabled). For example, 'YNNNNNN' means that the feature runs on Sunday only.
Start Window	The run-time window in "HH:MM - HH:MM" format. The first number is the desired run time. The second number indicates the time span in which this schedule can be started.
Last Attempt	The date and time at which the schedule last expired. This timestamp will be updated regardless if the feature was or was not run.
Status	The status of the last attempted run. Values are: **Successful**, **Running**, **Locked**, **Terminated**, **Failed**, **None** The **Terminated** status indicates that the software was shut down in the middle of its run. The **Failed** status will also list a description of the failure.

FEATURE TYPES

The -f option specifies one of the following feature types:

activevault

Creates a vaulting policy that helps manage the SNSM licensed capacity by periodically vaulting qualified media to a library vault. The **fsactivevault**(1) command is run when scheduled to manage the vaulting process. At each run, the system's SNSM used capacity is computed and checked against licensed capacity. When used capacity exceeds a high-water-mark percentage of licensed capacity, **fsactivevault**(1) initiates vaulting actions until used capacity is below a low-water-mark percentage of licensed capacity. The selection criteria for media to be vaulted is fully configurable by the user. When adding the *activevault* feature, the **--** option must be used to specify the vaulting parameters and criteria for the activevault policy. See the **fsactivevault**(1) man page for more info.

archive_cmp

(archive compare) Invokes the **/usr/adic/MSM/util/archive_cmp.pl** utility, which validates that the Media Manager archive configurations match the Tertiary Manager's configurations. When run within the scheduler, the utility is always run in daemon mode (-d). See the **archive_cmp**(l) man page for more info.

clninfo Clean up files marked for removal by the fsrminfo command. It is equivalent to running **fsclean -r**.

clnver Cleans up old versions and instances of files from the database that have exceeded either an optional per-class cleanup interval or the system cleanup interval. Per-class cleanup intervals can be set with the **fsaddclass** and **fsmodclass** commands. Setting the per-class cleanup interval to zero disables per-class cleanup. The CLEAN_VERSIONS_EXPIRATION system parameter configures the system cleanup interval. When the system parameter is not configured, the system cleanup interval defaults to seven years. The format of the system and per-class cleanup interval values is an integer followed by an interval factor. For example: 90d (90 days), 5w (5 weeks), 7m (7 months) or 3y (3 years). System cleanup applies to all classes, so per-class cleanup intervals should be shorter than the system cleanup interval to have an effect. For a further description of the system parameter see the *fs_sysparm.README* file. For a further description of the per-class parameter see the fsaddclass.1 and fsmodclass.1 man pages. Note that the system cleanup feature is equivalent to running **fsclean -t -P**, and the per-class cleanup feature is equivalent to running **fsclean -c <class> -t** for each class where the feature has been configured. When this feature is turned off or locked out, there is no automated purging of database namespace, so you should use the **-P** option of **fsclean** to replace the system-wide effect of the *clnver* feature. See the **fsclean**(1) man page for more info.

defrag Defragments media that have reached the appropriate fill and fragmentation levels. The **fsdefrag -d -n** <*num*> command is run when scheduled to take care of the defragmentation process. Note that this schedule item should normally be scheduled a few hours after the *clnver* feature. The two work together in managing out-of-date media contents. The *clnver* feature cleans up database information for old inactive file segments. The *defrag* feature replaces media that have become fragmented by cleaning. Note that the DFG_MAX_MEDIA_TO_DEFRAG system parameter is used by the schedule daemon to determine the value for the **-n** <*num*> option when running the **fsdefrag**(1) command. This option limits the number of media an individual *defrag* schedule item will process. A value of 0 for this parameter indicates no maximum, so all fragmented media will be processed. (The **-nP option to fsdefrag**(1) is left off in this case.) See the **fsdefrag**(1) man page for more info.

expcopy Removes a copy of files where that copy is set to expire after the current time exceeds the last access time of the file plus the expiration delta. See the **fsexpcopy**(1) man page for more information.

f_backup (Full Backup) Runs a full backup of the Quantum storage system by executing **snbackup**(1) utility. By default, the full backup is scheduled to run once per week on Sunday night.

healthck (Health Check) Invokes the health check framework to run on all checks listed in the system at the *Light* level. RAS Alerts are sent for each individual health-check failure.

objratelimit (Object Rate Limit) Rate limit data transferred to or from object storage systems. When adding the *objratelimit* feature, the **--** option must be used to specify the rate limit parameters.

p_backup (Partial Backup) Runs a partial backup of the Quantum storage system by executing **snbackup -p**. By default, the partial backup is scheduled to run at midnight every weekday, excluding Sunday.

quota_report (Quota Full Report) Invokes the generation of a quota full report. Report files will be generated in the reports directory, and also attached to an Admin Alert email (if Admin Alert emails are configured).

rebuild (Rebuild Policy) Maps the file systems and rebuilds the candidate lists. The mapping files created are prerequisite for several other TSM commands including fsrminfo (clninfo feature), so it must be run regularly. The rebuild feature spawns rebuild policies for each file system. The default number of concurrently running policies is 2. The sysparm attribute, MAX_CONCURRENT_REBUILDS may be specified to override the default behavior.

spolicy (Store Policy) Invokes a store policy for a specified policy class. The specified policy class must exist. When adding this feature, **-o** must be specified with a policy class on which to run the store policy. The policy is run at the scheduled time regardless of when the last policy was run and regardless if there are currently any files to store for the class.

OPTIONS

-a *add* Add a new schedule. This option requires **-n**, **-f**, and **-p**.

-m *modify* Modify an existing schedule. This option requires **-n**. If **-p** is specified, **-e** (or **-y**) must be specified. If the schedule was for an activevault policy or an object rate limit change, the **--** option could also be specified to modify the existing activevault policy or object rate limit at the same time. Note that the **--** option will have to be respecified entirely.

-d *delete* Delete a schedule. This option requires a schedule name.

-r *reset* Reset all schedules of a feature. This option requires a feature.

-n *name* The schedule name.

-f *feature* The feature to change. Values are: **activevault**, **archive_cmp**, **clninfo**, **clnver**, **defrag**, **expcopy**, **f_backup**, **healthck**, **objratelimit**, **p_backup**, **quota_report**, **rebuild**, **spolicy**

-l List schedules in the long report format.

-p *period* The period of the schedule. Values are:

daily A daily schedule is run on the specified weekday(s) at the specified runtime.

weekly A weekly schedule is run on the same day every week at the specified runtime.

monthly A monthly schedule is on the specified day of the month (-y)* or on the first weekday (-e).

* When the day specified by (-m) is 31, the feature is run on the last day of every month (Jan 31, Feb 28 or 29, Mar 31, Apr 30, ...). When the period type is bimonthly and day 31 is chosen, the runtime is the 15th of the month. When the period type is bimonthly and day 15 is

chosen, the runtime is the 30th (except for February, which is run on the last day of the month).

-e *weekday* The day of the week. Values are: **Sun**, **Mon**, **Tue**, **Wed**, **Thu**, **Fri**, **Sat**

-y *monthday* The day of the month. Values are: **1**-**31**

-t *runtime* The start time of the feature specified in HHMM format.

-w *window* The runtime window specified in HHMM format. By default, this is set to 0100 (1 hour). The runtime window is used if the system is not active when a scheduled time expires. When the system restarts, the schedules that are still within the runtime window are started.

-o *option* The option used by the feature. Currently, only the spolicy feature requires an option, which is an existing policy class.

-- The options specified for the activevault, archive_cmp or objratelimit features. Anything after **--** is unique to the activevault, archive_cmp, or objratelimit features. See the **fsactivevault**(1) or **archive_cmp**(l) or **fsobjratelimit**(l) man page for a complete list of valid options for this section.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

EXAMPLES

To report on a specific schedule named Midnight_CleanVer, issue the following command:

```
fsschedule -n Midnight_CleanVer
```

To report on just the rebuild feature, issue the following command:

```
fsschedule -f rebuild
```

To add a schedule named Morning_CleanVer for clnver feature to run daily, every morning at 2:00 AM, issue the following command:

```
fsschedule -a -n Morning_CleanVer -f clnver -p daily -t 0200
```

To add a schedule named Weekly_CleanInfo for the clninfo feature to run weekly, every Wednesday evening at 11:15 PM, issue the following command:

```
fsschedule -a -n Weekly_CleanInfo -f clninfo -p weekly \
    -e wed -t 2315
```

To schedule a full backup on Saturday at midnight, plus run a partial backup at midnight every day except Saturday, with a 3 hour window for each type, issue the following command:

```
fsschedule -a -n weeklyFullBackup -f f_backup -p weekly \
    -e sat -t 0 -w 300
fsschedule -a -n dailyPartialBackup -f p_backup -p daily \
    -e "mon,tue,wed,thu,fri,sun" -t 0 -w 300
```

To give the rebuild feature a 6 hour run window every first of the month at 10:00pm, issue the following command:

```
fsschedule -a -n MonthlyVersions -f rebuild -p monthly \
    -y 1 -t 2200 -w 600
```

To add a store policy named spolicy_class_1 for a store policy feature to run every day at 4:00AM, against policy class pc_1, issue the following command:

```
fsschedule -a -n spolicy_class_1 -f spolicy -p daily \
    -t 0400 -o pc_1
```

To schedule the archive_cmp feature to run every Saturday evening at 11:00 PM, issue the following command:

```
fsschedule -a -n ac_check_1 -f archive_cmp -p weekly \
    -e sat -t 2300
```

To schedule an activevault policy, named av_policy_1, to run daily at 1:00AM, to vault from an archive named i6k to a vault named vault01, at most 10 qualified media which have been used for copy number 1 and have not been accessed for at least 1 month, issue the following command:

```
fsschedule -a -n av_policy_1 -f activevault -p daily \
    -t 0100 -- -a i6k -v vault01 -limit 10 -copy 1 -age 1month
```

To modify the above av_policy_1 to run daily at 2:00AM, issue the following command:

```
fsschedule -m -n av_policy_1 -p daily -t 0200
```

To further modify the above av_policy_1 to run daily at 4:00AM, to vault from archives i6k_a and i6k_b to vault01, at most 20 qualified media that have not been accessed for at least 1 month, regardless of the copy number, issue the following command:

```
fsschedule -m -n av_policy_1 -p daily -t 0400 -- \
    -a i6k_a,i6k_b -v vault01 -limit 20 -age 1month
```

To request that a cumulative object storage send rate be set to 200 MB/sec and a cumulative receive rate to 50 MB/sec during the day and to request that a cumulative send rate of 400 MB/sec and a cumulative receive rate of 100 MB/sec be set at night, issue the following commands:

```
fsschedule -a -f objratelimit -n objrate-day -p daily -t 0700 -- -s 200M -r 50M
fsschedule -a -f objratelimit -n objrate-night -p daily -t 1900 -- -s 400M -r 100
```

To modify MonthlyVersions to run at 11:00 PM, issue the following command:

```
fsschedule -m -n MonthlyVersions -t 2300
```

To modify MonthlyVersions to run weekly on Sunday, issue the following command:

```
fsschedule -m -n MonthlyVersions -p weekly -e sun
```

To modify MonthlyVersions to have a window of 3 hours, issue the following command:

```
fsschedule -m -n MonthlyVersions -w 300
```

To delete a schedule named weeklyCleanVersions, issue the following command:

```
fsschedule -d -n weeklyCleanVersions
```

To reset all schedules of the clnver feature, issue the following command:

```
fsschedule -r -f clnver
```

SEE ALSO

fsactivevault(1) **fsaddclass**(1), **archive_cmp**(l) **fsclassinfo**(1), **fsclean**(1), **fsdefrag**(1), **fsmodclass**(1), **fsobjratelimit**(1), **fspolicy**(1), **fsrminfo**(1), **fsschedlock**(1), **snbackup**(1),

NAME

fsstate − Report the state of all Quantum storage subsystem drive components and storage subsystems and/or Tertiary Manager software.

SYNOPSIS

fsstate [*componentalias*|**-f**] [**-F** *type*]

DESCRIPTION

The **fsstate**(1) command is a user command that can be executed when Tertiary Manager software is active or nonactive. The **fsstate**(1) command reports the state of the Tertiary Manager software and/or all storage subsystems and drive components configured in the Quantum storage subsystem.

Submitting the **fsstate**(1) command with the *componentalias* option generates a report for a single Quantum components, i.e. drive(s), drive identifier(s), and Media Manager system(s).

The **-f** option reports the state of the Tertiary Manager software. Valid states are *active*, *not active*, *starting*, *stopping*, and *unknown*.

Submitting the **fsstate**(1) command without any options generates a report showing all storage subsystems and drive components currently configured in the Quantum storage subsystem and the state of Tertiary Manager software. The report uses information from both Tertiary Manager software and Media Manager software. The *MEDIA ID* field information is derived from the Media Manager software while the remaining fields are derived from the Tertiary Manager software. The *MEDIA ID* field denotes whether media is mounted or not mounted in a drive by displaying the media identifier. Sometimes, the **fsstate**(1) report may show a drive *IN USE*, but no media identifier is specified. This is a result of request delay timing. The special character '*' next to drive names denote configuration information. The '*' character denotes drives not configured in the Media Manager system.

REPORT STATUS

The parameters listed by the **fsstate**(1) command are as follows:

Component Alias
: The component alias names used to identify storage subsystems and drive components

Drive ID
: Component alias drive identifier

State
: The state of the drive or subsystem:

  UNAVAIL — Not Available

  MAINT — Diagnostics Mode

  ON — Component is On-Line

  OFF — Component is Off-Line

  UNKNOWN — Tertiary Manager software is unable to access drive state information from Media Manager software.

Status
: The status of a drive:

  FREE — No medium mounted

  IN USE — Medium mounted and in use

  DELAYED — Medium mounted but not in use. Dismount of the medium will occur after the delay timer expires unless the medium is needed for another request.

  DISMOUNT — Dismount has been requested for medium.

  CLEANING — Cleaning medium mounted

  USER MOUNT — Medium mounted via **fsmount**(1)

  OTHER — Medium mounted by something other than Tertiary Manager.

FAILED Drive failed

CALIBRATE Drive is calibrating/intializing the mounted media. This is only applicable to LTO media (generation 8 and newer) and only occurs if the cartridge has never been initialized. For LTO 9 media this operation could take up to two hours.

Media ID The medium identifier for any medium mounted in a drive or storage disk. Since Object Storage iopath devices can be used with multiple namespaces concurrently a specific media identifier can not be displayed so "****" are shown instead.

Tertiary Manager Software State

The state of the Tertiary Manager software will be shown at the end of the report and will consist of one of the following:

active Tertiary Manager software is operational

not active Tertiary Manager software is not operational

starting Tertiary Manager software is initializing.

stopping Tertiary Manager software is terminating.

unknown Tertiary Manager software is in an unknown state.

OPTIONS

componentalias

The alias for storage subsystem and drives. The system administrator configures the possible values for component aliases during system configuration or by using the **fsconfig**(1) command. If the *componentalias* contains spaces, use single quotes around the words.

-f Generates a report showing status of the Tertiary Manager software. Valid states are *active*, *not active*, *starting*, *stopping*, and *unknown*.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fschstate(1), **fsxsd**(1)

NAME

fsstats − Used to generate Tertiary Manager system usage reports.

SYNOPSIS

fsstats [**-sparse**] [**-perf**] [**-dir** *directory*] [**-beg** *date*] [**-end** *date*] [**-out** *directory*] [**-overwrite**] [**-<ioType>**]

DESCRIPTION

The **fsstats**(1) is a script to process the information in the log file(s) and generate hourly and daily Tertiary Manager system usage reports.

The **fsstats**(1), with no options, will generate the hourly and daily system usage information for the previous day. The reports focus on stores, retrieves, relocation (disk-to-disk), and media copy (tape-to-tape) requests. The information is appended to the existing report files in the /usr/adic/TSM/logs/reports directory.

In order for the reports to stay current, it is recommended that the **fsstats**(1) script be run via a cron job every day.

The **-beg** and **-end** options are intended to be used to specify a range of dates for which the hourly and daily reports will be generated.

By default the reports will report stats for stores, retrieves, disk-to-disk, and tape-to-tape I/O types. These reports can be limited by specifying specific I/O types using the **-Store**, **-Retrieve**, **-Disk**, or **-Tape** options.

OPTIONS

-sparse Only display non zero statistics. This option is recommended as it makes the reports easier to view

-perf Use the perf_00 log file(s) instead of the trace_01 log file(s). This option is primarily used to support legacy systems where the perf logs were the primary means for obtaining the necessary information for the reports. There should be no need to use the option as the trace_01 log is now the preferred means for gathering the report information.

-dir *directory*

Specifies the directory where the logs are located. The default is /usr/adic/TSM/logs/trace.

-beg *date*

Specifies the start date for which the reports will be generated. If specified, the script will generate the usage reports beginning at 00:00:00 on the specified start date. The format of the date is YYYY:MM:DD where:

YYYY = Numeric, four-digit year
MM = Numeric, two-digit month
DD = Numeric, two digit day

The default value is yesterday 00:00:00.

-end *date*

Specifies the end date for which the reports will be generated. If specified, the script will generate the usage reports ending at 23:59:59 on the specified end date. (See the **-beg** option description for more date related info.) The default value is yesterday 23:59:00.

-out *directory*

Specifies the directory where the reports are generated. The default is /usr/adic/TSM/logs/reports. NOTE: Reports may not be rolled/collected if placed in a non-default directory.

-overwrite

Overwrite reports instead of appending to the reports.

-<ioType> ...

Specifies the types of i/o to display in the report. If not specified, all i/o types will be displayed. The possible values are:

Store Displays statistics for Store requests.

Retrieve Displays statistics for Retrieve requests.

Disk Displays statistics for Disk-to-Disk requests.

Tape Displays statistics for Tape-to-Tape requests.

REPORT STATUS

The following reports will be generated:

daily_file_size_dist	Will report the file size distribution by i/o type for the day.
daily_system_report	Will report the file count, amount of data, effective rate, minimum file size, maximum file size, and average file size by i/o type for the day.
daily_policy_report	For each policy class defined, this will report the number stores, truncates, and relocates, the number of success and failed stores, and the amount of data truncated and relocated for the day.
daily_request_dist	Will report the request distribution by i/o type for the day.
hourly_system_report	Will report the file count, amount of data, effective rate, minimum file size, maximum file size, and average file size by i/o type for each hour in the day.
hourly_policy_report	For each policy class defined, this will report the number stores, truncates, and relocates, the number of success and failed stores, and the amount of data truncated and relocated for each hour in the day.
hourly_request_dist	Will report the request distribution by i/o type for each hour in the day.
hourly_file_size_dist	Will report the file size distribution by i/o type for each hour in the day.

NAME

fsstore − Expedite the storage of a file that currently resides on disk to media.

SYNOPSIS

fsstore *filename...* [**-t** *mediatype*] [**-c** *copies*] [**-f i**|**p**] [**-v** *drivepool*] [**-z** *minsize*] [**-u** *runtime*] [**-F** *type*] [**-S** *numstreams*] [**-T ANTF**|**LTFS**] [**-Z** *multistreamsize*]

fsstore [-H|-L]-R *directory* [**-t** *mediatype*] [**-c** *copies*] [**-f i**|**p**] [**-v** *drivepool*] [**-z** *minsize*] [**-u** *runtime*] [**-F** *type*] [**-S** *numstreams*] [**-T ANTF**|**LTFS**] [**-Z** *multistreamsize*] [**-V**]

fsstore -D *directory* [**-t** *mediatype*] [**-c** *copies*] [**-f i**|**p**] [**-v** *drivepool*] [**-z** *minsize*] [**-u** *runtime*] [**-F** *type*] [**-S** *numstreams*] [**-T ANTF**|**LTFS**] [**-Z** *multistreamsize*] [**-V**]

fsstore -B *batchfilename* [**-t** *mediatype*] [**-c** *copies*] [**-f i**|**p**] [**-v** *drivepool*] [**-z** *minsize*] [**-u** *runtime*] [**-F** *type*] [**-S** *numstreams*] [**-T ANTF**|**LTFS**] [**-Z** *multistreamsize*] [**-V**]

DESCRIPTION

The **fsstore**(1) command expedites the storage of data to media, instead of allowing it to be migrated automatically by Tertiary Manager software. Tertiary Manager software initiates storage of a file to Tertiary Manager media based on the migration parameters of the policy class to which the file is associated. The **fsstore**(1) command activates Tertiary Manager software to initiate the migration of the file to a media. Status is returned to the user upon completion of this command. The user cannot specify the specific medium on which the file is placed, but the user can specify a media type.

The user can specify a number of copies of the file to be stored to media when the command executes. This number can be up to, but not exceed, the number of copies specified as the policy class *maxcopies* parameter for the file. To store more copies of the file than are specified in the default copies parameter in the policy class, the file's copy attribute must be modified. Use **fschfiat** *filename* **-c** to change this attribute. Then the **-c** option can be used with the **fsstore**(1) command to store an additional copy of the file. The **fsstore**(1) command will never store more than one copy unless the **-c** parameter is set. If fewer than the maximum number of copies is specified with the **-c** option, that number is stored to disk. The rest of the copies are stored when policy is applied to the policy class. If the **-c** option is not specified, only one copy is stored when the command is executed, and any other copies are stored when the storage policy is applied. After the store command completes, the **fsfileinfo**(1) command can be used to see on which media the file is stored. Metadata such as copy number, last modification time, file offset, file key, path, segment number, and version, can later be added to the stored file using **fsobjmeta**(1).

The policy-class default file-cleanup action can be temporarily overridden using the **-f** option, provided the file is not marked for exclusion from cleanup. File data can be truncated immediately (**i**) or held for evaluation when the cleanup policy is applied (**p**).

The **-v** option will store files only using drives associated with the specified *drivepool*. This allows the number of drives used for storing to be throttled by only allocating a subset of the drives to the specified drivepool.

The **-z** option allows file size to be considered when determining which files are to be stored. Files larger than or equal to the specified *minsize* will be stored while files less than the *minsize* will not be stored during the command execution.

OPTIONS

filename...

The names of the file(s) on disk to store to media. The file path(s) must also be in a migration directory. The entire path name need not be entered. If preceded by a slash (/) in the command definition, the full path name, starting from the root directory, is required as input to the command. Otherwise, the Tertiary Manager command expands the directory name using the current working directory as the parent. If multiple files are entered, the files must be separated by spaces.

-R *directory*

The directory from which to start the recursive operation. All entries from the specified directory and any subdirectories will be processed. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this

general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-H This mirrors the functionality provided by the 'find' command and indicates that symbolic links should not be followed. It is only valid with the "**-R**" option. (Ex. ... "**-HR** *directory*" ...)

-L This mirrors the functionality provided by the 'find' command and indicates that symbolic links should be followed. It is only valid with the "**-R**" option. (Ex. ... "**-LR** *directory*" ...) This is the default behavior for all forms of this command which includes cases when **-R** is not specified.

-D *directory*

Only entries in the specified directory will be processed. This is not recursive.

-B *batchfilename*

The batch file contains the list of entries to be processed. Each entry must be listed one per line. Do not enclose the entry in quotes, use escape characters or any character not in the actual name. Success for individual items (files, directories, etc) will not be reported. Instead an overall progress status message will be reported periodically with this general format: "Processed <N> out of <TOTAL> entries..." It may take some time before the value of <TOTAL> is known. Until that time, <TOTAL> will be reported as <TBD>. By default, <N> is reported at intervals of 1000. This can be overridden by the system parameter RECURSION_BATCH_REPORT_INTERVAL. The parameter is an integer value. For a further description of the system parameter see the *fs_sysparm.README* file.

-c *copies*

Number of copies of the file(s) to be stored. The value is the total number of copies, including the primary copy of the file. This number cannot exceed the number of copies defined in the policy class *maxcopies* parameter. To store more copies of the file than are specified in the default copies parameter in the policy class, the file's copy attribute must be modified. Use **fschfiat** *filename* **-c** to change this attribute. Then, the **-c** option can be used with the **fsstore**(1) command to store additional copies of the file. If the number of copies stored is less than the number specified by the policy class definition or by the **fschfiat**(1) command, the remainder of the copies are stored when the storage policy is applied.

-f i|**p** The file retention policy for the filename specified. The files can be truncated immediately (**i**) or at policy application time (**p**) once all file copies are stored on a medium. If the **-f** option is not used, the file retention policy will be specified by the policy class definition.

-t *mediatype*

Defines the type of medium to be used for storage. Depending on the type of platform used, the following media types are supported by Tertiary Manager software:

AWS
AZURE
LATTUS
GOOGLE
GOOGLES3
S3
S3COMPAT
LTO
LTOW
3592
T10K
SDISK

If **-t** is not specified, the policy class definition will be used.

-T ANTF|LTFS

Specifies the media format type that will be used when formatting or selecting media. **ANTF** is the Quantum internal tape format. **LTFS** is the Linear Tape File System specification tape format. A value of **ANTF** will format media with a single partition containing ANTF volume labels. This data partition will store StorNext file data. A value of **LTFS** will format media with two partitions containing LTFS volume labels. The index partition will store LTFS metadata and the data partition will store StorNext file data. The **LTFS** media format type is only applicable to LTO media generation 5 and beyond (e.g. LTO-5, LTO-6, etc). Any attempt to assign **LTFS** to non-LTO media types will result in an error.

If the **-T** option is not specified, the default will be determined as follows:

A media type of Object Storage will default to **NONE**.
A media type that does not support **LTFS** will default to **ANTF**.
A media type that supports **LTFS** will default to the policy class copy definition if it is valid, otherwise it will default to the system parameter CLASS_MEDIA_FORMAT.

-v *drivepool*

Media Manager drive pool group used to store the file specified. The drive pool must be defined in Media Manager software. If the **-v** option is not used, the default drive pool group will be specified by the policy class definition. The special "_" character is permitted to identify the drive pool group.

-z *minsize*

Minimum File Size in bytes to be stored. Files larger than or equal to the specified *minsize* will be stored. Files with a size less than specified *minsize* will not be stored.

-u *runtime*

Maximum allowable time in hours for the command to finish. This command normally runs until it completes. This option can be used to limit how long it should remain active. If the store has not completed in the specified amount of time, then any outstanding activity will be canceled.

-V When running in recursive (**-R**), directory (**-D**) or batch (**-B**) mode, this option will enable more verbose output.

-S *numstreams*

Defines the number of streams to use for stores when a file in the request is equal to or larger than the size defined with option **-Z**. The value must be in the range of 1 and 64.

-Z *multistreamsize*

Defines the minimum file size for using multiple streams.The value of *multistreamsize* cannot be less than 20MiB. It can include one of the following suffixes:

B	bytes
KB	kilobytes (1000)
KiB	kibibytes (1024)
MB	megabytes (1000^2)
MiB	mebibytes (1024^2)
GB	gigabytes (1000^3)
GiB	gibibytes (1024^3)
TB	terabytes (1000^4)
TiB	tebibytes (1024^4)

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See

http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

SEE ALSO

fsretrieve(1), **fsfileinfo**(1), **fschfiat**(1), **fsclassinfo**(1), **fspolicy**(1), **fsxsd**(1) **fsobjmeta**(1),

NAME

fstypes − Displays valid drive and media types.

SYNOPSIS

fstypes -h

fstypes -d

fstypes -m [**-f** *format*] [*mediatype*]

DESCRIPTION

The **fstypes**(1) command will display the valid drive or media types to be used in the StorNext system.

OPTIONS

-h Help. Shows usage.

-m Lists the valid media type(s).

-d Lists the valid drive types.

-f *format*

The format of the media type to display. Valid options:

tsm Tertiary Manager

msm Media Manager

display Graphical User Interface

mediatype

Used to specify a media type to convert to the specified format.

EXIT STATUS

0 Command completed successfully.

1 Command failed.

NAME

fsusedspace − Report showing total amount of stored primary copy data in the Quantum storage subsystem.

SYNOPSIS

fsusedspace

DESCRIPTION

The **fsusedspace**(1) command shows the volume of primary copy data stored on media in the Quantum storage subsystem. The value is shown in Gigabytes (GB).

Primary copy data includes: - Space occupied by deleted files or older versions of existing files. - Fragmented space from files that were logically removed from media using **fsclean**(1) or replaced by **fsfilecopy -r**, but the source medium contains other files. - Used space on checked-out media.

Used storage can be reduced by defragmenting media using **fsmedcopy**(1), or executing **fsrminfo**(1) for checked-out media no longer necessary.

SEE ALSO

fsmedcopy(1), **fsrminfo**(1), **fsclean**(1), **fsfilecopy**(1)

NAME

fsversion − Report or change versions of a file

SYNOPSIS

fsversion -h

fsversion [**-av**] *name*

fsversion -c *ver* [**-f**] *name*

DESCRIPTION

The **fsversion**(1) command will generate a report of valid versions for a file. By default the report only displays the current version If *-a* is specified, then all the inactive versions are displayed too.

The current version can also be changed using this command with the **-c** option. When changing versions the current version will be truncated from disk, if is not already migrated. Then the current version becomes an inactive version, and the specified new version becomes the current version. The new version will appear as migrated, so it will have to be retrieved before the data can be used. Additionally, if the file is configured to have a stub file, then the stub will not be present until the new file version is retrieved and truncated again.

If the current version only exists on disk and does not have a valid tape copy the command will fail unless the **-f** option is used. The use of the **-f** option will prevent any further access to the data that was in the file unless a copy of that data resides somewhere else.

If the current version has a valid Alternate Store Location remote copy, the command will fail unless the **-f** option is used. The use of the **-f** option will invalidate the remote copy and exclude the new version of the file from being considered as an Alternate Store Location candidate. The fschfiat(1) command can be used to enable the Alternate Store Location remote copy for the updated version of the file and add it to the Alternate Store Location candidate list.

OPTIONS

name... The file name(s) of the file(s) to operate on. The *filename* can be a file name, partial path name, or full path name.

-h Displays usage.

-a Lists all available versions for the file which includes current and inactive versions.

-v Verbose listing. This will also display the following information for each version: file modification time associated with the version, the time the version became inactive, and the time each version was stored or recovered. If a file has never been removed or if the file has been removed but has not been recovered, then the store time is displayed. If a removed file has been recovered back to disk then the recover time is displayed.

-c *ver* This will change the current version to the specified version.

-f This forces the version change if the current version has not yet been stored on tape. This is dangerous and will prevent any recovery of the data that is currently on disk. If this is not the desired result, then the current version can be stored using **fsstore**(1) or a copy made using any standard operating system commands such as cp.

SEE ALSO

fsrecover(1),**fschfiat(1),**altstoreadd(1)

NAME

fsvsdiff − Report discrepancies between media entries in the Tertiary Manager mediadir database table and media in the Media Manager database tables.

SYNOPSIS

fsvsdiff [*mediaID* ...]

DESCRIPTION

The **fsvsdiff**(1) utility reports discrepancies between media entries in the Tertiary Manager mediadir database table and media in the FX_XXX_DATA, FX_XXX_REMOVE, and FX_XXX_MIGRATE media classes in Media Manager database tables.

This utility does NOT check media user criteria fields. It only checks for correct media class associations.

REPORT STATUS

The report output will look similar to the following:

```
NOTICE: the Tertiary Manager software is active.
This will impact system performance. Do you wish to continue (y/n): y
000004 Unknown by Media Manager software
000334 found in MC: F0_LTO_DATA should be in MC: F0_LTO_MIGRATE
001235 found in MC: F0_LTO_MIGRATE should be in MC: F0_LTO_DATA
002341 Unknown by Tertiary Manager software
012591 Unknown by Tertiary Manager software
```

OPTIONS

mediaID ...

List of media IDs to test. If not specified, all media IDs in the Tertiary Manager mediadir database table will be tested.

RESTRICTIONS

Media Manager must be up to run this utility.

Tertiary Manager can be up or down to run this utility.

Because this utility can degrade performance, it is recommended that **fsvsdiff**(1) be run when Tertiary Manager is down or when there are no queued mount requests.

NAME

fsvssync − Correct discrepancies between media entries in the Tertiary Manager mediadir database table and media entries in the Media Manager mediacriteria database table.

SYNOPSIS

fsvssync [*mediaID* ...]

WARNINGS

This utility should be used VERY carefully and only under the guidance of Quantum technical assistance.

DESCRIPTION

The **fsvssync**(1) utility will go through the specified media or all media (depending upon what is passed to the utility) and update the entries in the Media Manager mediacriteria database table so that the values match the values in the Tertiary Manager mediadir database table.

OPTIONS

mediaID ...

List of media IDs to synchronize. If not specified, all media IDs in the Tertiary Manager mediadir database table will be synchronized.

EXIT STATUS

0 Media was successfully updated.

1 No media could be successfully updated.

RESTRICTIONS

Media Manager must be up to run this utility.

Tertiary Manager cannot be active to run this utility.

NAME

fsxsd − Generate XSD files for commands in the Quantum storage system. An XSD is an XML Schema Definition and is often used to validate XML.

SYNOPSIS

fsxsd -h

fsxsd -l

fsxsd *command*

DESCRIPTION

The **fsxsd**(1) generates an XSD file for a given command.

OPTIONS

-h	Print the **fsxsd**(1) usage (help)
-l	Print the commands that have XSD output.
command	Command to generate the XSD specification for.

EXAMPLE

You can use the **fsxsd**(1) command, the *command*, and the Linux command *xmllint* to validate XML output against the XML schema.

For example, these commands create XML and XSD output and then validate the XML against the XSD using the Linux *xmllint* command.

```
fsfileinfo -F xml /stornext/path/and/file > fsfileinfo.xml
fsxsd fsfileinfo > fsfileinfo.xsd
xmllint --schema fsfileinfo.xsd fsfileinfo.xml
```

SEE ALSO

fscancel(1), **fsfileinfo**(1), **fsmedinfo**(1), **fsmedlist**(1), **fsqueue**(1), **fsrelocate**(1), **fsretrieve**(1), **fsrmcopy**(1), **fsrmdiskcopy**(1), **fsstate**(1), **fsstore**(1), **fsobjcfg**(1)

NAME

fs_mapper − Generate a new map for the indicated Tertiary Manager file system or relation point directory.

SYNOPSIS

fs_mapper [**-h**] [**-v**] [**-m**] *mountpt*

DESCRIPTION

The **fs_mapper**(1) command will generate the map files for the indicated Tertiary Manager file system or relation point directory.

There are Tertiary Manager commands, for example **fsclean -r**, that require a map of a file system to run. A file system map is a snap shot of the file system name space and metadata. The data for the map is kept in a set of mapping files. This map can be searched and processed much more quickly than by scanning the file system itself.

This command is typically only run manually if mapping files have been lost or if there is reason to believe the map has become corrupt. The maps are kept up to date automatically by the **fs_scheduler** daemon, which kicks off a rebuild policy on a regular basis after a backup is run. The mapping process used by the rebuild policy is analogous to running this command with the **-m** option.

A file system map can be generated from the backup (metadata dump) of a file system or by scanning the file system directly. The map can be generated more quickly from the backup than by scanning the file system. If however the backup is out of date, generating the map from the file system is more accurate. This command by default generates the map directly from the file system.

NOTE: While the processing of an existing map is efficient and quick compared to scanning a file system, the process of generating the map is slow. It also uses many cpu and I/O resources. Be aware of this when running the command by hand. Because the mapping process is time consuming any command that creates a map, including this one, will output a msg at the beginning of the mapping process to let the user know mapping is taking place. The user will observe a message similar to one of the two messages below when mapping begins. The first indicates the map is being created from the backup (metadump) for the file system and the second indicates the map is being created directly from disk.

```
Creating map for /stornext/snfs1 ....
```

or

```
Creating map for /stornext/snfs1 (from fs) ....
```

OPTIONS

-v Run the command in verbose mode.

-m Create the new map from the backup (metadump) for the file system.

mountpt

The mount point of the Tertiary Manager file system to be mapped.

The **-v** option indicates that the verbose mode of the command should be used. In this mode many of the errors that may be encountered when mapping are reported as output instead of only being logged to the TAC logs.

The **-m** option indicates that the map should be created from the file system backup (metadump) instead of scanning the file system directly. As indicated this is the more efficient of the two methods.

NAME

health_check − Perform a series of tests to ensure StorNext application health

SYNOPSIS

health_check [**-level** *aLevel*] [**-type** *Type Name*]... [**-component** *aComp*]... [**-ras**] [**-verbose**] [**-stopOnError**] [**-output xml**|**std**]

health_check -report -verbose [**-level** *aLevel*] [**-type** *"Type Name"*]... [**-component** *aComp*]... [**-stopOnError**] [**-output xml**|**std**]

health_check -describe levels|**types**|**components** [**-output xml**|**std**]

health_check -help

DESCRIPTION

This utility runs a set of health check operations to determine the health of the StorNext application. The arguments to *health_check.pl* control which operations get run and also control the way in which they run.

Arguments that have embedded spaces, such as in **-type** *"Type Name"*, must be surrounded by quotes. If the argument contains no spaces, the quotes are optional.

Health check operations that can run are defined in control files named *filelist* whose location is described by: <basepath>/<StorNextComponent>*/config/filelist* (i.e., */usr/adic/TSM/filelist*). These files contain health check entries, one per line, that represent health check operations to be run.

OPTIONS

Option specifiers need to be complete enough for uniqueness. Therefore **-level** is equivalent to **-lev** and **-l**.

The following options are supported:

-level *aLevel*

Runs all health check operations at or below the specified level. Three values are allowed: **0**, **1** or **2**. If no **-level** option is specified, the default action is to run health check operations at level 0. Multiple **-level** specifiers are not allowed.

The values **light**, **heavy**, and **excl** (or **exclusive**) can be used in place of **0**, **1**, and **2** respectively.

When a particular level is chosen, all health check operations at OR BELOW that level are run. Therefore, choosing **-level 1** would run all health check operations at either level 1 or level 0.

Each level is defined to mean the following:

0 - Light : Operations at level 0 should have no or minimal impact on StorNext application business operations. Users would not be expected to notice that the health check is running at this level.

1 - Heavy : Operations at level 1 are expected to have noticeable impact on the performance or behavior of the system. However, the impact is not expected to be severe. While these operations may delay user business operations somewhat, the user operations are guaranteed to still succeed.

2 - Exclusive : For operations at this level, it is expected that there is no other activity on the system except for the health check. It is recommended that the system be made unavailable to clients. Any client operations that are concurrently running cannot be ensured to complete successfully or reliably.

-type *"Type Name*"

Run the health check operations of a particular type. If no **-type** option is specified, the default action is to include ALL types.

Multiple **-type** specifiers are allowed.

-type specifiers are case insensitive. **-type** specifiers that contain spaces must be surrounded by quotes. If there are no spaces, the quotes are optional. All of the following values for **-type** "*Type Name*" are equivalent:

-type "Disk Space"
-type "disk space"
-type "DiskSpace"
-type DiskSpace

Use **health_check.pl -describe types** to get the complete list of types. The quotes will not appear in this listing of type names when -output std is specified (default).

-component *aComp*

Specifies a particular component for which health checks are to be run. If no **-component** option is specified, the default action is to include all components.

Multiple **-component** specifiers are allowed.

Use **health_check.pl -describe components** to get the complete list of components.

-verbose

If **-verbose** is specified, additional detailed output is written to the output stream.

-report -verbose

If **-report -verbose** is specified, instead of each test executing, information describing each test is printed, one description per line. If **-report** is specified and **-verbose** is not specified, the current behavior of health_check.pl is to print no output.

-ras If **-ras** is specified, **health_check.pl** will cause each specific health check operation failure to send a RAS Alert appropriate to the type of failure.

-stopOnError

If **-stopOnError** is specified, **health_check.pl** will halt after the first health check operation fails. Otherwise, it will run through all tests regardless of failures of any individual health check operation.

-output xml|std

If **-output** is specified, its specifier must be one of xml or std. If xml is specified, all output will be printed in xml format. If std is specified, all output will be printed in normal paragraph form. The default is std.

-help Print a usage giving a short description of the health check operations. Invalid combinations of option specifiers will also print the usage.

-describe levels|types|components

Describe (show information) for the indicated aspect of the health check framework. Only one of the given aspects can be described at a time.

levels: information about all levels will be listed, one level per line:

0 : Light (minimal impact on system)

1 : Heavy (major impact on system)

2 : Exclusive (all users must be off the system)

components: scan across all components, and for each that contains a *filelist* file in its config directory, the component will be output. The list will contain one component per line, in alphabetical order.

types : scan all *filelist* files across all components, and gather all **type** elements that are found in health check entries. List them in alphabetical order one per line. Types that differ only in case or in spaces vs underscores are considered equivalent and will only be output once. The format of the listed element will be defined by:

A. Convert all underscores to spaces (create words)

B. The first letter of each word is uppercase.

C. The other letters of each word are lowercase.

Therefore, "disk space" would be output as "Disk Space".

EXIT STATUS

Exit codes for the **health_check.pl** command are:

0 All tests were run; No errors or warnings occurred.

1 At least one health check operation failed.

2 There were no failures, but there was at least one warning during the health check run.

Other An error occurred in the execution of the health check framework itself, or a health check operation returned an unexpected exit code (not 0,1,2).

EXAMPLES

health_check.pl -c DSM -c TSM -l 1 -t Network -t "Disk Info" -t "Disk Space"

Run all level 1 and level 0 operations of type "Network", "Disk Info", and "Disk Space" from the DSM and TSM components. NOTE: These examples use the minimum acceptable characters for option specifiers. **-c** is the same as **-component**. **-t** is the same as **-type**.

health_check.pl -t "Disk Info" -c DSM -t Network -c TSM -l heavy -t "Disk Space"

The options are ordered unusually, but the result is exactly the same as the above example. Note the use of the alias "heavy" for level 1.

health_check.pl -c DSM -c TSM -l 1 -t Network -t "Disk Info" -t "Disk Space" -r -v

The same as the first example, but with the "-report" option included. None of the tests will actually be run, but a short description of their purposes will be listed instead.

health_check.pl -c DSM -c TSM -c DSM -l 0 -t "Disk Info" -t "Disk Space"

Invalid. -component values cannot be duplicates.

health_check.pl -c DSM -c TSM -l 0 -t Network -t "Disk Info" -l 1

Invalid. Multiple -level specifiers not allowed.

health_check.pl -c DSM -c TSM -l light -t Network -t "Disk Info" -t "disk info"

Invalid. -type values cannot be duplicates. Note the comparison is case insensitive.

health_check.pl -l light

Run the health check operations of level 0 for all types, across all components.

health_check.pl -l light -ras

Run the health check operations of level 0 for all types, across all components. For each health check operation that fails, send a RAS Alert appropriate for the type of failure.

health_check.pl -l

Invalid. "-l" requires a value.

health_check.pl -d types

Describe (list) all the type values that are found in all component filelist files.

health_check.pl -d types -c TSM

Invalid. A -d specifier cannot be used with any other specifier.

health_check.pl -d type -d level -d comp

Invalid. Only one -d value is allowed at a time.

THE FILELIST FILE And Health Check Section And Entries:

In *filelist* files, the **health_check** section lists the operations (executable binaries or scripts) that will be run by the **health_check.pl** utility. Each entry represents one or more individual health check operations.

Each entry below must conform to the following syntax or it will be ignored:

health_check : <level> : <type> : <execution string> : <timeout> where

level : must be one of 0, 1, or 2.

type : Each operation type should be a short but readable phrase and can allow spaces. All of the following examples are equivalent:

Disk Space

disk space

DISK SPACE

DiskSpace

execution expression : A pathname for an executable to be run. The rules for expressing a pathname are as follows, IN ORDER:

- If the pathname has no "/" or "*" characters, it is an atomic executable name that will be found via the PATH.

- If the path starts with a "/", it is considered a fully specified path. Otherwise, it is considered relative to the component containing that filelist file.

- If the path specifies a directory (and does not contain wildcards) then all executable files in the entire subtree of that directory are selected.

- Wildcarding (the use of the "*" character) is allowed. It represents the selection of one or more executable files. However, wildcarding does not apply to directories.

- Arguments to the executable follow the file name.

- Note wildcarding does not select on directories, only on executables, so directory recursion is not possible with wildcarding.

timeout : The number of seconds the operation is allowed to run, after which it is considered to have failed. A value of zero indicates no timeout (run to completion).

Each individual health check operation that is represented by a filelist entry in the health_check section must conform to a set of rules:

- It must return a status code of 0 for success, 1 for failure, or 2 for warning. Any other returned status code will be considered a failure (1).

- It must "clean up" prior to exiting, leaving the system in the same state that it was in upon starting.

EXAMPLES OF FILELIST health_check SECTION ENTRIES:

These examples show a few of the ways that **health_check** section entries can be written to correspond to the ways that health check operations (scripts and executables) can be organized in directory trees.

health_check : 0 : Disk Space : util/hchk/diskspace/light_* : 30

All binaries and scripts in the util/hchk/diskspace/ directory whose names begin with "light_" are organized as being of intrusion level 0 and of type "Disk Space". If they run, each is expected to complete in less than 30 seconds.

health_check : 1 : Disk Space : util/hchk/diskspace/heavy_* : 180

This entry is the same as the above except the operation names begin with "heavy_", are at intrusion level 1, and each is expected to complete within 180 seconds.

health_check : 0 : media : exec/fsmedchk -a -gen 2 -perdrive : 60

This entry identifies one specific health check operation, "exec/fsmedchk". It also specifies arguments to that operation.

health_check : 1 : policy : systest/policy -c tempClass : 300

This example illustrates "directory" recursion - under the assumption that *systest/policy* is a directory. All operations anywhere within the entire directory subtree of the *systest/policy* directory are chosen (at intrusion level 1 for type "policy"). Every such operation will be passed "-c" "tempClass" as its arguments.

NAME

keydb_init − Will initialize the Tertiary Manager keydb table.

SYNOPSIS

keydb_init

DESCRIPTION

The **keydb_init**(1) utility is used to initialize the Tertiary Manager keydb table after the database is reinitialized.

WARNINGS

If the database has not been reinitialized, there is no reason to run this utility.

Users should not run this command, unless directed to do so by Quantum technical assistance.

NAME

maptst − Examine the mdarchive or Tertiary Manager mapping files

SYNOPSIS

maptst [**-a**] **list**|**mlist**

maptst <**-g**|**-m**> [**-b**] *mount_point*

maptst <**-i**|**-r**> *map_dir mount_point*

maptst [**-R**] *mount_point*

DESCRIPTION

This command can be used to browse the contents of the current metadata archive database (mdarchive) or the current set of mapping files for a file system. The default is to browse the mdarchive. For browsing a menu is displayed allowing the user to traverse through the mdarchive (or mapping files) examining their contents.

This command can also be used to generate a new set of mapping files before continuing to the browsing of those files. The mapping files can be generated from the current backup (metadata dump) or from the file system directly.

Finally the command can be used to list current file systems. It can list mounted and unmounted file systems. Also StorNext file systems and regular file systems can be listed. Some basic file system attributes are provided in the list.

MAP GENERATION

As mentioned, the default operation for this command is to browse the mdarchive. (No mapping files are generated.) It is possible however with the "**-g**" or "**-m**" options (or the directory specific options) to generate a set of mapping files before proceeding to the browsing operation. See the man page for **fs_mapper**(1) for more info on map generation.

Note that if map generation is desired it is only done via the command options. There is a mapping system parameter (MAPPING_SOURCE) which when set will affect other processes that generate the mapping files. That sysparm is ignored by this command. Map generation is done (or not done) purely by the command options provided.

OPTIONS

-g This will cause new mapping files to be generated directly from the file system instead of just browsing the mdarchive.

-m This will cause new mapping files to be generated from the current backup (metadata dump) instead of just browsing the mdarchive.

-R The default behavior of the command is to browse the contents of the mdarchive. By specifying this option the existing set of mapping files will be read and browsed instead. This option not valid with the others as they all imply their own browsing behavior.

-b This will cause the mapping processing to scan for candidates (analogous to the rebuild policy) while the map files are being generated.

-i *map_dir*

This will map the specified directory using inodes instead of file keys and will always generate new mapping files. (It will not affect the default mapping files for a file system but will generate a new set for use in the browsing operation.)

-r *map_dir*

This will map the specified directory for retrieve processing, analogous to what is done by an **fsretrieve -R**, and will always generate new mapping files. (It will not affect the default mapping files for a file system but will generate a new set for use in the browsing operation.)

mount_point

The file system to operate on.

-a Report all file systems when **list** or **mlist** is requested and not just StorNext file systems.

list Report on all file systems, not just mounted file systems.

mlist Report only mounted file systems.

SEE ALSO

fs_mapper(1), **fspolicy**(1)

NAME

mmportinfo − Queries for import/export port information about an archive

SYNOPSIS

mmportinfo [**-h**] *archiveName*

DESCRIPTION

The **mmportinfo**(l) command is issued from the command line to in order to obtain import/export port information about an archive. This information consists of the following colon separated values:

- hardware name
- list of media types
- portID

The *portID* value is required by the **vsarchiveenter**(l) and **vsarchiveeject**(l) commands when moving media in and out of an archive. There is no import/export port information for vaults so this command will return nothing if a vault is specified.

OPTIONS

archiveName

Identifies the archive to obtain port information about.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-h Requests help for the entered command.

The Help option returns the usage for the entered command.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

EXIT STATUS

0 The **mmportinfo**(l) command completed successfully

1 An error is detected by the Media Manager software.

EXAMPLES

Request port information for a scsi archive

mmportinfo archive1

Output returned:

```
16:LTOW,LTO:0,0,15,16
```

In this example the hardware name is 16, the media type list is LTOW,LTO and the portID is 0,0,15,16

Request port information for an acsls archive

mmportinfo archive2

Output returned:

```
0,0,0:LTOW,LTO:0,0,15,1
```

In this example the hardware name is 0,0,0, the media type list is LTOW,LTO and the portID is 0,0,15,1

NOTES

None.

SEE ALSO

vsarchiveeject(l), **vsarchiveenter**(l)

NAME

objimport.json − Object Import File

DESCRIPTION

In some cases it may be desirable to modify the list of objects to be imported, and/or the attributes of some or all of those objects, prior to the import. This can be achieved as follows:

1. Use the fsobjimport command to generate a manifest file for the object import operation. Refer to Example 4 of the fsobjimport man page for guidance on how to do this.
2. Edit the manifest file to make the desired changes.
3. Ingest the objects specified in the modified manifest file, by running the fsobjimport command, this time specifying the **-f** *manifest-file* and either **-t** *files* or **-t** *media* options.

SYNTAX

The object import file should specify the media ID of the storage bucket along with a list of the objects to be imported and their attributes. The JSON format for this list is as follows:

```
{
    "media": AWS000,
    "file_list": [
        {
            "objectId": "0002839C-DC81-4900-AA35-13ECC7C63C9D",
            "size": 100,
            "time": 1519956672,
            "mode": 420,
            "gid": 0,
            "uid": 0,
            "destPath": ""
        }
    ]
}
```

media Specifies the ID of the media containing the objects to import. The media must be known to the Tertiary Manager system.

objectId
ID of the object on the object storage system.

size Size in bytes of the object.

time Data and time the object was last modified. Time is the number of seconds since the Epoch.

mode The file mode to set on the imported object where *mode* is a numeric representation of the mode bits.

gid The group identity to set on the imported object.

uid The user identity to set on the imported object.

destPath
The destination path for the imported object, relative to the destination directory specified with the **-d** option of the fsobjimport command.

SEE ALSO

fsobjimport(1)

NAME

showc − Show candidate information

SYNOPSIS

showc [**-l**|**-p**] [**-sntrTCD**] [<*mnt_pt*>...]

showc [**-l**|**-p**] [**-sx**] [**-f** < *file_name*>...]

showc -a [**-l**|**-p**] [-s] [<*mnt_pt*>...]

showc -E <time_delta> [**-l**|**-p**] [-s] [<*mnt_pt*>...]

showc -T [<*mnt_pt*>...]

showc -h

DESCRIPTION

This will report the count of store candidates for each specified file system or all file systems if none are specified. It can also report the number of truncation (**-t**), relocation (**-r**), alternate store (**-a**) or expiration (**-E**) candidates. Instead of reporting counts, file handles (**-l**) or complete path names (**-p**) can be displayed.

By default, the reported information is based on the contents of the database. This is where candidate entries eventually end up. Prior to being put in the database, each candidate is placed in a temporary file. The **-C** and **-D** options can be used to generate reports based on the contents of these files. Additionally, the **-f** option can be used to report information based on a specific event or candidate file.

For the truncate option (**-t**), when reporting the count, for performance reasons, the count may be an estimate if there are more than 1 million candidates. The value used for the estimate threshold can be changed by setting the SHOWC_TRUNC_ESTIMATE sysparm value. The larger the value, the longer it may take to determine the number of candidates.

OPTIONS

-l List file-handle information for candidate entries.

-p List complete path for candidate entries. This can be time-consuming.

-s This will perform a stat system call on each candidate to verify that it still exists on disk.

-n Report based on new candidate files

-t Generates a report of truncation candidates.

-r Generates a report of relocation candidates. (When used with the **-U** option, resets the progress of the upgrade processing. See UPGRADE PROCESSING below.)

-a Generates a report of alternate store candidates.

-E <time_delta>
Generates a report of expiration candidates based on expire time delta. Time delta format <integer>[m|h|d]. Default is minutes.

-T Report oldest and newest store candidate times. This is based on the create event files and the database.

-C Report based on unprocessed (directory) create event files.

-D Report based on unprocessed destroy event files.

-x Treats event files specified with **-f** as destroy event files.

-f *file_name*...
Report based on the specified candidate/event file(s).

-h Display the basic help / usage information for the command.

PATH ERRORS

When reporting candidate counts, or optionally the handles of candidates when using the "**-l**" option, this command just reports on the contents of the indicated candidate table or file. When the "**-p**" option is used the path names must be generated from the candidate information. If the parent and file handles are both

available they will be used to generate the path. If the parent handle is not available, or there was an error using the parent, then only the file handle will be used for the generation.

The path generation process can fail with different processing errors. If the path generation succeeds for a candidate the class index and path are reported as follows:

```
class - 3 path - /stornext/snfs1/class1/sub1/file.01
```

If the path generation fails the class index is still displayed along with an unknown path indicator and handle information for the candidate:

```
class - 3 path - (unknown): pino - 82 pgen - 0 ino - 71 gen - 2 name - file.01 (g
```

If the parent handle is not available only the handle info is provided:

```
class - 3 path - (unknown): ino - 71 gen - 2 name - file.01 (gone)
```

Note that the text in the parentheses at the end of the failure message is the reason for the generation failure. The current generation failures are:

```
(gone) - the file has been removed
(path too long) - the path is longer than the expected system maximum
(path generation error) - unexpected generation failure
```

One final note is that the candidate name is not always known. In the case of a generation failure where the name is not available "NA" will be reported for the name.

WARNINGS

This command is expensive to run on large candidate sets when showing info beyond just the count of the candidates.

Because of the expense, and potential effect on other running processes, it is recommended that this utility is ONLY used for reporting candidate info if a problem is suspected with existing candidates or candidate processing. It should NOT be used as a continuous monitoring tool.

Note that this command can have an especially adverse effect on the snbackup process. While backups are running it is ok to run the command to check candidate counts, but do not use options for reporting further candidate information.

EXIT STATUS

O for success. Anything else is a failure.

NAME

showc − Show candidate information and provide interface to mdarchive

SYNOPSIS

showc [**-l**|**-p**] [**-sntrTCD**] [<*mnt_pt*>...]

showc [**-l**|**-p**] [**-sx**] [**-f** < *file_name*>...]

showc -a [**-l**|**-p**] [-s] [<*mnt_pt*>...]

showc -E <time_delta> [**-l**|**-p**] [-s] [<*mnt_pt*>...]

showc -T [<*mnt_pt*>...]

showc -h [full]

showc [**-eIRS**] [**-l**[**-v**][**-T**]|**-L**] **-d** <*fsname*> <*field_name field_value*>...

showc -H <fsname> <file_key> [<file_key> ...]

showc -A <tiername> ...

showc -d <fsname> -U<**chk**|**set**|**clr**> [**-r**][**-m**]

DESCRIPTION

This will report the count of store candidates for each specified file system or all file systems if none are specified. It can also report the number of truncation (**-t**), relocation (**-r**), alternate store (**-a**) or expiration (**-E**) candidates. Instead of reporting counts, file handles (**-l**) or complete path names (**-p**) can be displayed.

By default, the reported information is based on the contents of the database. This is where candidate entries eventually end up. Prior to being put in the database, each candidate is placed in a temporary file. The **-C** and **-D** options can be used to generate reports based on the contents of these files. Additionally, the **-f** option can be used to report information based on a specific event or candidate file.

For the truncate option (**-t**), when reporting the count, for performance reasons, the count may be an estimate if there are more than 1 million candidates. The value used for the estimate threshold can be changed by setting the SHOWC_TRUNC_ESTIMATE sysparm value. The larger the value, the longer it may take to determine the number of candidates.

This utility can also be used for accessing information in the mdarchive. While not a complete set various capabilities are provided for querying the mdarchive and displaying contents.

OPTIONS

-l List file-handle information for candidate entries.

-p List complete path for candidate entries. This can be time-consuming.

-s This will perform a stat system call on each candidate to verify that it still exists on disk.

-n Report based on new candidate files

-t Generates a report of truncation candidates.

-r Generates a report of relocation candidates. (When used with the **-U** option, resets the progress of the upgrade processing. See UPGRADE PROCESSING below.)

-a Generates a report of alternate store candidates.

-E <time_delta>
Generates a report of expiration candidates based on expire time delta. Time delta format <integer>[m|h|d]. Default is minutes.

-T Report oldest and newest store candidate times. This is based on the create event files and the database.

-C Report based on unprocessed (directory) create event files.

-D Report based on unprocessed destroy event files.

-x Treats event files specified with **-f** as destroy event files.

-f *file_name...*
Report based on the specified candidate/event file(s).

-h [full]
Display the basic help / usage information for the command. If the 'full' arg is provided with the "**-h**" then this will cause the help / usage information for the mdarchive functionality to be displayed along with the basic usage for the command. (Full usage is also displayed with the "**-h**" if any of the mdarchive specific options have been specified.)

MDARCHIVE SPECIFIC OPTIONS

-d <fsname>
Perform a query of the mdarchive for the indicated file system.

-e Query the ext table for the extended attributes and external representations of files. The default is to query the file table for basic file information.

-R Use the REST interface to the FSM for submitting the queries rather than issuing a 'raw' query to the mdarchive. See the "**REST QUERIES**" section below for more information.

-S <*respType*>
Set the response type for the queries (what fields are to be returned). The larger the query response the slower the query. For queries where many hits are expected it would be best to use the smallest size which provides the needed info. The choices (from fewest returned fields to all) are:

basics The handle info and file times.

keys The metadb key fields. (Fields in the main file table that do not require a second query for the complete inode info.)

full Return all but the debug fields.

all Return all fields. (default value for command)

-I Interactive mode. Can be useful for entering multiple queries.

-l List the entries found and some basic info on those entries. The default is just to show counts.

-v When used while listing will show additional information.

-T When used while listing will show additional time information.

-L List the entries found in a long raw format. (Similar to the output from the cvfsdb queries.)

-H <fsname> <file_key> [<file_key> ...]
Query the indicated file system and get the file handle(s) for the provided file key(s).

-A <tiername>
Convert the provided tier name(s) to the internal mdarchive format. Needed if you want the tier/affinity name that is to be specified as part of a query.

-U <chk | set | clr>
Check, set or clear the file system indicator on whether or not upgrade processing for the file system is complete. This option should only be used under the guidance of Quantum technical assistance. (More info below.)

-m When setting or clearing the upgrade flag on a file system send a message to the Tertiary Manager software.

PATH ERRORS

When reporting candidate counts, or optionally the handles of candidates when using the "**-l**" option, this command just reports on the contents of the indicated candidate table or file. When the "**-p**" option is used the path names must be generated from the candidate information. If the parent and file handles are both available they will be used to generate the path. If the parent handle is not available, or there was an error using the parent, then only the file handle will be used for the generation.

The path generation process can fail with different processing errors. If the path generation succeeds for a candidate the class index and path are reported as follows:

```
class - 3 path - /stornext/snfs1/class1/sub1/file.01
```

If the path generation fails the class index is still displayed along with an unknown path indicator and handle information for the candidate:

```
class - 3 path - (unknown): pino - 82 pgen - 0 ino - 71 gen - 2 name - file.01 (g
```

If the parent handle is not available only the handle info is provided:

```
class - 3 path - (unknown): ino - 71 gen - 2 name - file.01 (gone)
```

Note that the text in the parentheses at the end of the failure message is the reason for the generation failure. The current generation failures are:

```
(gone) - the file has been removed
(path too long) - the path is longer than the expected system maximum
(path generation error) - unexpected generation failure
```

One final note is that the candidate name is not always known. In the case of a generation failure where the name is not available "NA" will be reported for the name.

MDARCHIVE USAGE

NOTE:

The functionality for querying the MDARCHIVE is provided primarily as a tool for support personnel. It should not be used on any regular basis without first checking with Quantum technical assistance to verify its use.

GENERAL INFORMATION:

This command can be used for reporting the contents of the mdarchive. The mdarchive is a basic database containing information on the metadata of a file system. Note that querying the mdarchive can be expensive and should not be done too often as it may affect system performance. Within the mdarchive there are two tables: the ext table and the file table. The ext table contains the extended attributes and external representations of files. The file table contains the basic file information. (Typically the Tertiary Manager software is interested in the file table information.)

The interface for making queries is based on the interface used in the cvfsdb utility. An individual query will be made up of one or more mdarchive fields and the values to use. Here is a list of valid fields which can be provided to a query. Note that unless otherwise noted the values for the field can be an individual integer value or an integer range (begin and end values).

inode_id
: This is a unique identifier of a file, defined and used internally in the file system.

snea_key
: This is a unique identifier of a file, defined and used by the Tertiary Manager software. (Also referred to as the file key.)

inode_num
: This is the inode number of a file.

parent The inode_id of the parent directory. This can be used to find the immediate children of a directory. (If a recursive search is needed use the farey values below.)

last_tid The last transaction id associated with the file. File system activities are assigned transaction ids as they occur and can be used for queries. (Typically a value of ’current’ is provided for this field to indicate only current files are of interest.)

ftype The type of file to query for. A value of 1 indicates directories and a value of 2 indicates files.

mtime The file modification time.

atime The file access time.

blocks The number of blocks allocated for a file.

affinity The name of the affinity where the file is located. (Note that like the other fields an integer value is expected here. The **-A** option can be used to convert a name to a hex value which can then be provided in a query.)

ea_class
The class index corresponding to the policy class of the file or directory.

smfile_state
This is a value that indicates the current state of a file within a managed file system. The valid state values are:

```
1 - file has not completed add relation processing
2 - file does not yet have the key value set
3 - this is a managed directory
4 - file has a key but no copies stored
5 - file has some but not all copies stored
6 - all copies stored for file but not truncated
7 - all copies stored for file and it is truncated
8 - not all copies have been stored but the file is truncated
```

When using this field it is best used with the other fields making up the indexes mentioned below.

name The name of a file or directory. The name field is not indexed and in general should not be used for queries unless combined with the parent to find a specific file in a directory. Note that there are no wildcards and you can only query for specific names.

Farey Values. A farey value is a rational value between 0 and 1. Every file in a file system is assigned a farey value. Every directory is assigned two values, a start and an end. All children of a directory, files and directories, will have farey values between the start and end values of the parent. This is true even for recursive descendants of that directory. When specifying a farey value for a query it will require two integer values, a numerator and a denominator in that order, whose quotient is the desired rational number.

See the note below which indicates that the farey values are part of some indexed queries. If for a query there is no need to limit to one directory then use this for 'all' contents: farey_start 0 1 1 1

NOTE: if the desire is to determine immediate children only use the parent field.

farey_start
A rational number for the starting farey value of a directory.

farey_end
A rational number for the ending farey value of a directory.

OTHER QUERY OPTIONS:

Following are other options/instructions that can be added to a query to adjust what is returned.

limit Limit the number of returned rows to the number provided. (A single integer value.)

ascending
If it is desired to have the output sorted on a field, provide the ascending key word with the field name. For example: "mtime ascending". It is also allowed to shorten this keyword to "asc".

descending
Another sort option is to go in descending order on a field. Provide the descending key word with the field name. For example: "atime descending". It is also allowed to shorten this key-

word to "desc".

current The current keyword is useful for entering the maximum int value. For example if you want a query where all mod time values are returned the syntax would be: "mtime 0 current". This keyword can be used on non-time fields as well: "blocks 1 current". It is also allowed to shorten this keyword to "curr".

MDARCHIVE TABLE INDEXES:

The mdarchive tables have indexes set on them to make common queries operate more quickly. While you can query on any of the fields mentioned above it is STRONGLY recommended that you make only indexed queries.

In general for the mdarchive queries, the more fields that can be provided for a query the more efficient the query will be. That is true whether the fields are part of the index or not. One field that is always a good idea to provide is the *last_tid*. (usage: last_tid curr) This indicates you are only interested in active files or directories.

The one field that can be used without others and still operate efficiently is the *inode_id*. This field is the primary index for both the file and ext tables. When it is the only field provided this command will actually query both the file and ext tables and report the results. (When the REST interface is used the ext table contents are not shown. See the REST QUERIES section below for more info.) Note that the inode_id is the only index on the ext table.

Here are the indexes available for use. The index name shown here is not to be provided to the command, it is just being used to describe the index. It is optimal to provide all fields when using an index but it is not required by this command.

Children
This index is made up of the *parent* and *last_tid* fields. It is used to get the immediate children of a directory.

File_key
This index is made up of the *snea_key* and *last_tid* fields. It is used to get the file or directory with the provided file key.

Inode_number
This index is made up of the *inode_num* and *last_tid* fields. It is used to get the file or directory with the provided inode number.

Store_candidates
This index is made up of the following fields: *mtime*, *ftype*, *smfile_state*, *ea_class*, *last_tid* and *farey_start*. It is used internally to find files that have not been stored.

Trunc_candidates
This index is made up of the following fields: *atime*, *ftype*, *smfile_state*, *ea_class*, *blocks*, *affinity*, *last_tid* and *farey_start*. It is used internally to find files that have been stored and are ready for truncation.

MDARCHIVE QUERY EXAMPLES:

Here are some examples of useful queries.

Get the base information on a file residing on a file system with name snfs1 given a file key (7).

```
showc -ld snfs1 snea_key 7 last_tid current
```

Get extended information on a file residing on snfs1 given an inode id (0x16).

```
showc -lvd snfs1 inode_id 0x16 last_tid curr
```

Get extended information on a file residing on snfs1 given an inode number (192).

```
showc -lvTd snfs1 inode_num 192 last_tid curr
```

Get the base information on files residing on snfs1 under parent directory with inode_id 0x21. Sort by the key value ascending.

showc -ld snfs1 snea_key asc parent 0x21 last_tid curr

Get a count of store candidates on snfs1 that reside in the class with index 2.

showc -d snfs1 mtime asc mtime 0 curr ftype 2 smfile_state 2 5 ea_class 2 last_tid curr farey_start 0 1 1 1

UPGRADE PROCESSING

As mentioned above the **-U** option can be used to check, set or clear the upgrade flag for a file system. If a file system existed in a version of StorNext prior to the existence of the mdarchive then there is some work needed to upgrade inodes for use by the Tertiary Manager software. This processing is done behind the scenes while normal functionality proceeds.

As the upgrade processing proceeds, progress within the file system is tracked. This is so if the software is restarted, etc., the processing can take up where it left off. When the **-r** option is provided with **-U clr**, in addition to clearing the upgrade flag for the file system, the upgrade progress is reset causing the whole file system to have to be reprocessed. Note that the **-r** option is only allowed when clearing the upgrade flag.

If the **-U** option is used to set or clear the upgrade flag then by default that change will not be recognized until the next time the Tertiary Manager software is cycled. By using the **-m** with the **-U** option a message is sent so that cycling the software is not required.

CAUTION: There should never be a reason for a site to have to manipulate, or even check, the upgrade flag for a file system. Again, these options are provided as a tool for support personnel.

REST QUERIES

It should be noted that there are some differences in behavior when using the REST interface over the default raw interface. The first of the differences is when file system activity will show up in a query. Typically a file system action (such as truncating a file) will show up via a REST query before it shows up in a raw query. This is because the FSM cache is usually updated first before results are reflected in the mdarchive.

Another difference is what fields can be used in a query with the REST interface. The REST interface only provides a set of "canned" queries that are typically used within SNSM operations. (Some examples: get files that need storing, get files that are not yet truncated, get the info on a current file with the specified key.) If 'extra' fields are provided to a REST query the showc command does not give any error/warning, but just ignores the extraneous fields.

For example, a REST query given a key value will only look at the key and last_tid values. So these two queries will both return the file with a key of 9 regardless of whether or not the file has any blocks:

```
showc -lRd snfs1 snea_key 9 blocks 0 last_tid curr
showc -lRd snfs1 snea_key 9 blocks 1 curr last_tid curr
```

For the same queries using the raw interface, only the query specifying the correct block values will return the file with the key of 9.

```
showc -ld snfs1 snea_key 9 blocks 0 last_tid curr
showc -ld snfs1 snea_key 9 blocks 1 curr last_tid curr
```

REST query operations also differ from raw query operations when using the inode_id as a query parameter. There is no access to the ext tables with REST interface. So when an inode_id is provided, and REST is in use, only info from the file table will be shown.

WARNINGS

This command is expensive to run on large candidate sets when showing info beyond just the count of the candidates.

Because of the expense, and potential effect on other running processes, it is recommended that this utility is ONLY used for reporting candidate info if a problem is suspected with existing candidates or candidate processing. It should NOT be used as a continuous monitoring tool.

Note that this command can have an especially adverse effect on the snbackup process. While backups are running it is ok to run the command to check candidate counts, but do not use options for reporting further

candidate information.

EXIT STATUS

O for success. Anything else is a failure.

NAME

showsysparm − Report the value for the specified Tertiary Manager system parameters.

SYNOPSIS

showsysparm [**-F** *type*] *sysparm...*

DESCRIPTION

The **showsysparm**(1) command takes a Tertiary Manager system parameter and will report the associated value for that item.

OPTIONS

sysparm

The Tertiary Manager system parameter to report the value of.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

REPORT STATUS

The resulting output will be of format <sysparm>=<value>

EXAMPLE

```
% showsysparm DEFAULT_MEDIA_TYPE CLASS_DRIVEPOOL BACKUPFS BOGUS_SYSPARM
DEFAULT_MEDIA_TYPE=LTO
CLASS_DRIVEPOOL=fs_F0drivepool
BACKUPFS=/stornext/snfs1
```

EXIT STATUS

0 Command found and reported the values of the sysparms specified.

n Command could not find n number of sysparms specified.

NAME

smproject − Manage Tertiary Manager projects

SYNOPSIS

smproject -a -n *project* [*dirs...*]

smproject -d -n *project*

smproject -m -n *project* **-r** *newproject*

smproject (**-p**|**-u**) **-n** *project dirs...*

smproject [**-l**] [**-n** *project*]

smproject -q *files...*

DESCRIPTION

The **smproject**(1) command is the administrative tool for creating, reporting, changing, and deleting Tertiary Manager projects. Projects are used for tracking tertiary-storage usage and to enforce quota limits. They are named collections of directories and sub directories, repetitively, to the bottom of their directory hierarchies. When tertiary copies of files are created, they are associated with projects by their directory paths. When a file is moved, its associated project does not change until new copy versions are created as the file is stored or the directory is unassigned from the project.

OPTIONS

-a
: Add a new project.

-d
: Delete a project.

-l
: List the directories comprising a project. Only the directories that contain files are listed.

-m
: Modify a project.

-n *project*
: Operate on the specified project.

-p
: Assign directories to a project.

-q
: Query the project assignment for the specified file(s).

-r *newname*
: Change a project's name to *newname*.

-u
: Unassign directories from a project.

SEE ALSO

smusage(1), **smquota**(1), **fsrmversions**(1)

NAME

smquota − Administer Tertiary Manager quotas and report on quota status

SYNOPSIS

smquota -a -q *quotaname* (**-u** [*user*] | **-g** [*group*] | **-p** [*project*])
[**-m** *mediaid*] [**-t** *mediatype*] [**-f** *fs*] [**-s** *softlimit*] [**-h** *hardlimit*] [**-T** *grace*]

smquota -c -q *quotaname* [**-u** [*user*] | **-g** [*group*] | **-p** [*project*]]
[**-m** *mediaid*] [**-t** *mediatype*] [**-f** *fs*] [**-s** *softlimit*] [**-h** *hardlimit*] [**-T** *grace*] [**-n** *newname*]

smquota -d -q *quotaname*

smquota -l [**-q** *quotaname*] [**-u** [*user*] | **-g** [*group*] | **-p** [*project*]] [**-m** *mediaid*] [**-t** *mediatype*] [**-f** *fs*] [**-H**]
[**-e**]

smquota -r [**-x**] [**-u** [*user*] | **-g** [*group*] | **-p** [*project*]] [**-m** *mediaid*] [**-t** *mediatype*] [**-f** *fs*] [**-H**]

smquota -k *files...*

DESCRIPTION

The **smquota**(1) command is the administrative tool for the Tertiary Manager quota feature. This command allows quota definitions to be added, modified, deleted, listed, reported, and updated. Tertiary Manager quotas can place *hard* and *soft* limits per quota definition on the amount of tertiary storage a consumer may have. Consumers are users, groups, and projects.

A consumer that has less storage than the soft limit has the status: **good**. A consumer that has more storage than the soft limit and less storage than the hard limit has the status: **grace**. A consumer that has more storage than the soft limit for longer than the grace period has the status: **exceeded**. A consumer that has storage at or above the hard limit has the status: **exceeded**. When the hard limit is reached, further store operations for that consumer's quota are blocked. When the soft limit is reached, store operations continue to succeed until the soft limit has been exceeded for longer than a quota-defined grace period.

Quotas can be defined for specified resource criteria and all consumers, or for specified users, groups, or projects. Primary disk storage and storage on vaulted media are not included in the limits.

Quota limits can be applied to the following resources: *mediaid* (not for individual tape media, and not for hard limits or soft limits with grace periods), *mediatype*, and *File system*.

Note: Changing a file's owner, group, or project and then storing that file causes all previously stored versions to be associated with the new consumer.

OPTIONS

-a

Adds (defines) a new quota.

-c

Changes an existing quota definition.

-d

Deletes a quota definition, directory associations, and related quota statistics.

-e

When listing quota definitions for specified consumers and resource criteria, do not display the <any> wildcard value.

-f *fs*

Associates the named *fs* file-system resource with the quota definition.

-g

Operates on all groups.

-g *group*

Associates the named *group* consumer with the quota definition, or operates on the specified group.

-H

Prints full-precision byte counts and seconds values in reports.

-h *hardlimit*

Sets the hard limit for the quota definition. The *hardlimit* value is in bytes specified as an integer with an optional multiplier suffix. See below for the set of BYTE SUFFIXES.

-k

Reports on the status of quotas that apply to the specified file(s).

-l

Lists quota definitions.

-m *mediaid*

Associates the named *mediaid* resource with the quota definition. This is only for non-tape media, and cannot be used with hard limits or soft limits with grace periods.

-n *newname*

Renames an existing quota definition to **newname**.

-p

Operates on all projects.

-p *project*

Associates the named *project* consumer with the quota definition, or operates on the specified project.

-q *quotaname*

Specifies the *quota* name.

-r

Produces a report from the quota-status table. When used with **-x**, updates the table.

-s *softlimit*

Sets the soft limit. The *softlimit* value is in bytes specified as an integer with an optional multiplier suffix. See below for the set of BYTE SUFFIXES.

-T *grace*

Specifies the grace time period for the soft limit. The time value is specified with a numeric value followed by an optional single character suffix indicating the units. The time value can be set in units of minutes, hours, days, weeks, months or years using the following suffix values:

s - seconds (default if no suffix specified)
m - minutes
h - hours
d - days
w - weeks
M - months (30 days)
y - years (365 days)

-t *mediatype*

Associates the named *mediatype* media-type resource with the quota definition. The allowed media-type values are:

AWS
AZURE
LATTUS
GOOGLE
GOOGLES3
S3
S3COMPAT
LTO
LTOW

3592
T10K
SDISK

-u

Operates on all users.

-u *user*

Associates the named *user* consumer with the quota definition, or operates on the specified user.

-x

Updates the quota status table.

BYTE SUFFIXES

Byte-count limits are specified as an integer followed by a multiplier suffix. The set of suffixes and their multiplier values is as follows:

B	bytes
KB	kilobytes (1000)
KiB	kibibytes (1024)
MB	megabytes (1000^2)
MiB	mebibytes (1024^2)
GB	gigabytes (1000^3)
GiB	gibibytes (1024^3)
TB	terabytes (1000^4)
TiB	tebibytes (1024^4)
PB	petabytes (1000^5)
PiB	pebibytes (1024^5)

SEE ALSO

fsrmversions(1), **fsschedule**(1), **fsschedlock**(1), **smproject**(1), **smusage**(1)

NAME

smusage − Report Tertiary Manager resource usage and manage the usage tracking feature

SYNOPSIS

smusage [**-u** [*user*] | **-g** [*group*] | **-p** [*project*]]
[**-a** | [**-m** [*media_id*]] [**-t** [*media_type*]]] [**-f** [*fs*]] [**-H**]

smusage (**-d** [*directory*] | **-D** *directory)* [**-s**] [**-u** [*user*]] [**-g** [*group*]] [**-p** [*project*]]
[**-a** | [**-m** [*media_id*]] [**-t** [*media_type*]]] [**-f** [*fs*]] [**-H**]

smusage (**-c**|**-e**|**-i**|**-r**|**-x**) [**-f** *fs*] [**-y**]

DESCRIPTION

The **smusage**(1) command reports Tertiary Manager usage tracking information that is stored in database tables. Control of the usage tracking in the tables is also done with this command. To use this feature the **USAGE_TRACKING** system parameter needs to be *true* (on) and usage tracking needs to be enabled with the **smusage**(1) command for the managed file systems.

There are four kinds of tables (numbered): user (1), group (2), project (3), and file directory path (4).

The *user*, *group*, and *project* tables each track usage by the ID associated with their table type and by media ID, media type, and file system. The *directory-path* table tracks usage by media ID, media type, and file system, and also by user ID, group ID, and project ID. Usage is further divided into storage for active copies and inactive copies. Active copies are for the current version of each file.

The **smusage**(1) command reports the contents of the tables in summary or detail according to the selected options.

When run with no options, the **smusage**(1) command reports the contents of the user, group, and project tables with consolidated totals for active copies and for all copies by user ID, group ID, and project ID, respectively. The *media_id*, *media_type*, and *file system* fields of these tables can be selected to print in the report to separate the usage counts into the amounts associated with each of these fields.

The directory-path report is requested by specifying the **-d** option. As with the other tables, the directory-path report can include the *media_id*, *media_type*, and *file system* fields, and also fields for *user*, *group*, and *project*.

When the **smusage**(1) command is used for controlling the usage tables for one or all managed file systems, it can: enable and disable usage tracking, delete and initialize data in the usage tables, and check the status of usage tracking.

OPTIONS

-a

Add fields to each report row for media ID, media type, and file system. This option is applicable to all four table types.

-c

Check (report) the usage tracking in effect by file system. The **-f** option may be used to limit the report.

-d [*directory*]

Print the directory-path report. When the *directory* option argument is specified, it must be a full path beginning at the root of the file name space. All the directory paths in the table having the option-argument path as a prefix are reported unless the **-s** option is specified. When the *directory* option argument is not specified, the **-f** option must be specified with an option argument.

The report provides: the directory path(s) where files having tracked copies reside, the total bytes used for copies, and the bytes used for active copies. The report also includes the file system if the **-f** option is specified. The report can have additional fields as they are requested with the following options: **-a**, **-g**, **-m**, **-p**, **-t**, and **-u**.

-D *directory*

Print the directory-path report for the files in the specified *directory* without descending into its sub directories. Refer to the **-d** option for further details about the directory report.

-e

Enable usage tracking for all file systems or the **-f** *file_system* option can be used to enable a single file system. This operation deletes the current usage data, enables usage tracking, and initializes usage data. This can take a long time to complete after the command has ended. Stopping the TSM subsystem while using this option is recommended to avoid a possibility of incorrect usage tracking. A confirmation-request prompt is printed unless the **-y** option is specified.

-f [*fs*]

Print file-system information in reports. When the option argument is used, constrain the report to the file system specified by file-system mount-point name. This can also be used to constrain the processing done by the **c**, **e**, **i**, **r**, or **x** options. The option without an argument can be used to add the file-system field to user, group, and project reports. The option with an argument can be used to specify the file system for directory-path reports (**-d**), which also adds the file-system field to the report.

-g [*group*]

Print group information in reports. When the option argument is used, constrain the report to the group specified by name or numeric value. When used with the **-d** option, add the group field. When used without the **-d** option, print the group table. In the latter case, the **-g** option cannot be used with either the **-u** or **-p** option. The group names and numbers can be found in the */etc/group* file.

-H

Display full-precision byte counts.

-h

Display command-line usage information.

-i

Request initialization of usage data in all four tables for all file systems. The **-f** option may be specified with an option argument to limit initialization to the specified file system. This operation does not delete usage data. Use of the **-e** option is preferable to **-i** because it deletes existing usage-tracking data before initializing. A confirmation-request prompt is printed unless the **-y** option is specified. At a later time, the request will be fulfilled. This operation can take a long time to complete after the command has ended.

-m [*media_id*]

Add the *media_id* field to each report row. Constrain the report to the specified *media_id* if the option-argument is also provided. This option is applicable to all four table types.

-p [*project*]

Print project information in reports. When the option argument is used, constrain the report to the project specified by project name or numeric value. When used with the **-d** option, add the project field. When used without the **-d** option, print the project table. In the latter case, the **-d** option cannot be used with either the **-g** or **-u** option. The project names and index numbers can be listed with the *smproject* command.

-r

Disable usage tracking for all file systems. The **-f** option may be used to limit the effect to a specified file system. A confirmation-request prompt is printed unless the **-y** option is specified.

-s

Specify a summary directory-path report. The summary byte counts apply to all of the directories at and below the single directory named by the argument to the **-d** option.

-t [*media_type*]
Specify adding the *media type* field to each report row and limit the report to the specified *media type* if the option-argument is also provided. This option is applicable to all four table types.

-u [*user*]
Print user information in reports. When the option argument is used, constrain the report to the user specified by name or numeric value. When used with the **-d** option, add the user field. When used without the **-d** option, print the user table. In the latter case, the **-u** option cannot be used with either the **-g** or **-p** option. The user names and numbers can be found in the */etc/passwd* file.

-x
Delete usage data in all four tables for all file systems. This option does not change the status of usage tracking per file system. The **-f** option may be specified with an option argument to limit deletion to the specified file system. A confirmation-request prompt is printed unless the **-y** option is specified.

-y Suppress the confirmation-request prompt and use *y* as the response.

EXAMPLES

Reports

Generate a report of consumed storage for all users, groups, and projects:

```
smusage
```

List all users, groups, and projects by resources:

```
smusage -a
```

List users with a media-type field:

```
smusage -u -t
```

Report usage by users of LTO media:

```
smusage -u -t lto
```

Management

Check to see if usage tracking is enabled on *managed1* file system:

```
smusage -c -f /stornext/managed1
```

SYSPARM

The **USAGE_TRACKING** parameter is used to enable or disable the usage tracking feature. When USAGE_TRACKING is *false* (off), usage tracking will be disabled for all new managed file systems and will also disable the enforcement of Tertiary Manager quotas. When USAGE_TRACKING is *true* (on), usage tracking will be enabled for all new managed file systems. The **smusage**(1) command will need to be run to enable or disable usage tracking on existing file systems. The **USAGE_TRACKING** system parameter is *false* (off) by default and can be overridden in *fs_sysparm*. See *fs_sysparm.README* file for details.

To enable usage tracking set the **USAGE_TRACKING** parameter to *true* then enable each existing file system by running the following commands:

```
TSM_control stop
smusage -e [ -f <mountpoint> ]
TSM_control start
```

To disable usage tracking for all file systems set the **USAGE_TRACKING** parameter to *false* then disable each file system by running the following commands:

```
TSM_control stop
```

```
smusage -r [ -f <mountpoint> ]
TSM_control start
```

To remove the usage counters in the usage tracking tables run the following commands:

```
TSM_control stop
smusage -r [ -f <mountpoint> ]
smusage -x [ -f <mountpoint> ]
TSM_control start
```

FILES

/usr/adic/TSM/config/fs_sysparm.README

SEE ALSO

smproject(1), **smquota**(1), **fsrmversions**(1)

NAME

sm_diva_media_cleanup − Remove DIVA metadata from the Storage Manager database

SYNOPSIS

sm_diva_media_cleanup -a *archive_name* **-d** *DIVA_MySql_database_name* [**-M** *map_file*] [**-v** *vault_name*] **-T**

DESCRIPTION

Remove DIVA metadata from the Storage Manager database. This is useful in testing and restarting the importing of DIVA metadata. This program must be run as the *root* user.

OPTIONS

-a *DIVA_Archive*

Names the media archive containing the online DIVA media that have been imported.

-d *DIVA_MySql_database_name*

Names the MySql database having tables of the DIVA metadata that has been imported.

-M *map_file*

Names an optional CSV file of mappings from media type name in the DIVA database to the corresponding media name and media class name in the Storage Manager database. When this file is not specified, default mapping information is used.

-v *vault_name*

Names an optional media vault having the offline DIVA media that have been imported. When this name is not specified, the name *vault* is used.

-T Directs the program to clean DIVA metadata from the Storage Manager mediadir table, and calls *fsclean -r* to remove related metadata from other Storage Manager tables. This can add a lot of processing time.

EXAMPLES

sm_diva_media_cleanup -d diva_import -a diva_archive -v diva_vault -M mediamap.txt -T

Cleans all DIVA metadata from the Storage Manager database, which allows the importing of DIVA metadata to be restarted.

NOTES

Restarting the importing of DIVA metadata without cleaning the information from previous import operations will cause cross references to be incorrect in the Storage Manager database.

The command reports the number of media that were cleaned from the Storage Manager database.

SEE ALSO

diva_db_export(1), **diva_sm_dbload**(1), **fsimport_diva**(1), **fsclean**(1), **sm_diva_media_import**(1),

NAME

sm_diva_media_import − Import DIVA media into the Storage Manager database

SYNOPSIS

sm_diva_media_import -a *archive_name* **-d** *DIVA_MySql_database_name* **-g** *tg_file* **[**-h**] -m** *media_file* [**-M** *map_file*] **[**-p**] [**-t**] [**-T**]** [**-v** *vault_name*]

DESCRIPTION

This program references a DIVA database in MySql to bring DIVA media (data tapes) into the Storage Manager database. This program must be run as the *root* user.

OPTIONS

-a *DIVA_Archive*
: Names the StorNext archive containing the online DIVA media that are being imported.

-d *DIVA_MySql_database_name*
: Names the MySql database holding the tables of DIVA metadata.

-g *Tape_Group_File*
: Names an optional CSV file containing a list of tape groups to be imported. When unspecified, all DIVA media are imported.

-h
: Prints the command help message.

-M *map_file*
: Names an optional CSV file of mappings from media type name in the DIVA database to the corresponding media name and media class name in a Storage Manager database. When this file is not specified, default mapping information is used. Provide this mapping file when the internal mapping does not cover the DIVA media types or when importing specified media types. Each entry contains three fields: diva media type name, MSM media type name, and MSM media class name.

-p
: Prints the internal media mapping list.

-t
: Prints the type and the state count (online/offline) for each media type in the DIVA database.

-T
: Prints the name and the media count for each tape group in the DIVA database.

-v *vault_name*
: Names an optional media vault holding the offline DIVA media that are being imported. When this name is not specified, the name *vault* is used.

EXAMPLES

sm_diva_media_import -d diva_import -a diva_archive -v diva_vault -g tape_groups -M mediamap.txt
: Reference the *diva_import* MySql database, the *diva_archive* tape library, the *diva_vault* tape vault, and the *mediamap.txt* map file, to import all the media in the tape groups named in the *tape_group* file.

sm_diva_media_import -p
: Prints the internal media mapping list, which is:

```
LTO-800G,LTO,F0_LTO_DATA
LTO-1.5T,LTO,F0_LTO_DATA
LTO-2.5T,LTO,F0_LTO_DATA
T10000T2,T10K,F0_T10K_DATA
```

SEE ALSO

diva_sm_dbload(1), **diva_db_export**(1), **fsimport_diva**(1), **sm_diva_media_clean**(1)

NAME

snbackup − Execute backup of configuration, database, and file system metadata.

SYNOPSIS

snbackup [**-F** *type*] [**-h**] [**-p**] [**-s**]

DESCRIPTION

The **snbackup**(1) command generates a backup of StorNext software using SNSM which can later be used to restore the database, configuration files, and file system metadata. Each backup component (database, configuration, and file system metadata) is created in a compressed tar file. Each archive file is stored to media that was selected for use by the defined backup class.

A specific class is used for all backup activity. For SNSM, this class is named _adic_backup. Users can modify backup parameters for SNSM with the **fsmodclass**(1) command. Refer to **fsmodclass** man page for more information. Any modifications to the _adic_backup class class will take effect during the next invocation of the **snbackup**(1).

The **snbackup**(1) command can make a full backup or a partial backup. A full backup is a complete checkpoint of the database, a full snapshot of each file system's metadata, and a full snapshot of the system configuration files. A partial backup contains an incremental checkpoint of the database since the last backup, a full snapshot of each file system's metadata, and a full snapshot of the configuration. A request to create a partial backup is upgraded to a full backup if there is no recorded full backup.

snbackup(1) will generate manifest files that are used by **snrestore**(1). The manifest files are created in */usr/adic/TSM/internal/status_dir* with a redundant copy made to */usr/adic/mysql*. Each backup will produce a *snbackup_manifest* file that catalogs the backup files and associated media. A *device_manifest* file is created for storage disk or Object Storage media to facilitate access to these media types during **snrestore**(1). If Object Storage media is used for backups, then a third manifest file is generated called *snobjs_manifest* that will map files to Object Storage object IDs.

By default, the SNSM software is configured to run full backups once a week on Sunday, with a partial backup every day from Monday through Saturday.

Backup notifications will be done by email and RAS events. Emails will be sent to users configured to receive backup notifications and will contain status information as well as copies of the manifest files. These mail notifications should be retained in the event that the system needs to be recovered using **snrestore**(1). RAS events will be generated for failed backups.

REPORT STATUS

For the TEXT output format, the status report produces a full listing of the most recent backup or the currently running backup. For the JSON format, a more limited status report is generated:

State The current state of snbackup (JSON output only). Valid values: **idle, running, unknown**

Result The final results of the most recent run. Valid values: **completed, failed**

Result Details
: Additional information on the results of the most recent run. Valid values: **Backup Successful** or information specifying the type of failure that occurred

Start Time
: The date and time when the most recent run started

End Time
: The date and time when the most recent run ended

OPTIONS

-p Create partial backup

-s Get status for currently running backup or results of previous backup.

-h Print usage information.

-n DEPRECATED: This option is now ignored. It previously skipped creating gzip'd metadata-dump-file copies.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

ENVIRONMENT

The **BACKUPFS** setting in the *fs_sysparm* file will determine which file system **snbackup**(1) will try to use when run. If this setting is not defined, **snbackup**(1) will attempt to automatically chose a file system that has either been configured for SNSM and will then update the *fs_sysparm* file accordingly.

The **POST_UPGRADE_BACKUP** environment variable indicates the location of the **POST_UPGRADE_BACKUP** file, which determines if the next backup should be a full backup. The **POST_UPGRADE_BACKUP** file is created the first time the Tertiary Manager software is started following a StorNext upgrade. If the **POST_UPGRADE_BACKUP** file is present, then the backup will be forced to be a full backup. Upon completion of the backup, the **POST_UPGRADE_BACKUP** file will be deleted if it exists. The default location of **POST_UPGRADE_BACKUP** can be found at */usr/adic/TSM/internal/status_dir/post_upgrade_backup*.

FILES

/usr/adic/TSM/config/fs_sysparm.README
/usr/adic/TSM/internal/status_dir/snbackup_manifest
/usr/adic/TSM/internal/status_dir/device_manifest
/usr/adic/TSM/internal/status_dir/snobjs_manifest
/usr/adic/TSM/internal/status_dir/post_upgrade_backup
/usr/adic/mysql/snbackup_manifest
/usr/adic/mysql/device_manifest
/usr/adic/mysql/snobjs_manifest

SEE ALSO

snrestore(1), **snbkpreport**(1), **fsmodclass**(1),

NAME

snbkpreport − Display available SNSM backups

SYNOPSIS

snbkpreport [**-h**]

DESCRIPTION

The **snbkpreport**(1) command provides a listing of available SNSM software backups which can be used for a restore operation. Information will be listed in a tabular form. Full backups and all subsequent partial backups will be grouped together. Each copy will also be grouped together. Media required for each set of backups will be shown in the media column.

OPTIONS

-h Help option shows usage.

SEE ALSO

snbackup(1), **snrestore**(1)

NAME

sncompare − Validate the consistency between the Tertiary Manager database and the filesystems.

SYNOPSIS

sncompare -m *mountpoint* -c at [**-d** *dirname*] [**-D** *debuglevel*] [**-P** *count*] [**-y**]

sncompare -m *mountpoint* [**-c db**] [**-d** *dirname*] [**-t**] [**-D** *debuglevel*] [**-P** *count*] [**-y**]

sncompare -m *mountpoint* −c fs [**-d** *dirname*] [**-D** *debuglevel*] [**-P** *count*] [**-y**]

sncompare -m *mountpoint* −c ff [**-d** *dirname*] [**-l** *dirlevel*] [**-D** *debuglevel*] [**-P** *count*] [**-y**]

sncompare -m *mountpoint* −c qf [**-d** *dirname*] [**-l** *dirlevel*] [**-D** *debuglevel*] [**-P** *count*] [**-y**]

sncompare -m *mountpoint* −c qd [**-d** *dirname*] [**-D** *debuglevel*] [**-P** *count*] [**-y**]

DESCRIPTION

This command is used only for Quantum support and should only be executed at the direction of Quantum technical support.

The **sncompare** command is used to validate the consistency of the database versus a filesystem. All tests validate only one mountpoint at a time.

The quick tests, 'qd' and 'qf', provide a snapshot of the consistency of the Tertiary Manager database and whether all files in the specified file system are listed in the database. These quick tests can help determine which filesystems, if any, may need to be repaired.

You can skip the above quick tests, and run the '-c db' and the '-c ff' tests on each file system to verify all problems and generate repair operations. Alternately, you can run the 'at' and 'fs' tests as they run faster than the respective 'db' and 'ff' tests, but are less thorough in scope. All of these tests generate SQL and shell commands to repair the database and files.

See the TEST DESCRIPTIONS and QUICK TEST DESCRIPTIONS sections for more details about the problems each test detects.

No changes are made to the database or the filesystem during the execution of the sncompare utility. All repair commands are written to files that can be run at a later time to repair any problems found. See section REPAIR OUTPUT FILES for more details about these repair files.

Several logfiles are generated during the execution of this utility. They are stored in the directory specified by the **-d** option. If not specified then */tmp/sncompare_output* is used.

OPTIONS

-m *mountpoint*

Specifies the mountpoint of the filesystem to validate. This option is required for all tests.

-c at | db | fs | ff | qd | qf

Specifies which test to run. The default is to run the **db** test. Only one test can be run at a time. The tests are:

at Faster version of the **db** test that performs fewer checks
db Validate database vs. filesystem on all entries in FILEINFO table
fs Validate filesystem vs. database on all directories in DIRPATH table
ff Run filesystem validation on all directories and files in the mountpoint
qd Run quick database checking tests
qf Run the quick filesystem test

-d *dirname*

Specifies the name of the directory where all the various repair and log files are stored. If this directory does not exist then it will be created. If this directory exists then sncompare will display an error message and terminate. The default directory is */tmp/sncompare_output*.

The specified directory name can be a relative or a full pathname. A full pathname must start with a "/" (forward-slash) character. A relative pathname begins with "./".

-h Print the usage for the command.

-H This option displays a help menu describing the various debug features supported by **sncompare**. These debug options should only be used by Quantum technical support.

-b Specifies the number of files to include in a **fsretrieve** batch file, and the default is 300. This is usually a good value, but it should be increased if the system consists of mostly small files, e.g. less than a 10-20MB, or decreased if the systems consists of very large files, e.g. larger than 50GB.

-i This option enable running the two slow '-c qd' tests, and the default is not to run them. Both tests can take several hours to run on a large system, and can be skipped unless there are suspected issues with duplicate keys in the FILEINFO database tables.

-l *levels*

This option specifies the number of directory levels to use when distributing the workload across processes to decrease the overall execution time. This option only applies to the '-c ff' and '-c qf' tests, and is ignored by all other tests. The option range is 1..5, with a default of 1.

A value of 1 specifies only use the directories immediately under the mountpoint, a value of 2 specifies the directories immediately under the mountpoint plus their subdirectories, etc. The tests recurse only for the directories at the specified level and above. For example, a value of 2 means the tests will not recurse on the directories immediately under the mountpoint but will recurse through all their subdirectories.

This allows more directories to be spread across the processes to decrease the execution time of these two tests.

This option creates temporary files named **sncompare_DirList** appended with the level, e.g. **sncompare_DirList1**. Each contains a list of directories found at that level. These can be retained for analysis by specifying the -K option on the command line. Below is an example of one these files.

```
# cat /tmp/ff/sncompare_DirList1
/stornext/snfs1/sncompare_testdir4
/stornext/snfs1/extreps
/stornext/snfs1/unmanaged
/stornext/snfs1/OPS
/stornext/snfs1/sncompare_testdir3
/stornext/snfs1/sncompare_multiseg
/stornext/snfs1/sncompare_testdir2
/stornext/snfs1/special_files
/stornext/snfs1/sncompare_testdir1
```

-p *processes*

This option specifies the number of child processes to use for all tests, except the '-c qf' testing. The range is 1 through 16, inclusive, and the default is 8.

-s Disables the generation of repair output files. This option is intended for debug purposes only, and should not be specified in most cases.

-t Enables the gathering and reporting of the total disk space and truncated blocks for all files. The total number of blocks truncated does not include the stub size, if any. This option only applies to the '-c db' test.

-y When specified, the "Do you wish to continue (y/n)" confirmation prompt is skipped and the test starts immediately.

-B Enables a 300 second sleep when a child process starts to allow debugging of any child process (for debug purposes only).

-D *level*

Enables debugging output. When enabled, the logfile will grow about 700MB per 1 million files processed. The debug levels are **0-5**, **8**, and **9**. A higher debug level includes all lower levels. For example, "-D 5" includes levels 0 through 5. The debug levels include the following information:

- **0** informational debug messages
- **1** function entry/exit and call/return status
- **2** operational data points
- **3** database operations
- **4** file status, such as FOUND, REMOVED, MOVED, etc.
- **5** miscellaneous debug
- **8** very verbose debugging mode used only by the '-c fs' test; includes list of all entries found and processed
- **9** enables dump of inode xrep data and xrep debug; used only by the '-c fs' test; log filenames are "sncompare_debug_xrep[0-N]", where N is the number of processes.

-E Exclude testing of file_keys in the range specified by the -W/-X options (inclusive). This option is only valid for '-c at' or '-c db' tests. The default is off.

-F *pathname*

This option runs the '-c fs' test only on the specified file. It also sets the debug level to the maximum level. The test runs on a single process.

-K When specified these temporary files are not removed when **sncompare** terminates. This option is meant for debug purposes only.

```
Process statistics file: sncompare_stats.txt
FF/QF directory lists  : sncompare_DirList*
```

-L *pathname*

Enables checking of a list of file_keys, one file_key per line, by the '-c at' and '-c db' tests. In other words, the verification is limited to just these file_keys. The test runs on a single process.

-N *count*

Enables logging when unstored files and moved files are detected. If the -s option is specified then then removed files are also logged when detected. Using this option on a full file system may result in a very large log file. This creates the **sncompare_normal.txt** file. file.

-P *count*

Enables logging of progress entries to the **sncompare_log.txt** file. The *count* field specifies how often to log a progress message. For example, **-P 2000** will log a progress message in the logfile every 2,000th entry. The default value is 10000.

-V *processNumber*

Enables debug logging only by the specified process, values are 1 to 16. Also see the -p option. This option is for debug purposes only.

-W *key* This option specifies the first file_key for either the '-c at' or '-c db' test to verify. If the -X option is not specified then only this file_key is verified. If the -E option is not specified then the '-D 0' option is automatically enabled, and the test runs on a single process.

-X *key* This option specifies the last file_key for '-c at' or '-c db' test to verify. This option also requires the -W option to be specified.

-Y *directory*

This option specifies a directory for the ’-c at’, ’-c db’, ’-c fs’, or ’-c ff’ test to verify. The parameter must be a full pathname, i.e. it must begin with a ’/’. All tests except ’-c fs’ run recursively on the directory. The ’-c fs’ test only verifies the entries in the directory and ignores subdirectories. The ’-c fs’ test runs on a single process, and all other tests run on the specified number of processes (see **-p** option).

-Z *pathname*

This option enables logging of debug information for each file pathname in the specified file. The parameter must be a full pathname, i.e. it must begin with a This option applies to all tests except ’-c qd’.

USAGE

This should only be run at the direction of Quantum support. Here are the commands to run the six different tests with default values.

```
sncompare -m <mountpoint> -d /tmp/sncompare_at -c at -y
sncompare -m <mountpoint> -d /tmp/sncompare_db -c db -y
sncompare -m <mountpoint> -d /tmp/sncompare_fs -c fs -y
sncompare -m <mountpoint> -d /tmp/sncompare_ff -c ff -y
sncompare -m <mountpoint> -d /tmp/sncompare_qd -c qd -y
sncompare -m <mountpoint> -d /tmp/sncompare_qf -c qf -y
```

Below are some examples for running the **sncompare** utility. This is not an exhaustive list but just the main ones that have been run. All examples use the *-y* option to skip the confirmation prompt.

1. Run the ’-c db’ test on all file_keys in the specified directory and all subdirectories. This is useful when the problems are in a specific subdirectory.

```
sncompare -m <mountpoint> -d /tmp/sncompare_db -c db -Y /stornext/snf
```

2. Run the ’-c db’ test with debug enabled, gather/report total disk disk space and truncated blocks.

```
sncompare -m /stornext/snfs1 -c db -d /tmp/run1 -D 5 -t -y
```

3. Run the ’-c db’ test on a single file_key.

```
sncompare -m /stornext/snfs1 -c db -d /tmp/run2 -W 1000000 -y
```

4. Run the ’-c db’ test on a range of file_keys (inclusive) with a relative pathname for the **-d** option. The directory "run3" will be created in the current working directory.

```
sncompare -m /stornext/snfs1 -c db -W 1000000 -X 2000000 -d ./run3 -y
```

5. Run the ’-c db’ test on all file_keys except those between 1000000 to 2000000 (inclusive). This is useful to skip a range of file_keys affected by product alert 50.

```
sncompare -m /stornext/snfs1 -c db -E -W 1000000 -X 2000000 -d /tmp/r
```

6. Run the ’-c fs’ test on single directory. Only the contents of this directory are checked, i.e. there is no recursion on subdirectories.

```
sncompare -m <mountpoint> -d /tmp/sncompare_fs -c fs -Y /stornext/snf
```

7. Run the ’-c ff’ test on single directory. Unlike the ’-c fs’ test, this test does recurse on subdirectories. Thus all entries under this directory are checked.

```
sncompare -m <mountpoint> -d /tmp/sncompare_ff -c ff -Y /stornext/snf
```

8. Run the ’-c fs’ test and log debug information for the files listed in the file specified in the -Z option. Below is an example of the -Z option file contents with no escape characters and one file per line. In this example, all of the special characters, like the ’´ and ’/’ characters, are part of the directory and/or file name.

```
/stornext/snfs1/sncompare_testdir3/not_all_copies_01.txt
/stornext/snfs1/sncompare_testdir3/not_all_copies_02.txt
/stornext/snfs1/sncompare_testdir3/not_all_copies_03.txt
/stornext/snfs1/sncompare_testdir3/_!(!!cg"v!(@!˜@"p!’4!|w"3!’8!cg"r!
/stornext/snfs1/sncompare_testdir3/dir2file-\).*_\!\(\!\!cg"v\!\(@\!˜
```

This is the command to run.

```
sncompare -m <mountpoint> -c fs -d ./fsZ -Z /tmp/zfiles -y

### And these are the entries logged in the debug output logs.
# grep "Reporting on file:" fs/sncompare_debug_p*
fs/sncompare_debug_p2.txt:Mon Jan 10 09:51:16 2022.597216:     Report
fs/sncompare_debug_p2.txt:Mon Jan 10 09:51:16 2022.606680:     Report
fs/sncompare_debug_p2.txt:Mon Jan 10 09:51:16 2022.613529:     Report
fs/sncompare_debug_p2.txt:Mon Jan 10 09:51:16 2022.614549:     Report
fs/sncompare_debug_p2.txt:Mon Jan 10 09:51:16 2022.725817:   Reportin
```

9. The **-Z** option also can be combined with the **-Y** option as follows. However, in this case there is no recursion into subdirectories so the last entry is not checked and no debug information is logged for it.

```
sncompare -m /stornext/snfs1 -c fs -d ./fs1 -Y /stornext/snfs1/sncomp

# grep "Reporting on file:" fs1/sncompare_debug_p*
fs1/sncompare_debug_p0.txt:Mon Jan 10 09:56:40 2022.736089:    Report
fs1/sncompare_debug_p0.txt:Mon Jan 10 09:56:40 2022.752678:    Report
fs1/sncompare_debug_p0.txt:Mon Jan 10 09:56:40 2022.757846:    Report
fs1/sncompare_debug_p0.txt:Mon Jan 10 09:56:40 2022.758259:    Report
```

10. Run the ’-c ff’ test on the entire mountpoint by distributing the subdirectories at levels 1 through 3 across 16 processes. Recursion only occurs at level 3 and above. Directories at the following levels will be distributed.

```
/<mountpoint/*
/<mountpoint/*/*
/<mountpoint/*/*/*
```

For example, say level 1 contained 9 subdirectories, level 2 containing 33 subdirectories, and level 3 contained 23 subdirectories. Thus, a total of 65 subdirectories would be distributed across 16 processes, and recursion would only occur starting with the 23 level 3 subdirectories and above.

This is the command to run.

```
sncompare -m <mountpoint> -d /tmp/run10 -c ff -l 3 -p 16 -y
```

TEST DESCRIPTIONS

Below is a list of conditions reported by the **-c at** test.

- File found in the file system and database state is correct.

- File found in the file system, Inode key is 0, inode oneup is non-zero, and the file is not store excluded.
- Active-Replaced - Original file has been moved, new file with same name was created/stored, and database shows the original file as active. A possible cause is the FILECOMP table entry was not updated correctly, for example bug 78959.
- Active-Removed - Original file is not on the file system, but database shows it as active. A possible cause is the FILECOMP table entry was not updated correctly.
- Parent directory not found in the database.
- Moved files.
- Removed files.
- Replaced files.
- File_key not found in FILEINFO and/or FILECOMP database table. If the FILEINFO table entry is missing, the "**fsmedinfo -l**" command will list the file as "PATH UNKNOWN".

Below is a list of conditions reported by the **-c db** test.

- All issues found by the **-c at** test.
- Missing FILECOMP database table entries.
- Files with hard-links.
- Files not found in the file system.
- Correct number of copies are made but the inode cpymap is greater than the store policy cpymap value

Below is a list of conditions reported by the **-c fs** or **-c ff** test. The **-c fs** is faster but will miss directories that are not in the database. The **-c ff** is slower but does process all subdirectories regardless if they are in the database or not. See the note in the DESCRIPTION section above.

- FILE_KEY found in Database
- FILE_KEY not found in Database
- No Parent Found for File
- File with no FILECOMP database table entries
- File with no FILEINFO or FILECOMP database table entries
- Unknown File: non-zero file_key and stored copies but file size and blocks are both 0
- Correct number of copies are made but the inode cpymap is greater than the store policy cpymap value
- Non-Stored File
- Moved/Renamed File
- File with a matching file_key entry in FILEINFO database table but a different pathname (could be an ORPHAN or a hard link)
- 0-Length File
- Files with hard links
- Managed Directory

  These are directories associated with a store policy, i.e. the class index in non-zero.
- Other entries such as ".", "..", links, non-regular files, etc.
- Unmanaged Directory

These are directories not associated with a store policy, i.e. the class index is zero.

- External Representation (xrep) Data

Command Xrep Fields

```
Wrapper Length
Wrapper Segment number
```

All Media Fields

```
mediaid       xrp_seg_medium (required field)
fcmodtime     xrp_seg_time   (required field)
fcadddate     xrp_add_date   (required field)
dm_oneup      xrp_oneup      (required field)
fcversion     xrp_version    (required field)
fccksum       xrp_checksum   (required field)
fcobjid       xrp_seg_uuid   (optional)
path          xrp_path       (optional)
fcocomprtype  xrp_compr_type (optional)
fcocomprsize  xrp_compr_len  (optional)
fcoentype     xrp_encr_type  (optional)
fcodpkid      xrp_encr_keyid (optional)
```

Tape Media Only Fields

```
fcmcdbn       xrp_cdbn (optional)
fcmfsn        xrp_fsn  (optional)
fcmldbn       xrp_ldbn (optional)
```

Below is a list of database tables checked by the **-c qd** test.

```
DIRPATH
FILECOMP
FILEINFO
MEDIADIR
OLDMEDIA
MEDIA
MEDIACRITERIA
ARCHIVEMEDIA
```

When the **-c qf** option is specified, a quick test is run to validate that all files in the specified filesystem are listed in the Tertiary Manager database.

QUICK TEST DESCRIPTIONS

The following database checks are run in the **-c qd** test on the mountpoint specified by the **-m** option. No repair operations are generated.

check_dirpath
: Find all duplicate DIRPATH database table entries.

check_filecomp
: Find all FILECOMP database table entries without a corresponding FILEINFO database table entry. There is no checking of the media state.

check_filecomp1
: Runs a similar check as the check_filecomp but this will also check if the media associated with the file has had **fsrminfo(1)** run on them. If this is the case then the only way to fix is to remove the FILECOMP database table entries.

check_filecomp2
Find FILECOMP entries without a corresponding MEDIADIR entry.

check_filecomp3
Find FILECOMP entries where the checksums of all copies of the same file_key/version/segment do not match.

check_fileinfo
Find FILEINFO entries with a name like "KEY.*". This usually indicates the file was moved and modified while it was being stored. The name in the FILEINFO entry is not updated unless the user does a manual **fsstore(1)** on the file.

check_oldmedia
Find all entries in the OLDMEDIA database table that do not have a corresponding entry in the MEDIADIR database table. Removing these entries will allow **fsclean(1)** to run as not removing them will cause **fsclean(1)** to die reporting that it could not find the media in the MEDIADIR database table.

check_relpts
Check that all relation point directories found in the CLASSDIR database table exist in the filesystems.

check_removed_files
Obtain a list of files that have been removed.

check_not_all_copies_made
Check all FILEINFO entries to ensure that all expected copies are stored. It will report either:

- Not all copies are stored.
- Too many copies are stored.

check_parent_keys
Find all FILEINFO database table entries where the parent file_key is either 0 or is missing from DIRPATH database table.

check_dmapi
Parses the TSM/internal/mapping_dirs files and logs the number of directories and files found. This check is only run if the **-i** option is specified.

check_fileinfo_path_duplications
Check for FILEINFO entries where there are duplicate names assigned to different file_keys, the same pfile_key, and both are active. This check is only run if the **-i** option is specified.

check_media
Check all media found in the MEDIADIR database table for the following conditions:

```
MEDIADIR file count equals FILECOMP matching entry count
Media exists in MEDIACRITERIA database table [MSM]
MEDIADIR.status equals MEDIACRITERIA.field2
MEDIACRITERIA file count equals FILECOMP matching entry count
Media exists in MEDIA database table [MSM]
MEDIA.locationstate = 1 (media is in an archive or vault)
Media exists in ARCHIVEMEDIA database table [MSM]
MEDIA.currentarchiveid = ARCHIVEMEDIA.archiveid
```

The **-c qf** test runs on the file system specified by the **-m** option. This test verifies whether all files found in the filesystem are in the database. The goal of this test is to be thorough, i.e. check all filesystem entries, but also run very quickly. The following checks are made:

- Search for file_key of the parent directory in the DIRPATH database table

- Search for leaf name and parent file_key in the FILEINFO database table

If either SQL query fails then an appropriate message is logged indicating an error was detected. No repair operations are generated.

EXIT STATUS

Exit codes for the **sncompare** command are:

```
0    Command completed successfully.
1    Command syntax error.
           An invalid argument was specified on the command line.
           Retry the command with the correct argument.
2    Stornext environment initialization error.
3    DMAPI initialization error.
4    DMAPI read error.
5    Call to read file/directory stats failed.
6    Memory allocation error.
7    [This exit code is unused.]
8    Linked list error.
9    File open error.
10   File read error.
11   Specified directory path is null.
12   Directory not found.
13   Directory create error.
14   Directory open error.
15   Directory read error.
16   Database open/connection error.
           Verify that sncompare is being run on the primary MDC
           and that the SNSM database component is running.
17   Database data error.
18   Database query error.
19   File key is zero error.
20   Parent not found.
21   User canceled command.
22   No relationpoints found in CLASSDIR database table.
           Indicates the check_relpts check, part of the -c qd test,
           did not find the expected entries in the CLASSDIR database table.
           This is unexpected because the mountpoint is managed.
23   [This exit code is unused.]
24   File data not found.
25   File not found.
26   Unexpected child failure.
27   Database exception error occurred.
28   Cannot run sncompare due to database failure.
29   Directory exists--choose a non-existent directory name.
           The directory specified on the -d option already exists.
           Either remove the directory or choose a non-existent
           directory.
30   A call to system() failed.
31   Not enough free space in directory.
           Verify there is adequate disk space in the /tmp directory
           to dump the DIRPATH database table.
33   Maximum pathname length of 1024 characters exceeded.
           Indicates that a pathname exceeded the maximum pathname
           length of 1024 characters. sncompare logs this event in
           the "sncompare_log.txt" logfile.
```

```
34   Mountpoint pathname specified does not exist.
           Check that the mountpoint specified on the -m option
           is a valid managed file system.
35   Full pathname required for directory.
           The directory specified on the -d option must be a
           fully-qualified directory pathname, i.e. it must start
           with a "/" character.
36   Cannot run on specified mountpoint--used by storage disk.
           sncompare cannot be run on a filesystem that is used
           as an SDISK.
37   The filesystem specified by the mountpoint is not mounted.
           Check if the filesystem is mounted and if not then
           mount it.
38   Failed to get handle for mountpoint.
           Verify that the filesystem is a valid StorNext filesystem.
39   Call to read file/directory stats failed due to I/O error.
40   Mountpoint not found in database. Verify -m option is valid.
           Check that the mountpoint specified on the -m option
           is a valid managed file system.
41   General sncompare compare detected.
42   External representation (xrep) failure.
43   File chmod error.
44   Invalid NULL pointer argument error.
```

REPAIR OUTPUT FILES

These executable files contain SQL and/or shell commands that will fix or remove information from the filesystem and/or database to resolve problems and inconsistencies that were found during the validation.

All repair scripts are created in the *repair* subdirectory in the directory specified by the **-d** *dirname* option. For example if the **-d** option was specified as */tmp/sncompare_run1* then the subdirectory would be */tmp/sncompare_run1/repair*.

Below is a list of each SQL repair script created and a description of its contents. The SQL repair script names, except for "sncompare_oldmedia.sql" are now suffixed with the file system device_key (denoted by the ’X’ in the name), for example, "sncompare_fileinfo1.sql".

sncompare_dirpathX.sql

Contains SQL commands to add or update the parent directories in the DIRPATH database table for the following types of errors:

- Parent is not found in the database table. This can also occur if the file FILEINFO entry has the pfile_key field set to 0 and there is no entry in the database table for its parent directory.
- Database table entry exists but the path is null.

sncompare_moved_dirpathX.sql

Contains SQL commands to add or update the parent directories in the DIRPATH database table for a file that has been moved since it was stored.

sncompare_filecompX.sql

Contains SQL commands to fix entries in the FILECOMP database table for the following types of errors:

- Delete entries from the FILECOMP database table that have no corresponding entry in the FILEINFO database table and the media are waiting to be cleaned, i.e. they are in the OLDMEDIA database table.
- Delete entries from the FILECOMP database table where the media are not found in the MEDIA database table.

- Set ENDTIME in the FILECOMP database table entries of moved (or replaced) files so that **fsclean(1)** will clean them up.

sncompare_fileinfoX.sql
Contains SQL commands to fix entries in the FILEINFO database table for the following types of errors:

- Update the parent_key if the current parent_key in the database table is 0.
- Add an entry to the FILEINFO database table since it is missing for some unknown reason.

sncompare_moved_fileinfoX.sql
Contains SQL commands to fix entries in the FILEINFO database table for the following types of errors:

- Update the parent_key to the new directory for a file that has been moved since it was stored.

sncompare_oldmedia.sql
Contains SQL commands to remove unneeded OLDMEDIA database table entries, i.e. media not in MEDIADIR database table.

sncompare_rmfileinfoX.sql
Contains SQL commands to update the RMFILEINFO database table. These commands will either remove unneeded entries or insert missing entries.

Below is a list of each repair shell script created and a description of its contents.

sncompare_chfiat.sh
Contains fschfiat commands to modify the file attributes such as the number of copies.

sncompare_dmutil.sh
Contains dm_util commands to clear the ALL_COPIES_MADE flag in the extended attributes of a file so it can be retrieved.

sncompare_rmcopy.sh
Contains "**fsrmcopy -c**" commands to invalidate the specified copy of files so new store candidates are generated. For example, if the external represntations are missing from the file inode then this will restore the file to hopefully update the inode correctly.

sncompare_rm.sh
Contains "**rm -f <file>**" commands to remove files from the filesystem that are truncated and do not exist in the database anymore.

sncompare_rtv.sh
Contains "**fsretrieve(1) -a -B**" commands for each file that has fewer copies stored to media than expected by the store policy or if the external represntations are missing from the file inode. Each fsretrieve processes a batch of files which are named **sncompare_batch_xxx**. This in conjunction with updating the extended attributes of the file will result in storing the missing copies.

sncompare_xrep.sh
This is unused at this time because there is no method to determine whether the inode xrep or the FILECOMP database table data is correct. All discrepancies must be manually repaired.

sncompare_batch_xxx
Each of these files contain a list of files, one per line, to be retrieved by the **fsretrieve(1)** commands contained in the **sncompare_rtv.sh** repair script. Each batch file contains 300 files, this is the default, which can be changed by the **-b** option.

REPAIR OPERATIONS

These the most common types of database problems that have been found.

1. Fsmedcopy fails to copy all files (see bug 78959)

This occurs because the file(s) are not on disk but the FILECOMP table "endtime" field is set to MAXINT. Either the **-c at** or the **-c db** test will detect this condition and generate repair operations.

After running either test, run these two repair scripts in this order to correct the FILECOMP table records and insert any missing records in the REMFILEINFO table.:

sncompare_filecompX.sql
sncompare_rmfileinfoX.sql

2. General Overall Database Problems

Over time, the database can get out of sync with the file systems due to software bugs, incorrect usage/procedures, configuration changes, migratrions, etc. Here are some of the problems that could exist.

1. Directories missing in the database
2. Files in the database but not on the FS
3. Files on the FS and on tape but not in the database
4. Files with incorrect or differing attributes in the database and/or FS
5. Media entries missing from the database
6. Database contains old records that are no longer needed

Run the **-c db** and the **-c ff** tests to detect all failures and generate repair operations.

After running both tests, run the following repair scripts in the order listed to correct all tables. First run these scripts in the **-c db** repair directory and then the same scripts in the **-c ff** repair directory.

sncompare_dirpathX.sql
sncompare_filecompX.sql
sncompare_fileinfoX.sql
sncompare_rmfileinfoX.sql
sncompare_oldmedia.sql

When each of the above repair scripts is executed, please redirect the output to a new file. Examine this output to determine if the repair script successfully executed. Do not proceed to the next repair script until all problems with the current repair script are resolved.

3. Moved/Renamed Files

This can be a concern when performing a disaster recover (DR). Run the **-c db** and the **-c ff** tests to detect all these types of failures and generate repair operations.

These two scripts can optionally be executed to update the database so the parent directory of moved files references their current location and not their stored location. Whether these are executed or not does not affect StorNext functionality, it is an expected condition that the names in the database will be stale when files are moved. If executed then follow the same guidelines as described above.

sncompare_moved_dirpathX.sql
sncompare_moved_fileinfoX.sql

4. Incorrect Number of Copies

To correct the number of copies stored to media run the **-c db** and the **-c ff** tests to detect all these types of failures and generate the needed repair operations.

After running both tests, run the following repair scripts in the order listed to correct all tables. First run these scripts in the **-c db** repair directory and then the same scripts in the **-c ff** repair directory. The sncompare_rtv.sh script will only contain entries if any of the files are truncated. Thus if this file is empty then it does not need to be run.

sncompare_dmutil.sh
sncompare_rtv.sh
sncompare_rmcopy.sh
sncompare_chfiat.sh

After running the sncompare_dmutil.sh script, a rebuild policy should be run to create store candidates for the files that are updated. By default, this is run Saturdays, however, it can be run manually to ensure the files are stored as soon as possible.

Note that the rebuild policy now makes use of information in the mdarchive for determining candidates. This archive can run a few minutes behind the time of the actual file system activity. It will be best to wait a few minutes after running the sncompare_dmutil.sh script before running the rebuild policy.

The rebuild command is:

```
fspolicy -b -y filesystemmountpoint
```

See **fspolicy(1)** for more information.

5. Remove Old Copies

The last file, *sncompare_rm.sh*, can be optionally run to remove old files from the filesystem.

LOGGING OUTPUT FILES

These files contain information about what types of entries were found and whether they were repairable or not.

Below is a list of each of these logfiles created and their contents. These are in the directory specified by the **-d** option.

sncompare_summary.txt
Contains a summary of the number of errors/warnings from each test that was run. It also includes the arguments passed to **sncompare** and start and end dates/times of each test.

sncompare_log.txt
Contains various error and warning messages.

It also contains entries for unknown files whose state cannot be ascertained. No action is taken on these files. An example of this type of file is one with a non-zero key and copies but 0 size and 0 blocks.

Also includes entries for 0-length files and pure sparse files found in the filesystem. These entries are okay, but not they are not stored on media and cannot be recovered from the database via the **fsrecover(1)** command.

sncompare_debug_p[0-N].txt
Contains debugging information enabled by the **-D** option levels 0-5, 8, and 9 on the command line. There is a file for each process, 0 through N, where N is the number of processes.

sncompare_debug_xrep[0-N].txt
Contains the file xrep data dumped when the **-D** option level 8is specified on the command line. There is a file for each process, 0 through N, where N is the number of processes.

SNCOMPARE_SUMMARY.TXT FILE DESCRIPTION

Common Header Description

The beginning of the file contains the following information:

1. Starting date/time of the **sncompare** run
2. StorNext release version
3. List of arguments

Below is an example:

```
Start time = Fri Sep 17 09:31:58 2021

sncompare, Version = 7.0.2(102687D)
Arguments: -m /stornext/snfs1 -d /tmp/baseline_db -c db -y
```

AT Test Summary Description

If the **at** test is run, the format of this file is as follows:

```
Start time = Wed Sep  1 12:44:33 2021

sncompare_rh6_6311, Version = 6.3.1.1(88319E)
Arguments: -m /stornext/ARCV -c at -d /scratch/SR596496/sncompare_ARCV_nodups_di

*************************************************************************
*******                  Checking active files vs. FS              *******
*************************************************************************

  Total number of FILEINFO entries processed  = 317998053
  Total number of files with dirpath errors   = 0
  Total number of files found in FS           = 317687880
  Total number of files not found in FS       = 0
  Total number of removed files (ignored/ok)  = 727
  Total number of active-removed files        = 1552
  Total number of renamed files (ignored/ok)  = 28874874
  Total number of replaced files (ignored/ok) = 1168328
  Total number of active-replaced files       = 302006
  Total number of files with hard-links       = 0
  Total number of missing external reps       = 0
  Total number of bad external reps           = 0
  Total number of files: no filecomp records  = 0
  Total number of files: unneeded RMFILEINFO  = 7
  Total number of non-stored files (ok)       = 0
  Total number of other entries (ignored)     = 0
  Total number of excluded entries (ignored)  = 307894
  Total number of repair operations           = 607123

End time = Mon Sep 13 16:05:31 2021
```

DB Test Summary Description

If the **db** test is run, the format of this file is as follows:

```
*************************************************************************
*******              Checking database vs. filesystem             *******
*************************************************************************

  Total number of FILEINFO entries processed  = 4339
  Total number of files with dirpath errors   = 21
  Total number of files found in FS           = 4335
  Total number of files not found in FS       = 0
  Total number of removed files (ignored/ok)  = 3
```

```
  Total number of active-removed files        = 1
  Total number of renamed files (ignored/ok)  = 15
  Total number of replaced files (ignored/ok) = 2
  Total number of active-replaced files       = 0
  Total number of files with hard-links       = 0
  Total number of missing external reps       = 0
  Total number of bad external reps           = 0
  Total number of files: no filecomp records  = 2
  Total number of files: unneeded RMFILEINFO  = 1
  Total number of non-stored files (ok)       = 0
  Total number of other entries (ignored)     = 0
  Total number of excluded entries (ignored)  = 0
  Total number of repair operations           = 8

End time = Fri Sep 17 09:32:08 2021
```

Report FS Disk/Truncation Space (DB test only)

Reports the total disk space used by all files verified in the **db** test. It also reports the total disk space that has been truncated. This report is present only if the **-t** option is specified.

Below is an example of the disk/truncation space report in the *sncompare_summary.txt* file. The data is displayed in binary units, e.g. MiB and GiB.

```
Report FS Disk/Truncation Space:

  Total file disk space on FS = 1.40e+02MiB
  Total file space truncated  = 1.14e+00MiB
```

FF Test Summary Description

If the **ff** test is run, the format of this file is as follows:

```
**********************************************************************
*******          Checking full filesystem vs. database          *******
**********************************************************************
Start time = Wed Apr 27 09:34:42 2022


  Total number of FS entries found                = 5616
  Total number of subdirectories in FS (ignored)  = 422
  Total number of skipped entries in FS (ignored) = 844
  Total number of special files in FS (ignored)   = 3
  Total number of files found in DB               = 4299
  Total number of files not found in DB           = 0
  Total number of files: parent missing           = 21
  Total number of files: filecomp records missing = 2
  Total number of files: fileinfo records missing = 6
  Total number of files: fileinfo/filecomp missing = 0
  Total number of unknown files                   = 0
  Total number of non-stored files (ok)           = 0
  Total number of renamed files (ok)              = 15
  Total number of replaced files (ok)             = 1
  Total number of files with hard-links (ok)      = 1
  Total number of orphaned/softlinked files (ok)  = 0
  Total number of zero length files (ok)          = 2
```

```
  Total number of directory mismatches (DB vs FS)  = 0
  Total number of directories not found in FS      = 0
  Total number of unmanaged directories in FS (ok) = 3
  Total number of missing external reps            = 3
  Total number of bad external reps                = 6
  Total number of repair operations                = 25

Intermediate time = Wed Apr 27 09:35:04 2022

End time = Wed Apr 27 09:35:04 2022
```

FS Test Summary Description

If the **fs** test is run, the format of this file is as follows:

```
**************************************************************************
*******             Checking filesystem vs. database             *******
**************************************************************************
Start time = Wed Feb  9 12:51:21 2022


  Total number of FS entries found                 = 5589
  Total number of subdirectories in FS (ignored)   = 414
  Total number of skipped entries in FS (ignored)  = 828
  Total number of special files in FS (ignored)    = 3
  Total number of files found in DB                = 4299
  Total number of files not found in DB            = 0
  Total number of files: parent missing            = 11
  Total number of files: filecomp records missing  = 2
  Total number of files: fileinfo records missing  = 6
  Total number of files: fileinfo/filecomp missing = 0
  Total number of unknown files                    = 0
  Total number of non-stored files (ok)            = 0
  Total number of renamed files (ok)               = 12
  Total number of replaced files (ok)              = 1
  Total number of files with hard-links (ok)       = 1
  Total number of orphaned/softlinked files (ok)   = 0
  Total number of zero length files (ok)           = 1
  Total number of directory mismatches (DB vs FS)  = 2
  Total number of directories not found in FS      = 11
  Total number of unmanaged directories in FS (ok) = 1
  Total number of missing external reps            = 3
  Total number of bad external reps                = 6
  Total number of repair operations                = 20

Intermediate time = Wed Feb  9 12:51:33 2022

End time = Wed Feb  9 12:51:33 2022
```

Quick Database Test Summary Description

The format of this file is described below. At the top are the version and arguments following by a header to the device_key (or mountpoint) being tested.

```
Start time = Fri Feb  4 12:32:16 2022
```

```
sncompare, Version = 7.0.3(105108D)
Arguments: -m /stornext/snfs1 -c qd -d /tmp/baseline_qd -y

**************************************************************************
******* Running the Quick Tests for device_key:    1             *******
**************************************************************************
```

After this is a summary of the results of each quick test separated by a line of "*" characters. It also includes the start date/time of each quick test. Below is an example of this output.

```
Start time = Fri Feb  4 12:32:16 2022

 Results for check_dirpath() test:
==========================================================================
Found 3 DIRPATH1 duplicate entries

End time = Fri Feb  4 12:32:16 2022

**************************************************************************
Start time = Fri Feb  4 12:32:16 2022

 Results for check_filecomp() test:
==========================================================================
Found 7 FILECOMP1 entries with no FILEINFO1 entry

End time = Fri Feb  4 12:32:16 2022

**************************************************************************
Start time = Fri Feb  4 12:32:16 2022

 Results for check_filecomp1() test:
==========================================================================
Found 1 FILECOMP1 entries with no FILEINFO1 entry and cleaned media (errors)

End time = Fri Feb  4 12:32:16 2022

**************************************************************************
Start time = Fri Feb  4 12:32:16 2022

 Results for check_filecomp2() test:
==========================================================================
Found 4 FILECOMP1 entries missing MEDIADIR entry (errors)

End time = Fri Feb  4 12:32:16 2022

**************************************************************************
Start time = Fri Feb  4 12:32:16 2022

 Results for check_filecomp3() test:
==========================================================================
Found 0 FILECOMP1 checksum mismatches

End time = Fri Feb  4 12:32:17 2022

**************************************************************************
```

```
Start time = Fri Feb  4 12:32:17 2022

 Results for check_fileinfo() test:
==========================================================================
Found 1 FILEINFO1 entries with "KEY.*" name (errors)

End time = Fri Feb  4 12:32:17 2022

**************************************************************************
Start time = Fri Feb  4 12:32:17 2022

 Results for check_relpts() test:
==========================================================================
Found 0 missing relation points (errors)

End time = Fri Feb  4 12:32:17 2022

**************************************************************************
Start time = Fri Feb  4 12:32:17 2022

 Results for check_removed_files() test:
==========================================================================
Found 4 FILEINFO1 entries that have been removed

End time = Fri Feb  4 12:32:17 2022

**************************************************************************
Start time = Fri Feb  4 12:32:17 2022

 Results for check_not_all_copies_made() test:
==========================================================================
Found 4 FILEINFO1 entries in error (processed 4335 entries)

End time = Fri Feb  4 12:32:17 2022

**************************************************************************
Start time = Fri Feb  4 12:32:17 2022

 Results for check_parent_keys() test:
==========================================================================
Found 12 FILEINFO1 entries with incorrect/missing parent key

End time = Fri Feb  4 12:32:17 2022

**************************************************************************
******* Running the media specific Quick Tests                   *******
**************************************************************************
Start time = Fri Feb  4 12:32:17 2022

 Results for check_oldmedia() test:
==========================================================================
Found 2 OLDMEDIA entries with no MEDIADIR entry

End time = Fri Feb  4 12:32:17 2022
```

```
*************************************************************************
Start time = Fri Feb  4 12:32:17 2022

 Results for check_media() test:
=========================================================================
Found 7 of 16 MEDIADIR entries with errors

End time = Fri Feb  4 12:32:17 2022
```

The last entries include the total number of errors, the total number of warnings detected by the quick tests, and finally the end date/time of the **sncompare** run. For example:

```
*************************************************************************
*************************************************************************

Total Errors   = 20
Total Warnings = 25

End time = Fri Feb  4 12:32:17 2022
```

Quick Filesystem Test Summary Description

If the filesystem quick test, option **qf**, is run, the format of this file is shown below.

```
*************************************************************************
*******                    Running FS Quick Test                 *******
*************************************************************************
Start time = Fri Feb  4 12:32:17 2022

*************************************************************************
Start time = Fri Feb  4 12:32:25 2022

 Results for filesystem quick test:
=========================================================================
Found 422 managed directories
Found 3 unmanaged directories
Found 46 of 4356 filesystem entries with errors

End time = Fri Feb  4 12:32:25 2022
```

SNCOMPARE_LOG.TXT FILE ERROR EXAMPLES

This logfile contains the following information for each test run:

- Processes started and their status. When running in multi-processing mode, up to 8 child processes are run.
- Log file for each process.
- Errors and warnings detected by each test.

Each error entry is prefixed by one of the following strings:

[ER] Indicates an error was detected that prevents the processing of the file or directory.

[FX] Indicates the problem is fixable by running the repair scripts.

[MF] Indicates the problem may be fixable, but **sncompare** was unable to determine the fix. Manual intervention is required.

[NR] Indicates the problem is not fixable.

[OK] Indicates there is no problem to be repaired.

[RE] Indicates the file may be recoverable. For example using the information in the file inode if the file is on-disk.

[UK] This indicates the cause of the problem is unknown. It is also unknown how to repair the problem.

[OT] This indicates an entry was found on the file system that is not a managed file, directory, '.' entry, or '..' entry. Examples of this type of entry are a socket file and a block device file. These entries are ignored.

[NM] This indicates a non-managed directory was found.

[BX] This indicates one of two possible problems. The first problem is that the xreps exists but there were issues reading the contents. The second problem is where the FILECOMP entry and xrep data do not match. In either case the FILECOMP entry and the xrep exist, and the full pathname, copyid and segment are logged for the entry containing the error.

Below is the format of the log entry for problem #1 in the "sncompare_log.txt" logfile. Below is the format of the "sncompare_log.txt" log entry for problem #1.

[BX] Bad xrep (data decode failure): path=<path>, cpyid=<cpyid>, seg=0,

Below is the format of the "sncompare_log.txt" log entry for problem #2. The lastfield specifies the name of the mismatched xrep field.

[BX] FILECOMP/Xrep mismatch: path=%s, cpyid=%d, seg=%d, errcnt=%d, lastfield='%s'

Manual intervention is required to repair either of these failure types as the problem could either be in the inode or the DB. The first step is to determine which contains the more recent copy. This may be known only by the system administrator (who, e.g., may have restored the database or the file system from a backup). In that case, the staler copy should be restored from the more recent copy, if possible.

If the inode copy is the more recent then:

- Force retrieval using the xrep entry. If the file has a valid database entry with the same version, that will always be used. So the workaround is either reassign the copy in the inode/xrep, or delete/reassign the copy number in the DB, so that the last-resort retrieve from the inode xrep will be tried. For example, if "**fsretrieve -c 3 <file>**" is run, it will retrieve using the inode xrep copy 3 if and only if the file has no current copy 3 in the DB.
- Invalidate this copy by running "fsrmcopy -c <cpyid> <file>".
- Now store the file.

If the database copy is the more recent then:

- Retrieve the file normally as the database entry will always used.
- Invalidate this copy by running "fsrmcopy -c <cpyid> <file>".
- Now store the file.

[MX] This indicates an xrep is missing from the inode. The full pathname, copyid and segment are listed for the entry missing the xrep value.

Below are some example entries that could be logged in this logfile for the '-c at', '-c db', '-c ff', and '-c fs' tests.

[FX] FILE_KEY NOT IN DB (1100):
/stornext/snfs1/test/large1/popdir_1/popdir_1_1/popdir_1_1.1.090

Indicates the specified file does not have a FILEINFO entry. Both the file_key, the number in the parentheses, and the full pathname of the file are included.

[RE] FILECOMP Records Not Found:
'/stornext/snfs1/sncompare_testdir/managed/popdir_4/popdir_4_1/popdir_4_1.100.099'

Indicates the specified file does not have a FILECOMP entry.

[NR] Media Entry Not Found For File_Key 695339: ndx=999999, gen=0, classndx=7

Indicates the media entry for the specified file_key does not exist in the MEDIADIR database table. The mediandx, medgen, and classndx of the missing media are included.

[OK] HARDLINK(2) File Found: '/stornext/snfs1/sncompare_testdir2/popdir_2_2.1.007.hlnk'
Database File(good):
/stornext/snfs1/sncompare_testdir2/popdir_2/popdir_2_2/popdir_2_2.1.007

Indicates a hard link was found. The link pathname is listed on the first line. The pathname to which it is linked is listed on the second line, and this is the entry found in the database. The number in parentheses is the number of links detected. This is not a problem for StorNext. Below is a snippet from the **dm_info** output showing there are two links ("lnks: 2").

```
# dm_info /stornext/snfs1/sncompare_testdir2/popdir_2/popdir_2_2/popdir_2_2.1.007
Filename:     /stornext/snfs1/sncompare_testdir2/popdir_2/popdir_2_2/popdir_2_2.1.007
handle (hex): 0005cc31a9ced6dc000e00000000004e0000000002150746
  fsid: 0x0005cc31a9ced6dc  dev: 44  rdev: 44  aff: n/a
  size: 80    block size: 4096    number of blocks: 8
  ino: 34932550   gen: 78   type: S_IFREG   mode: 0100664   lnks: 2
  extended attributes:   attrname: SNEA   attr_ver: 7.40   key: 1683
    class: 2  oneup: 1683  vsn: 001  totvers: 1  cpymap: 0x3
    flags: 0x2   events: 0x160000   stub size,len: -1,0
    flags: ALL_COPIES_MADE
  eventlist:  0x00160000  WRITE  TRUNCATE  DESTROY
  region events: WRITE  TRUNCATE
  atime = 1631888661 -> Fri Sep 17 09:24:21 2021
  ctime = 1631888661 -> Fri Sep 17 09:24:21 2021
  mtime = 1631888661 -> Fri Sep 17 09:24:21 2021
```

[RE] PARENT NOT FOUND: file_key/pfile_key=691361/690877, name=popdir_1_1_3.100.041

[FX] PARENT NOT IN DB:
'/stornext/snfs1/sncompare_testdir/managed/popdir_2/popdir_2_2/popdir_2_2.100.002'

This indicates the parent was not found in the DIRPATH database table.

[FX] ACTIVE-REPLACED FILE FOUND:
/stornext/snfs1/sncompare_testdir1/managed/popdir_3/popdir_3_1/popdir_3_1_1/popdir_3_1_1.1.007
(fk=4493/pk=396/newfk=4597)

This indicates the file with file_key 4493 (fk) is still active, the FILECOMP.endtime field is 2147483647 and there is no RMFILEINFO entry, and the pathname matches that of the file with file_key 4597 (newfk). In other words, the original file was renamed and a new file with

the same file name created then stored.

[FX] ACTIVE-REMOVED FILE FOUND:
/stornext/snfs1/sncompare_testdir2/popdir_2/popdir_2_1/popdir_2_1.1.008 (fk=1674/pk=114)

This indicates the file with file_key 114 (fk) is still active, the FILECOMP.endtime field is 2147483647, but the pathname was not found on the file system. In other words, the file has been removed yet TSM views it as still active on the file system.

[RE] File Truncated && No Stored Copies:
/stornext/snfs1/sncompare_testdir2/popdir_4/popdir_4_1/popdir_4_1.1.008

This indicates a file that is on-disk, is truncated, is in the FILEINFO database table, **dm_info** indicates copies have been made, but there are no FILECOMP entries. The file may be recoverable using the inode information.

[FX] File has unneeded RMFILEINFO table entry:
/stornext/snfs1/sncompare_testdir2/popdir_2/popdir_2_1/popdir_2_1.1.005 (fk=1671)

This indicates a file with a RMFILEINFO database table entry that should not exist. The file is on-disk and the FILECOMP database table entry indicates the file is active. The repair operation is to remove this RMFILEINFO database table entry.

[RE] Not All Copies Stored:
/stornext/snfs1/sncompare_testdir2/popdir_3/popdir_3_1/popdir_3_1.1.008
[RE] Not All Copies Stored and inode version is 0: /snfs1/DATA/Incoming
Data/BRA/DATASUS_TABNET/NATIONAL_IMMUNIZATION_PROGRAM/Tocantins/Coverage/2017tocanti

Both messages indicate all copies have not been archived for this specified file. The second variation also is reported if the version field in the inode, vsn, is 0.

[NR] Mediadir Entry Not Found For File_Key 2524: ndx=999999, gen=0, classndx=2

This indicates an invalid mediandx/mediagen was found in the FILECOMP database table entry for the file with that file_key.

[RE] REMOVED FILE FOUND:
B/stornext/snfs1/sncompare_testdir2/popdir_2/popdir_2_1/popdir_2_1.1.008 (1674/114)

This indicates the specified file has been removed from the file system. The values in the parentheses are the file_key and the parent file_key, respectively.

[OK] MOVED FILE FOUND-NOW:
/stornext/snfs1/sncompare_testdir3/tempved(‘"!).moved/18 BayÃ¢t-e RÃ¢je’ and Suz-o GodÃ¢z.mp3.moved (fk=4611/pk=6)
or
[FX] RENAMED File Found: ’/stornext/snfs1/sncompare_testdir3/tempved(’‘"!).moved/18 BayÃ¢t-e RÃ¢je’ and Suz-o GodÃ¢z.mp3.moved’

This indicates that the specified file has been renamed (moved). The pathname of the file on the filesystem (displayed in the log entry) differs from the pathname generated from the database. Running **fsfileinfo** on the pathname will display both paths. Here is the output for the above file:

fsfileinfo /stornext/snfs1/sncompare_testdir3/temp\\Moved\(\'\`\"\!\).moved/18\ BayÃ¢t-e\ RÃ¢je\'\ and\ Suz-o\ GodÃ¢z.mp3.moved
---
File Information Report Mon Sep 20 10:02:46 2021
Filename: /stornext/snfs1/sncompare_testdir3/temp\Moved(’‘"!).moved/18 BayÃ¢t-e RÃ¢je’ and Suz-o GodÃ¢z.mp3.moved
Stored Name: /stornext/snfs1/sncompare_testdir3/18 BayÃ¢t-e RÃ¢je’ and Suz-o GodÃ¢z.mp3 ----------
Last Modification: 17-sep-2021 09:26:12

Owner: root Location: DISK AND ARCHIVE
Group: root Existing Copies: 2
Access: 664 Target Copies: 2
Expired Copies: 0
Target Stub: 0 (KB) Existing Stub: n/a
File size: 57 Store: MINTIME
Affinity: n/a Reloc: MINTIME
Class: sncompare_copy2 Trunc: MINTIME
Alt Store Copy: Disabled Clean DB Info: NO
Media: E00000(1) E00001(2)
Checksum: N
Encryption: N
Object Ids: N

[OK] REPLACED File Found:
'/stornext/snfs1/sncompare_testdir1/managed/popdir_3/popdir_3_1/popdir_3_1_2/popdir_3_1_1.1.007.moved_r

This indicates that the specified file has been renamed (moved) and a new file created using the original pathname. The pathname of the file on the filesystem is displayed.

In this case the pathname of the file generated from the database matches an existing file in the filesystem. The files do have different file_key values so there is no problem. Running **fsfileinfo** on the pathname will display both paths. Here is the output for the above file:

fsfileinfo
/stornext/snfs1/sncompare_testdir1/managed/popdir_3/popdir_3_1/popdir_3_1_2/popdir_3_1_1.1.007.moved_

File Information Report Sat Aug 6 12:04:08 2016
Filename:
/stornext/snfs1/sncompare_testdir1/managed/popdir_3/popdir_3_1/popdir_3_1_2/popdir_3_1_1.1.007.moved_
Stored Name:
/stornext/snfs1/sncompare_testdir1/managed/popdir_3/popdir_3_1/popdir_3_1_1/popdir_3_1_1.1.007

[OK] Pure Sparse File: '/stornext/snfs1/sncompare_testdir/managed/sparse/sparse_file'

Indicates the file is a pure sparse file. A sparse file is one that marks empty blocks, i.e. containing nothing, in the files metadata rather than allocating disk blocks. A "pure" sparse file is one that is totally empty, and has no allocated disk blocks.

[OK] 0-Length File: '/stornext/snfs1/large3/lg3_1/stornextgui.log'

Indicates the file is a 0-length file, i.e. a file using 0 bytes of disk space. Examples of this file type include a soft link or a file created by the **touch** command. No action is required.

[MX] Missing xrep: path=/stornext/snfs1/sncompare_testdir3/not_all_copies_01.txt, cpyid=4, seg=1
[BX] Bad xrep (data decode failure):
path=/stornext/snfs1/sncompare_testdir3/not_all_copies_01.txt, cpyid=3, seg=1
[BX] FILECOMP/Xrep mismatch:
path=/stornext/snfs1/sncompare_testdir3/not_all_copies_01.txt, cpyid=3, seg=1, errcnt=1, lastfield='Wrapper Length'
[BX] FILECOMP/Xrep mismatch:
path=/stornext/snfs1/sncompare_testdir2/popdir_3/popdir_3_1/popdir_3_1.1.008, cpyid=2, seg=1, errcnt=0, lastfield='Medium UUID'

The above messages are examples of External Representation, or xrep, errors. This is information stored in a file's inode about stored segment(s), if any, and should match the information in the FILECOMP database table.

The first error message indicates a missing xrep value for the specified file, copy number, and segment number. These error types are indicated with the "[MX]" notation.

The other error messages indicate mismatches between the FILECOMP database table and the inode xreps. These are indicated with the "[BX]" notation. The filename, copy number, segment, and, if known, the mismatched xrep field name are logged.

[RE] FILE_KEY NOT FOUND IN FILECOMP (fk=3374): '/stornext/snfs1/sncompare_testdir2/popdir_4/popdir_4_1/popdir_4_1.1.008'

Indicates that no entries were found in the FILECOMP database table for this file.

[RE] UNSTORED FILE (fk=3376): '/stornext/snfs1/sncompare_testdir2/popdir_4/popdir_4_1/popdir_4_1.1.010'

Indicates a file that has not been archived. This is most likely because it is being processed or has not met its store mintime.

[OK] MOVED FILE WITH HARD-LINKS FOUND: database path='%s' fslocate path='%s' (fk=%ld/pfk=%ld/links=%d)0,

Indicates the file has been moved and a hard-link was found, i.e. the inode link count is greater than 1. The database pathname, pathname returned by the **fslocate** command, file_key, parent file_key, and the number of links are reported.

[NM] Non-managed directory found: /stornext/snfs1/sncompare_testdir1/managed/unmanaged

Denote this directory is not assigned to a policy class.

[OT] Other File Type Entry Found: '/stornext/snfs1/special_files/my-char'

Indicates a file that is neither a regular file or a directory. These files are not archived by SNSM. The above example is a character file.

crw-rw-r-- 1 root root 100, 1 Apr 27 08:38 /stornext/snfs1/special_files/my-char

[UK] Unknown File: '/snfs1/Project/abce/bgd/collection/verification/missing/tracking_sheet_BGD.xlsx'

Indicates the file has a valid file key and copies, but its size is 0 and it has no blocks. The file may be recoverable by retrieving it from media.

[FX] FILE INODE KEY IS 0 (oneup=%ld): '%s'

Indicates a file where the inode file_key is zero, but the inode oneup is non-zero (See CR 78847). Also the file size is non-zero and is not store excluded. A repair is generated to update the inode key.

[RE] Incorrect DIRPATH entry: '%s'
[NR] Incorrect DIRPATH entry: '%s'

Indicates whether the parent name was not found in the DIRPATH database table. A repair operation is generated to update the DIRPATH entry with the correct path.

[NR] Truncated & Not On Tape: '/stornext/snfs1/sncompare_testdir2/popdir_4/popdir_4_1/popdir_4_1.1.008'

Indicates a file that is truncated and not archived. Even though it exists on the file system and in the database, it is not a recoverable file.

[RE] Not On Tape & May Be Truncated: '/stornext/snfs1/testDir/possiblyLostFile'

Indicates a file that has not been archived, the READ event is not set in the inode, and the number of blocks is zero. It may be recoverable by retrieving the file from the media specified in the **dm_info** xreps output, if any, and then clearing the ALL_COPIES_MADE flag. If there are no inode xreps then the file is not recoverable.

[NR] NO DM_INFO:
'/snfs1/DATA/temp/AM_J_CLIN_NUTR/2001/O'CONNOR_COMPARISON_TOTAL_ENERGY_EXPENDITU

Indicates an error occurred when running the **dm_info** command on the file. It could be due to a corrupted inode or metadata. This is a very rare error.

[FX] Mismatch between policy class copies (1) and inode copies (2):
/stornext/snfs4/testDir/testFile00

Indicates the policy class default copies and the number of inode xreps do not match. The example above shows the default copies is one, but there are two xreps in the inode.

[UK] FILE SIZE IS 0 OR STORE EXCLUDED: /stornext/snfs1/testDir/excludedFile1.txt (fk=12345678,size=8192,excluded=true)
[UK] FILE SIZE IS 0 OR STORE EXCLUDED: /stornext/snfs1/testDir/excludedFile2.txt (inode fk=0,DB fk=12345678,size=8192,excluded=true)

Both messages indicate the file will not be stored by SNSM because either its size is zero or it matches a pattern in the **/usr/adic/TSM/config/excludes.store** configuration file. The second variation also reports that the file key in the inode is 0, which means that it does not match the database value.

[UK] OTHER FILE FOUND (skipping): %s (fk=%ld/pk=%ld)

Indicates that a special file such as a character, block, or pipe was found; and these are ignored by SNSM. This error is only reported by the '-c at' test.

[OK] Directory mismatch for parent key 391 (using fslocate path)
database Path:
/stornext/snfs1/sncompare_testdir1/managed/popdir_2/popdir_2_3/popdir_2_3_1
fslocate Path:
/stornext/snfs1/sncompare_testdir1/managed/popdir_2/popdir_2_3/temp/popdir_2_3_1

Indicates a directory that has probably been renamed/moved. This is a normal condition as the database is not updated when a directory is renamed.

[ER] Directory not found: '/stornext/snfs1/sncompare_testdir2' (pkey=888882) (skipping)

Indicates the directory with that parent key was not found by fslocate, and therefore it is not processed.

[ER] Relpt not found in dirpath1 table: fk=915117

An entry like the above denotes that an entry in the CLASSDIR database table was not found in the DIRPATH database table. It probably contains no files. The number, i.e. 915117, is the file_key of the directory and can be used to retrieve the entry from the CLASSDIR database table.

Below are some example entries that could be logged in this logfile for the '-c qd' test.

Found 6 duplicate entries for path sncompare_testdir2

This indicates that there is more than one DIRPATH database table entry for the specified parent path.

No FILEINFO1 entry for file_key: 447

This indicates that there is no FILEINFO database table entry for the specified file_key but there are FILECOMP database table entries.

File_key 999999997 has no FILEINFO1 entry but mediandx 999990 has OLDMEDIA entry

This indicates for the file_key that there is no FILEINFO database table entry, there are FILECOMP database table entries, and the medium has had **fsrminfo** run on it but the database has not been cleaned by **fsclean**.

File_key 2524 missing MEDIADIR entry for mediandx: 999999

This indicates there is not a MEDIADIR table entry matching one or more of this file_key FILECOMP table table entries.

File_key 3370 has name 'KEY.3370'

This probably indicates the parent file_key was not found in the DIRPATH database table.

File_key 4365 has been removed

This indicates the FILECOMP database table entries show the file has been removed, i.e. endtime is not set to MAXINT.

TOO MANY COPIES STORED FOR FILE_KEY: 4659 (act=4/exp=2)

This indicates there are more copies stored than expected for this file.

NOT ALL COPIES STORED FOR FILE_KEY: 4660 (act=1/exp=2)

This indicates that not all copies of this file have been archived.

MULTIPLE ACTIVE VERSIONS EXIST FOR FILE_KEY: 888897

This indicates there are more than one active version for this file. Specifically this means are multiple FILECOMP table entries with a different version and with the endtime set to MAXINT.

NUMBER OF SEGMENTS IS INCORRECT FOR FILE_KEY: 888984 (copy error mask=0x1)

This indicates that the number of segments, it could be less or greater than expected, for the copy of this file is incorrect. In this particular error, the error mask indicates copy1 is in error.

PARENT KEY OF 0 FOUND FOR FILE_KEY: 773

Indicates that the parent file_key in the FILEINFO database table is 0.

PARENT KEY (372) NOT FOUND FOR FILE_KEY: 4287

Indicates that the parent file_key was not found in the DIRPATH database table.

OLDMEDIA entry with no MEDIADIR entry: 999990

Indicates the media id exists in the OLDMEDIA database table but was not found in the MEDIADIR database table.

Media E00000 Error: MEDIADIR file count mismatch: fcnt=406042/filecomp=406039

Indicates the MEDIADIR 'fcnt' value does not match the number of FILECOMP database table entries found with the media index/generation values.

Media E00015 Error: MEDIACRITERIA file count mismatch: field6=-5/filecomp=0

Indicates the MEDIACRITERIA table 'field6' value, number of segments stored on this medium, does not match the number of FILECOMP database table entries found with the media index/generation values.

Media E00004 Error: MEDIA table location field not archive or vault (Unknown)

Indicates the MEDIA database table shows this medium in a different location than an archive or vault.

Media E00002 Error: MEDIADIR/MEDIACRITERIA table status mismatch (MD=A/MC=N)

This indicates that the MSM and TSM status, whether the media is available for use or not, is different. The first letter is the TSM status: A - available, U - unavailable. The second letter is the MSM status: Y - available, N - not available.

Below are some example entries that could be logged in this logfile for the '-c qf' test.

FILE NOT IN DB: popdir_3.1.001

The specified file name was not found in the FILEINFO database table. The SQL query searches for a name AND parent file_key match in the table.

PARENT NOT IN DB (pk=-1): sncompare_testdir3/temp\\Moved\(\'\`\"\!\).moved

The specified parent path was not found in the DIRPATH database table. The SQL query searches for a path match in the table.

NOTES

The '-c fs' test does not process subdirectories which means that new subdirectories may not be processed if they have not been added to the database. However, the '-c ff' test does process all subdirectories and their contents.

Below is a list of bugs whose conditions are detected by the **sncompare** utility. Repair operations are generated to fix both problems.

78847	fsmedcopy fails to copy files with inode "key" set to 0 and "oneup" is correct
78959	Deleting file during store causes database to show endtime as active

WARNINGS

This utility should be used carefully and only under the guidance of Quantum technical support.

SEE ALSO

dm_info(1), **fsclean**(1), **fslocate**(1) **fspolicy**(1) **fsrecover**(1), **fsretrieve**(1), **fsrminfo**(1), **fsstore**(1)

NAME

sndisk_scan − Scan a managed directory

SYNOPSIS

sndisk_scan [**-t** *thread_count*] [**-f**] [**-d**] [**-u**] [**-n**] [**-i**] [**-e**] [**-P**] [**-N**] [**-M**[**-D**]] [**-S**] *managed_directory_path*

WARNINGS

This utility should be used VERY carefully and ONLY under the guidance of Quantum technical assistance.

DESCRIPTION

This utility can be used to scan any directory that is currently being managed by the Tertiary Manager software. It will work on relation points as well as any managed directory below a relation point. It will report on files and directories found under the provided directory based on the scan options provided. It can also take some actions on scanned files and directories based on the scan options provided. This functionality is not needed under normal operating conditions. It can be useful for tracking down, and possibly fixing, some issues that may exist under a managed directory. As mentioned only use with the guidance of Quantum technical assistance.

The user must be root or the utility must be owned by root with the 's' bit set.

SCAN OPTIONS

-t *thread_count*
: Use the indicated number of threads for the scanning process. The default is 4 threads and the max allowed is 32. (This should only be changed for one of two reasons: first, the host where this is to run has limited cpu resources and we don't want the scan to take too many; second, there is a need for the scan to run as quickly as possible and cpu resources are not an issue.)

-f Only consider files during the scan. (If used with the **-d** option, both files and directories are considered.) This option would be useful if only looking for non-stored files or some similar scan.

-d Only consider directories during the scan. (If used with the **-f** option, both files and directories are considered.) This option would be useful if only looking for keyless directories or some similar scan.

-u Find unmanaged files and directories. Unmanaged content here being defined as files and directories with a zero class value and/or an empty event mask. (If 'all' non-managed content is desired combine this with the **-e** option.)

-n Find non-stored files.

-i Find files and directories with inherited attributes.

-e Find files and directories which encountered errors while attempting to get the existing attributes. (The most common error that is likely to be encountered is 'no attributes present'.) If searching exclusively for files and directories with no attributes present or some other form of error, use this option. If searching for all non-managed content combine this with the **-u** option.

ACTION OPTIONS

-P Print matching entries found. (For example if using the **-n** option to find non-stored files, print the non-stored files found.)

-N Print non-matching entries found. (For example if using the **-n** option to find non-stored files, print the stored files found.)

-M Add the management attributes and events to the files and directories found.

-D When adding the management attributes and events to directories found, also set the new directory key.

-S For non-stored files found, add those files as store candidates.

managed_directory_path
: Path name of the managed directory to scan.

REPORTING

Note that the **-P** or **-N** options will alway generate a report, regardless of whether any of the other action options (**-M**, **-D**, **-S**) are specified. In the report, files that match scan criteria will be prepended by **"file:"**. Directories that match will be prepended by **"dir:"**. When reporting non-matching entries, the prepended string values will be **"file-"** and **"dir-"**. Here is a sample of the reporting output:

```
% sndisk_scan -unPN /stornext/snfs1/test1.1

SCANNING: /stornext/snfs1/test1.1   (4 threads)

file: /stornext/snfs1/test1.1/sub2/file.06
file: /stornext/snfs1/test1.1/sub2/file.11
file: /stornext/snfs1/test1.1/sub2/file.05
file: /stornext/snfs1/test1.1/sub2/file.12
 dir- /stornext/snfs1/test1.1/sub3
 dir- /stornext/snfs1/test1.1/sub2
file: /stornext/snfs1/test1.1/sub1/file.04
file: /stornext/snfs1/test1.1/sub1/file.03
file: /stornext/snfs1/test1.1/sub1/file.10
file: /stornext/snfs1/test1.1/sub1/file.09
 dir- /stornext/snfs1/test1.1/sub1
 dir- /stornext/snfs1/test1.1
scanning ... 12 inodes (0 dirs/8 files tracked) in 1 seconds at 12 inodes/sec.
Scan complete. 12 inodes (0 dirs/8 files tracked) in 1 seconds at 12 inodes/sec.
```

Note in the report summary, the number of files and directories "tracked" are those files and directories that match the scan options provided. The number of inodes is the total number of items visited.

EXAMPLES

Here are some examples of command usage.

Report all unmanaged directories under test1.1:

```
% sndisk_scan -duP /stornext/snfs1/test1.1
```

Report all non-stored files under test1.1:

```
% sndisk_scan -fnP /stornext/snfs1/test1.1
```

Report all files and directories under test1.1 missing management attributes:

```
% sndisk_scan -eP /stornext/snfs1/test1.1
```

Verify that all content under directory test1.1 is managed (roughly equivalent to fsaddrelation):

```
% sndisk_scan -unieMDS /stornext/snfs1/test1.1
```

SEE ALSO

fsaddrelation(1)

NAME

snmsm − Activates or terminates the Media Manager software and activates the system log display.

SYNOPSIS

snmsm [**-s**|**-q**|**-t**]

DESCRIPTION

The **snmsm**(l) command is issued from the command line to start or terminate the Media Manager software.

OPTIONS

-s Starts the Media Manager software in the single user mode. Only commands issued via **vsadm**(l) are executed. Client interface commands are refused.

-d Redisplays the Media Manager syslog consoles after the workstations are logged out and then logged back on.

-q Quits the Media Manager operations. The Media Manager software terminates immediately.

-t Terminates the Media Manager operations. The Media Manager software terminates gracefully. Outstanding commands are cancelled and any commands awaiting status are allowed to complete (within a certain time period).

EXIT STATUS

0 Command completed successfully - The Media Manager software started or completed normally.

1 Command did not complete successfully - The Media Manager software did not start or complete as expected.

EXAMPLES

Successful Media Manager system start-up.

snmsm

Requests the Media Manager software to start.

Output returned:

```
Media Manager Version 4.2.0 for Linux (Kernel:2618 OS:RHEL5) -- Copyright (C)   1

Initiating Media Manager software start up

Setup environment variables ok

Starting up process server .... Done

Process server started ok

Starting up Media Manager server processes ..... Done

Server processes started ok

Starting up Media Manager system processes .....Done

System processes started ok

Media Manager software start up completed
```

SEE ALSO

vsadm(l)

NAME

snnas_usage − Produce a report of tertiary storage usage per-share per-user, or per-share per-user per-group

SYNOPSIS

snnas_usage [-g]

DESCRIPTION

The **snnas_usage** program calls the **snnas_usage_engine** program for each StorNext managed file-system device to create a report file of tertiary storage usage per-share per-user for each file system. When the **g** option is used, the report is per-share per-user per-group. Input files are automatically generated per file system.

The SNNAS feature must be configured to run on both MDCs in HA mode. Other configurations, including SNNAS Gateways, are not supported by this program because it cannot access those SNNAS servers to collect the necessary share path information.

Share paths must be unique across share names and must not be nested. When duplicate or nested paths occur, a warning is printed and these paths are discarded in a consistent manner for the purposes of reporting usage.

Each file-system device has an output file in comma-separated-values format (CSV) with the following path name:

```
/var/shareUsageReports/<timestamp>/<mount name>/<mount name>_<timestamp>
```

For example:

```
/var/shareUsageReports/20151217111117/stornext_snfs1/stornext_snfs1_2015121711111
```

The output files from previous runs may be on either MDC since the path is not on a shared file system.

The CSV output file is suitable for importing into a spreadsheet where a pivot-table function can be used to sum the total space per user ID, per share, or per group if the group option is selected. The columns of the CSV output file are as follows:

```
Path Name, Share Name, Numeric User ID, Size in Bytes
or
Path Name, Share Name, Numeric User ID, Numeric Group ID, Size in Bytes
```

The digits of the timestamp are: year(4), month(2), day(2), hour(2), minute(2), second(2).

Note: Location information, User ID, and Group ID for a copy is recorded when the copy is made. When a file's location is changed to a different share or the User or Group IDs are changed, the **snnas_usage** report will continue to report the file's usage in the original share, UID, and GID until the file is changed and a new copy is made. The share-pathname discrepancy can be viewed with the **fsfileinfo** command in its *Stored Name* field, but the UID and GID discrepancies are not reported by **fsfileinfo**.

FILES

/usr/local/quantum/etc/brand.txt
File contains the *controller.port* value for configuring the TCP/IP port of the SNNAS controller.

SEE ALSO

snnas_usage_engine(1) **fsfileinfo**(1)

NAME

snnas_usage_engine − Process data for a tertiary-storage usage report

SYNOPSIS

snnas_usage_engine [**-dgh**] **-f** *join_table_file* **-m** *mount_path* **-o** *output_file* **-p** *dirpath_file* **-s** *JSON_shares_file*

DESCRIPTION

The **snnas_usage_engine** program generates a comma-separated-values (CSV) report of tertiary-storage usage per-share per-user for one file system. It is intended to be called by the **snnas_usage** script, which generates the input files per file system.

Share paths must be unique across share names and must not be nested. When duplicate or nested paths occur, a warning is printed and these paths are discarded in a consistent manner for the purposes of reporting usage.

OPTIONS

-d Debug mode. This generates information per share and per file, so the information can be very large on production systems.

−f *join_table_file*
CSV file with the results of joining the fileinfo and filecomp tables for one device. The columns are: parent file key, file key, numerical UID, size in bytes, and numerical GID for one segment. There is no header row.

-g Report group ID (GID) information in the output file.

-h Displays the usage of the command.

−m *mount_path*
Path of the mount point for one device. The mount path must not end with a slash (/).

−o *output_file*
CSV report of tertiary storage usage per-share per-user for one file system. The columns are: share pathname, share name, numerical user ID, size in bytes. When the **-g** option is specified, the columns are: share pathname, share name, numerical user ID, numerical group ID, size in bytes. There is no header row. The size is bytes of non-compressed data, so the value may need to be prorated for storage destinations that use compression. Alternatively, the charge-back rate could simply be scaled to reflect the cost of compressed storage. The non-share storage used for the device is accumulated into a pseudo share with path name *non-share files*, and name *non-share files*.

−p *dirpath_file*
CSV rows of the dirpath table for one device. The columns are: parent file key, path below the mount point. There is no header row.

−s *JSON_shares_file*
JSON formatted report of the configured shares from the SNNAS server.

EXIT VALUES

Returns zero on success and non-zero on failure.

SEE ALSO

snnas_usage(1)

NAME

snrestore − Restore elements of SNSM system.

SYNOPSIS

snrestore [**-h**] [**-e**] [**-r** *extract_dir*] [**-p** *temp_path*] [**-m** [*file_system_name*]] [**-d**] [**-c**] [**-a** *serverauth*]
[[**-U** *username* **-P** *password*] | [**-A** *CAP_agency* **-M** *CAP_mission*]] [**-S** *signingtype*]
[**-Y** *authenticationtype*] [**-N** *role*] [**-O** *storageclass*] [**-Z** *authentication_endpoint*]
[**-H** *CAP_hostport*] [**-C** *certfile* | **-R** *certpath*] [**-L** *clientcertfile*] [**-k** *clientkeyfile*] [**-w** *clientkeypass*]

DESCRIPTION

This command is used to restore elements of SNSM on backup media. Configuration information, database information, and file system metadata can be restored through this utility. Each one of the three elements of a restore package can be restored individually or all together. If a backup manifest file is present on the system, information about known backups are displayed by the utility. The user specifies the backup identifier and supplies the media required for the restore. These two pieces of information are supplied by the backup manifest or the backup email notification. All required media must be present to be able to restore the system.

The first part of the restore process involves retrieving the archived backup files from media. The retrieved backup files are stored in a directory specified by the **-p** option or in the default directory. The default is /usr/adic/TSM/tmp. Only selected files are retrieved from media. The **-r** option specifies a directory which contains backup files. If this option is specified, file retrieval is skipped and the restore process uses the files present in the directory specified. The **-e** option allows the user to only extract backup files from media without following through with the rest of the restore processing. The files are stored in the temporary directory or the directory specified by the **-p** option.

The user can select specific components they are interested in. The **-c** option selects configuration information. The **-d** option specifies the database component. The **-m** option selects the file system metadata. A specific file system can be specified by providing the optional file system name.

The user will be queried for different pieces of information during the restore. The user will first be queried for the backup identifier. This backup identifier will be used to select files from the tape device, storage disk or Object Storage media. The user will also be queried for the copy number. In addition, the user will be queried for a scsi device path, storage disk path or Object Storage URL.

Once files have been retrieved, they can be processed. The system software is shut down and the specified components are restored. In HA environments, the HaShared file system needs to be mounted prior to restoring components. Configuration information is restored for each of the specified software components. The database is restored from the most recent backup associated with the backup identifier. File system metadata is restored to disk and the previous metadata file is archived. Once restore processing is complete, the software remains shut down. Manual intervention is needed to complete the restore process and to bring the system software online.

OPTIONS

-p *temporary_path*
: Specifies a temporary storage directory for retrieved files. The default is /usr/adic/TSM/tmp. If used in conjunction with the **-r** option, *temporary_path* is used as a temporary directory in which to expand and rebuild/merge the files required for the restore step.

-e
: Extract backup files to a temporary directory. No further processing is done. The files are retrieved from the media and stored in the temporary storage area.

-r *restore_directory_path*
: Uses files contained with this directory for the restore process. No files are retrieved from media.

-c
: Restore all software configuration information. In HA environments, the HaShared file system must be mounted.

-d
: Restore database to /usr/adic/mysql/db. In HA environments, /usr/adic/mysql/db is a symbolic link to /usr/adic/HAM/shared/mysql/db, and requires that the HaShared file system is mounted.

-m [*file_system_name*]

Restores file system metadata information for all file systems or the selected optional file system. The metadata files are restored to the /usr/adic/database/mdarchives directory. In HA environments, /usr/adic/database/mdarchives is a symbolic link to /usr/adic/HAM/shared/database/mdarchives, and requires that the HaShared file system is mounted.

-mj [*file_system_name*]

DEPRECATED: Previously this option would restore file system metadata journals for all file systems or the optional selected file system name.

-a *serverauth*

The SSL certificate authentication mode to use when retrieving from Object Storage media. **0** indicates that no SSL certificate validation is to be done. **1** indicates to check only the peer. **2** is the default operation and indicates that both the host and peer are to be validated.

-C *cacert*

SSL certificate to use with HTTPS restore from Object Storage media.

-R *capath*

Directory of SSL certificates as prepared by **c_rehash** to use with HTTPS restore from Object Storage media.

-L *clientcertfile*

The location of the X.509 client certificate installed on the system for the CAP server to authenticate. Note that the certificate should be in PEM format and is required for CAP authentication.

-k *clientkeyfile*

Specifies the location of the client private key. This option is required for CAP authentication if the client private key is kept in a separate file rather than included as part of the X.509 client certificate file. This option is also required to authenticate requests to Google Cloud Storage. The specified Google private key file must be in JSON format.

-w *clientkeypass*

Client private key passphrase. This is required if the private key is protected by a passphrase.

-U *username* **-P** *password*

Username and password to access the URL if necessary. These are required if the Object Storage media is using S3 or Azure Blob REST API from the following providers:

AWS (standard and STS)
AZURE
GOOGLE
LATTUS
S3COMPATIBLE

-A *CAP_agency* **-M** *CAP_mission*

The agency and mission associated with the target C2S account. These options are valid only for the CAP authentication type.

-Y *authentication_type*

An authentication type is required for all Object Storage media except for AXR and GOOGLE. The following authentication types are supported:

STANDARD
STS_PUBLIC
STS_GOVCLOUD
CAP

If this option is not specified, the authentication type will be set to **STANDARD**.

The **STANDARD** type applies to all media and authenticates with a user name and password for Object Storage access. For S3 compatible media the user name and password are the Access Key Id and Secret Access Key. For Azure media, the user name and password are the Storage Account

Name and Storage Access Key.

The **STS_PUBLIC** type uses the AWS Security Token Service (STS) to obtain temporary credentials for access to Object Storage in the AWS public cloud. This authentication type applies only to AWS media and requires a role, username and password be specified.

The **STS_GOVCLOUD** type uses the AWS Security Token Service (STS) to obtain temporary credentials for access to Object Storage in the AWS GovCloud. This authentication type applies only to AWS media and requires a role, username and password be specified.

The **CAP** type uses the AWS Commercial Cloud Services (C2S) Access Portal (CAP) to obtain temporary credentials for access to Object Storage in the AWS Private Cloud for the Federal Government. This authentication type applies only to AWS media and requires a role, CAP mission, and CAP agency be specified.

-N *role* For **STS_PUBLIC** and **STS_GOVCLOUD** authentication, use the Amazon Resource Name (ARN) of the role to assume. For **CAP** authentication, use the Identity and Access Management (IAM) role associated with the target C2S account. A role is required by AWS STS and the CAP servers to obtain temporary credentials. This option is valid only with the **STS_PUBLIC**, **STS_GOVCLOUD**, and **CAP** authentication types.

-S *signing_type*

Used to specify the signing type for the requests sent to the Object Storage and/or authentication server. The following signing types are supported:

V2
V4
AZURE

The default is **V4** for AWS and S3 compatible media, **V2** for GOOGLES3 and Lattus S3 media, and **AZURE** for Azure media. This option is not valid for AXR and Google media. When configuring V4, chunked mode is preferable and should be used if supported by the object storage system. For object storage which does not support chunked mode, such as AWS Snowball, you must specify the full payload mode signing option, **-J**, for the object storage bucket.

-O *storageclass*

Defines the storage class for AWS, S3COMPAT, AZURE and GOOGLES3 media only. The following storage classes are supported:

azure_block_blob Azure Block Blob storage class. Blobs in this storage class are in either Cool or Hot tier. Cool or Hot tier is an attribute of the container, defined by the user during configuration of the container on Azure. Note: If the Azure lifecycle policies are configured to use the archive tier, then the **azure_block_blob_archive** storage class must be used instead of this one, otherwise files will not be retrievable from the archive tier.

azure_block_blob_archive Data written to this tier must be rehydrated before it can be read. This storage class utilizes the Azure archive tier. It should be used when Azure lifecycle policies are configured to use the archive tier. Note: this tier feature only applies to Azure Blob storage and General Purpose v2 (GPv2) accounts. General Purpose v1 (GPv1) accounts do not support tiering. For more information, check the Azure documentation on Blob storage tiers.

glacier S3 Glacier storage class using standard retrievals to access data from Glacier storage. When restoring from AWS S3 Glacier using standard retrievals, data can typically be accessed within 3-5 hours.

glacier_exp	S3 Glacier storage class using expedited retrievals to access data from Glacier storage. When restoring from AWS S3 Glacier using expedited retrievals, data can typically be accessed within 5 minutes.
glacier_bulk	S3 Glacier storage class using bulk retrievals to access data from Glacier storage. When restoring from AWS S3 Glacier using bulk retrievals, data can typically be accessed within 5-12 hours.
glacier_deep	S3 Glacier storage class using standard retrievals to access data from Glacier Deep Archive storage. When restoring from AWS S3 Glacier Deep Archive using standard retrievals, data can typically be accessed within 12 hours.
glacier_deep_exp	S3 Glacier storage class using the expedited retrieval option to access data in Glacier Deep Archive storage. This option is not supported by AWS.
glacier_deep_bulk	S3 Glacier storage class using the bulk retrieval option to access data from Glacier Deep Archive storage. When restoring from AWS S3 Glacier Deep Archive using bulk retrievals, data can typically be accessed within 48 hours.
standard	S3 Standard storage class
standard_ia	S3 Infrequent Access storage class

By default, the storage class is set to **standard** for AWS, S3COMPAT, and GOOGLES3 media, and set to **azure_block_blob** for Azure media. **standard** is the only supported storage class for GOOGLES3.

Note, the cost of retrieval from an AWS Glacier storage class will vary by retrieval time, with the quickest retrieval type having the highest cost.

Please refer to provider documentation for detailed information on supported storage classes, including cost and retrieval times.

-H *CAP_hostport*
Specifies the connection endpoint for the CAP server.

-Z *authentication_endpoint*
Specifies an authentication endpoint which overrides the default endpoint for the public or GovCloud STS server. This option is valid only with the **STS_PUBLIC** and **STS_GOVCLOUD** authentication types. If the endpoint is newly supported by Amazon, support within the Quantum storage system can be enabled by adding this endpoint along with its region name to the AWS regions configuration file (*/usr/cvfs/config/awsregions.json*).

-h help option shows usage

EXIT STATUS

0 when successful and 1 after any failure.

EXAMPLES

Restore all components from backup located on tape media

```
snrestore
```

Restore database from backup located on tape media and use /tmp as the temporary directory

```
snrestore -d -p /tmp
```

Restore from backup files located in the /backup directory

```
snrestore -r /backup
```

Restore configuration components

```
snrestore -c
```

Restore file system data for snfs1 file system

```
snrestore -m snfs1
```

Extract backups to a working directory, and restore from the working directory using a separate temporary directory for backup file expansion and merge.

```
mkdir /home/ejr/dump.test
snrestore -e -p /home/ejr/dump.test
mkdir /tmp/dump.work
snrestore -r /home/ejr/dump.test -p /tmp/dump.work
```

FILES

/usr/adic/TSM/internal/status_dir/snbackup_manifest
/usr/adic/TSM/internal/status_dir/device_manifest
/usr/adic/TSM/internal/status_dir/snobjs_manifest
/usr/adic/mysql/snbackup_manifest
/usr/adic/mysql/device_manifest
/usr/adic/mysql/snobjs_manifest

SEE ALSO

snbackup(1), **snbkpreport**(1)

NAME

sntsm − Activate or terminate Tertiary Manager software.

SYNOPSIS

sntsm [-f]

sntsm -c [-f] [**-y**]

sntsm -t [**force**] [**-y**]

sntsm -s

sntsm -S [*storage type* **]** [**-F** *type*]

sntsm -K [-f]

DESCRIPTION

The **sntsm** command activates or terminates the Tertiary Manager software. This command can also be used to provide capacity information for storage types related to the Tertiary Manager software.

OPTIONS

-c Initiates a contingency start of the Tertiary Manager software. This is recommended after an abnormal shutdown of the software. A contingency start reinitializes a subset of the Tertiary Manager system parameters and removes recovery processing files. A contingency start allows the Tertiary Manager software to return to a quiescent processing state when a normal start fails to do so.

-t force Initiates a graceful termination of the Tertiary Manager software. The Tertiary Manager software completes all requests that are currently being processed and rejects any requests received after the system termination request. If "force" argument is given, a force termination is initiated.

-y Answers yes to all command prompts.

-s This option indicates if startup of the Tertiary Manager software is allowed based on the current state of licenses.

-S [*storage type*]
Display a summary report for licensed storage types. Optionally a specific storage type can be specified. Valid types: **tape**, **sdisk**, **lattus**, **public_cloud**, and **private_cloud.**

-K Synchronize file key use so that file keys are tracked and new file keys are generated correctly. The **-K** option can be used whether the Tertiary Manager system is up or down, and does not change that state.

-f At startup, file key sychronization is performed. If there are issues, the "-f" option will allow startup to continue even if database errors occur during the synchronization. Otherwise, database errors during the synchronization will result in command failure.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default) or **JSON**.

TEXT is the "legacy" textual format.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application. See http://json.org for more information.

LICENSE EXPIRATION

When one perpetual license expires, access to all perpetual licensed features is suspended with the exception of the "maintenance" and "proxy" licenses. When a subscription license expires, access to the licensed feature is still allowed. License expiration times are shown in the report generated with the **-S** option. Expired licenses are annotated with an asterisk in the output.

EXIT STATUS

Exit codes for the **sntsm**(1) command with the **-s** option are:

0 The current license status allows the software to start

1 The current license status does not allow the software to start

EXAMPLES

When the command is used with the **-S** option a storage capacity summary report is generated. An example:

Feature **manager** is properly licensed for 10 terabytes and has no expiration.
Feature **sdisk** is properly licensed and has no expiration.
Lattus is properly licensed with a subscription license for 3 terabytes expiring 12/31/2017 and has reached capacity.
Public cloud storage is properly licensed with a subscription license for 2 terabytes expiring 12/31/2017.
Privatecloud storage is properly licensed with a subscription license for 1 terabyte which expired 12/31/2016.

```
% sntsm -S
------------------------------------------------------------------------------
Storage Type Capacity Summary Report                     Sun Jan  1 08:00:00 201

---------------------License---------------------  ----------Storage------------
Name               Expiration           Capacity   Type          Capacity  % Used
------------------ -------------------- ---------  ------------- --------- ------
manager            never                 10.0 TiB  tape           43.3 GiB     0.
                                                   sdisk          11.6 GiB     0.
flex_lattus        2017-11-04 23:59:59    3.0 TiB  lattus          3.0 TiB   100.
flex_public_cloud  2017-12-31 23:59:59    2.0 TiB  public_cloud    0.0   B     0.
flex_private_cloud 2016-12-31 23:59:59*   1.0 TiB  private_cloud   0.0   B     0.
```

SEE ALSO

fsconfig(1), **snlicense**(8)

NAME

sn_fpt_convert − Convert Storage Manager Database to File-per-table

SYNOPSIS

sn_fpt_convert -c [**-F**] [**-C** *directory*] [**-p** *port*]

sn_fpt_convert -s [**-F**] [**-C** *directory*] [**-p** *port*]

sn_fpt_convert -a [**-F**] [**-C** *directory*] [**-p** *port*]

sn_fpt_convert -G [**-C** *directory*] [**-p** *port*]

WARNING

Immediately prior to running the conversion, you must run a full backup to ensure that the database and its contents are protected in the event of a failure. You must make a valid and up to date full backup with no RAS tickets before proceeding with this procedure.

DESCRIPTION

This script is used to convert a Storage Manager MySQL database from global to per-table data using replication. It is intended to be used once during or prior to the upgrade process to convert the Storage Manager database to use one file-per-table. Systems installed using StorNext 5.2 or later will already have file-per-table enabled and do not need to run this conversion.

The script encapsulates the conversion process in 2 major parts:

1) Create (-c):

Start the conversion process and create a Storage Manager's MySQL database with per-table data
ex. sn_fpt_convert -c

2) Switch (-s):

Shutdown Storage Manager to perform a switchover to use the new database, then restart Storage Manager using the converted database
ex. sn_fpt_convert -s

Storage Manager must be restarted two times during the process, once during Create (part 1) to apply replication settings, and again during Switch (part 2) to perform the final change to use converted database files. The script will prompt the user prior to restarting Storage Manager. One may use the -a option to combine both parts and perform the conversion into a single command:

ex. sn_fpt_convert -a

The script will log its output to:

/usr/adic/mysql/logs/sn_mysql_fpt_log.out

SYSTEM STATES

The system will be in one of several states during the conversion process:

State 1: Unconverted, the state prior to running the script

State 2: State after the Create part completes. System is still unconverted, the slave has been created and is running.

State 3: State when the system has failed over or been restarted in state 2. System is still unconverted, the slave has been created, but is not running. Slave will be restarted when Switch is run (sn_fpt_convert -s).

State 4: State after Switch completes. System is converted. Slave instance is not running

Once the Create part (sn_fpt_convert -c) has completed a new Storage Manager MySQL Database slave instance will be created and running (state 2). The slave instance will remain running until the Switch part is run or the system is restarted or failed over (state 3).

Once the system has a slave created, Switch part (sn_fpt_convert -s) may be run. If the system is in state 3, Switch will restart the previously created slave instance, returning it state 2 before continuing. When Switch returns successfully, the conversion is complete and the slave instance will be shut down (state 4).

TO DETERMINE IF A SYSTEM HAS BEEN CONVERTED

Run sn_fpt_convert -c. Running Create on a converted system (state 4) will produce a warning message. The -F option may be used to re-run the conversion on an already converted system. Additionally, sn_fpt_convert -G will produce a warning message on converted systems as well.

RESTART / FAILURE RECOVERY

If an involuntary failover, I/O error, out of space error, Invalid slave replication, or system restart occurs at any time during the Create step:

sn_fpt_convert -c

or early in the Switch step (prior to actual switchover):

sn_fpt_convert -s

Move the working directory, start Storage Manager as needed, and re-run the Create part. The default working directory is different for HA and standalone installs. Storage Manager may need to be restarted after sn_fpt_convert error. The following needs to be run to restart this step for standalone:

mv /usr/adic/mysql/convert_mysql /usr/adic/mysql/convert_mysql.old
Start Storage Manager as needed
sn_fpt_convert -c

For a HA system the following needs to be run on the MDC:

mv /usr/adic/HAM/shared/convert_mysql /usr/adic/HAM/shared/convert_mysql.old
Start Storage Manager as needed
sn_fpt_convert -c

If an involuntary failover occurs during Switch:

sn_fpt_convert -s

Then the following needs to be run on the new MDC to restart this step:

sn_fpt_convert -s

OPTIONS

The default options for the directory location are sufficient for most systems; however, for systems with very limited disk space, the optional arguments provide a way to specify an alternate working directory location for the conversion process to use. Changing the default directory location may increase the overall conversion time and Storage Manager down-time due to an increased data transfer time.

For a successful conversion, identical optional arguments should be used for both the create and switch steps. Failure to do so will result in undesirable behavior.

-c Create and start a new Storage Manager database, the slave instance

-s Switch over Storage Manager to use the slave instance

-a Perform entire conversion at once (-c and -s in a single step)

-G Display database size information relevant to the conversion. Prints only, does not perform any conversion steps

-C *directory*
Working directory for the conversion process. The slave MySQL instance will be created under this directory
default for HA : /usr/adic/HAM/shared
default for non-HA : /usr/adic/mysql

-p *port* Port to use for slave instance, This only needs to change if the port is already in use
default : 3308

-F Skip confirmation prompt

EXIT STATUS

Exit codes for the **sn_fpt_convert**(1) command are:

0 The conversion step was successful

1 The conversion step was not successful, output will contain additional details

On completion, a message indicating a successful run will be printed to standard out and log file. If this message is not present, the conversion did not complete successfully.

EXAMPLES

To run the conversion in two parts with a confirmation prompt prior to Storage Manager restarts:

```
sn_fpt_convert -c
sn_fpt_convert -s
```

This approach is recommended for large systems where creating a 2nd instance will be longer and controlling when Storage Manager restarts is desired.

To run the conversion in a single step and no confirmation prompt prior to Storage Manager restarts:

```
sn_fpt_convert -a -F
```

This approach is recommend for smaller installs where the conversion time is short and both Storage Manager restarts will occur shortly after invoking the script.

NAME

vsarchiveconfig − Configures an archive for the Media Manager system.

SYNOPSIS

vsarchiveconfig -a stage -n *arch_name* **-o** *arch_mode*
-m *medtype~#bin...*
-k *medtype~actionmode...* [**-w** *medtype~low^high...*]

vsarchiveconfig -a scsi -n *arch_name* **-o** *arch_mode* **-T** *scsi_device*
[**-m** *medtype~#bin...*] [**-e** *medtype~eifId...*]
[**-k** *medtype~actionmode...*] [**-w** *medtype~low^high...*]
[**-i** *medtype~mediaclass...*] [**-c** *medtype...*]

vsarchiveconfig -a acsls -s *acs_num* **-n** *arch_name* **-o** *arch_mode* **-T** *acsls_Server*
[**-k** *medtype~actionmode...*] [**-w** *medtype~low^high...*]
[**-i** *medtype~mediaclass...*] [**-c** *medtype...*]

vsarchiveconfig -u stage -n *arch_name* [**-p** *new_arch_name*] [**-o** *arch_mode*]
[**-m** *medtype~#bin...*]
[**-k** *medtype~actionmode...*] [**-w** *medtype~low^high...*] [**-x** *medtype...*]

vsarchiveconfig -u scsi -n *arch_name* [**-p** *new_arch_name*] [**-o** *arch_mode*] [**-T** *scsi_device*]
[**-m** *medtype~#bin...*] [**-e** *medtype~eifId...*]
[**-k** *medtype~actionmode...*] [**-w** *medtype~low^high...*]
[**-i** *medtype~mediaclass...*] [**-c** *medtype...*]

vsarchiveconfig -u acsls -n *arch_name* [**-p** *new_arch_name*] [**-o** *arch_mode*] [**-T** *acsls_Server*]
[**-k** *medtype~actionmode...*] [**-w** *medtype~low^high...*]
[**-i** *medtype~mediaclass...*] [**-c** *medtype...*]

vsarchiveconfig -d *arch_type* **-n** *arch_name* [**-F** *media_actionstate*]

DESCRIPTION

vsarchiveconfig(l) is issued from the command line to request execution of the Media Manager Archive Configure command. The Media Manager software must be active in order to run this command.

In order to use an archive many actions must be performed. First the archive must configured with this command, then media classes created and associated with the archive, then drives configured and associated with the archive, and finally media added to the archive.

OPTIONS

arch_type

valid archive types are:
- stage (vault archive)
- scsi
- acsls

-a *arch_type*

This will add an archive of the specified type.

-u *arch_type*

The update option allows an archive of the specified type to be reconfigured. When updating the archive configuration the archive will become unavailable until the update is complete. This option will result in the software processes that are unique to the specified archive to be restarted.

-d *arch_type*

This will delete an archive of the specified type. By default, an archive can only be deleted when there are no media associated with it. However, it can also be removed if media are contained in it. The ability to delete an archive with media in it depends on the value of the DELETE_ARCHIVE_WITH_MEDIA environment variable. When set to **N**, any archive containing media can not be deleted. When set to **Y**, the value of the DELETE_ARCHIVE_MEDIA_ACTIONSTATE

environment variable determines how media are to be handled when a populated archive is deleted. These variables can be adjusted in the */usr/adic/MSM/config/envvar.config* file. The **-F** option can also be used as an alternative to the environment variables.

-n *arch_name*

This is the unique user defined name for the archive. Valid archive names may contain up to 16 characters, including spaces. Leading and trailing spaces are not permitted.

-p *new_arch_name*

This is the new unique user defined name to use for the archive. Valid names may contain up to 16 characters, including spaces. Leading and trailing spaces are not permitted.

-o *arch_mode*

The archive mode parameter indicates whether a human is available to perform media enter and eject operations for the archive. There are two archive mode settings:

a - attended

n - unattended

This option impacts the behavior of those commands that require human intervention. These are the Check-out Media, Export Media, Move Media, and (sometimes) Mount Media commands.

-s *acs_num*

For **acsls** type archives only. The ACSLS server's acs number of the library being configured. This number cannot be modified once set.

-T *scsi_device | acsls_Server*

For **scsi** archive types this is the scsi device name for the archive. Note that it is just the device name and not the complete path. For example: *sg123*

For **acsls** archive types this is the IP address of the ACSLS server.

-m *medtype~#bin*[,*medtype~#bin*]...

This indicates the number of bins available for the specified media types. This is required for **stage** archive types. For other archive types it can be computed automatically. Multiple media type/bin values can be specified when separated with a comma and no embedded spaces are used.

-e *medtype~eifId*[,*medtype~eifId*]...

This specifies the eject/insert facility identifier for each media type. This option usually does not need to be specified. Multiple media type/eifId values can be specified when separated with a comma and no embedded spaces are used.

-k *medtype~actionmode*[,*medtype~actionmode*]...

The action mode parameter specifies what action, if any, that is initiated when the number of media of the specified media type reaches the calculated high mark threshold or drops to the calculated low mark threshold. The possible action mode values are:

o - none. This mode indicates no action is be initiated and is the typical setting.

n - notify. The mode will generate a notification to the system syslog.

Multiple media type/action mode values can be specified when separated with a comma and no embedded spaces are used.

-w *medtype~low^high*[,*medtype~low^high*]...

This specifies the low and high watermark parameters for the specified media types. The action performed when watermark thresholds have been hit is defined by the **-k** option. Multiple media type/watermark values can be specified when separated with a comma and no embedded spaces are used. The watermark values are a number between 0 and 100. The low watermark must be less than the high watermark. These values default to 0 when not specified.

-i *medtype~mediaclass*[,*medtype~mediaclass*]...

This enables the automatic import capability for the specified media types. When the Audit command detects the physical presence of media that are unknown to Media Manager, then the media are automatically assigned to the specified mediaclass. Multiple media type/media class values can be specified when separated with a comma and no embedded spaces are used.

-c *medtype*[,*medtype*]...

This enables the automatic check-in capability for the specified media types. This applies to media that have been checked out with Check Out command. There are two operations that can detect the physical presence of checked-out media: the Audit and Enter commands. When the Audit or Enter commands detect the physical presence of media that are checked-out, then the media are automatically checked-in and remain assigned to the mediaclass which they were associated when they were checked-out. Multiple media type values can be specified when separated with a comma and no embedded spaces are used.

-x *medtype*[,*medtype*]...

This specifies the media types to remove from the archive. It is only valid for **stage** archive types. The specified media type can only be removed if there are no archive media classes for the media type associated with the archive. The media class associations can be viewed by using the **-c** option with the Archive Query command. Multiple media type values can be specified when separated with a comma and no embedded spaces are used.

-F *media_actionstate*

This will allow an archive to be deleted when it contains media. It has the same effect as if the DELETE_ARCHIVE_WITH_MEDIA environment variable is set to **Y**. The media action state value determines how the media are treated and behaves just like the DELETE_ARCHIVE_MEDIA_ACTIONSTATE environment variable. The possible **media_actionstate** values are:

i - intransit. This puts media into the intransit state.

e - export. This will cause media to be exported.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255

The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

Create an attended SCSI archive called lib1 that has a device path of /dev/sg5, supports LTO and LTO Worm media, and will import media into the LTO_ADDBLANK and LTOW_ADDBLANK media classes.

vsarchiveconfig -a scsi **-n** lib1 **-o** a **-T** /dev/sg5 **-k** LTO˜o,LTOW˜o **-i** LTO˜F0_LTO_ADDBLANK,LTOW˜F0_LTOW_ADDBLANK

Create an attended ACSLS archive called liba that uses an ACSLS server located at 10.20.230.123, supports LTO and LTO Worm media, and will import media into the LTO_ADDBLANK and LTOW_ADDBLANK media classes.

vsarchiveconfig -a acsls **-n** liba **-o** a **-T** 10.20.230.123 **-k** LTO˜o,LTOW˜o **-i** LTO˜F0_LTO_ADDBLANK,LTOW˜F0_LTOW_ADDBLANK

NOTES

The valid media type values that this command uses can be listed with the Media Type Query command.

SEE ALSO

vsarchiveqry(l), **vsmedtypeqry**(l), **vsenter**(l), **vscheckout**(l), **vsexport**(l), **vsmove**(l), **vsmount**(l), **vsaudit**(l)

NAME

vsarchiveeject − Ejects media out of an archive to be entered into another archive.

SYNOPSIS

vsarchiveeject [**-Ivh**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] [**-s** *portID*] [**-F** *type*] *mediaID...*

DESCRIPTION

vsarchiveeject(l) is issued from the command line to request execution of the Media Manager Archive Eject command.

A client uses the Archive Eject command to physically remove media from an archive. Media that are ejected are still known by the software, but are unavailable for client allocation. Media ejection does not require operator intervention.

Upon receipt of an Archive Eject request, the software verifies the specified media exists in an archive. The current archive of each specified medium is commanded to physically eject the medium from the archive system. The medium becomes an Intransit medium that can be entered into another archive with the Media Manager Archive Enter command. Archives can be either attended or unattended.

OPTIONS

mediaID...

Specifies a list of one or more media to be moved.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

The media ID must be known to the software and exist in an archive. Multiple media IDs can span archives, as long as the portID is not required for the archives, or if it is required matches is the same for all of the archives that it is required for.

Media not in the archive (in transit, checked out) cannot be ejected. Mounted media cannot be ejected, it must be dismounted first (see vsdismount). Media cannot be ejected from an offline archive. Media cannot be ejected if it is being checked in or imported. Media that has been exported (see vsexport) prior to calling this command will be physically ejected and removed from the system, and the corresponding Library Operator Interface request will be cleared.

-v Indicates that verbose output is desired.

If **-v** is specified, status is returned on all media specified in the Archive Eject request.

If **-v** is not specified, status is returned on only those media that were not successfully processed.

-s *portID*

Specifies the import/export port the media is ejected to (if applicable).

If the portID is not specified for archives of type scsi and acsls then a default of 0,0,0,0 will be tried. **vsarchiveqry -s** can be used to obtain the portID for an archive.

If the portID is specified for stage archives, it is ignored.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

EXIT STATUS

0 The **vsarchiveeject** command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Archive Eject request, verbose option specified.

vsarchiveeject MED016 MED023 MED044 MED048 **-v**

Requests the Media Manager software to eject media MED016, MED023, MED044, and MED048 out of an archive; and to return status on every specified medium.

Output returned:

```
Archive Eject 4 of 4 media was successful

   Media [MED016]      no error

   Media [MED028]      no error
```

```
Media [MED043]      no error

Media [MED048]      no error
```

Successful Archive Eject request, verbose option not specified.

vsarchiveeject MED013 MED016 MED028 MED031

Requests the Media Manager software to eject media MED013, MED016, MED028, and MED028 from their current archives.

Output returned:

```
Archive eject 4 of 4 media was successful.
```

Error(s) with verbose option specified (nonexistent media)

vsarchiveeject MED013 MED016 MED022 MED034 **-v**

Requests the Media Manager software to eject media MED013, MED016, MED022, and MED034 and to return status on every specified medium.

Output returned:

```
Archive eject 1 of 4 media was successful

Error VOL024: error in the list

   Media [MED013]   no error

   Media [MED016]   item not found

   Media [MED022]   item not found

   Media [MED034]   item not found
```

Error(s) with verbose option not specified

vsarchiveeject MED003 MED004 MED013 MED028 MED033 MED043

Requests the Media Manager software to eject media MED003, MED004, MED013, MED028, MED033, and MED043.

Output returned:

```
Archive eject 3 of 6 media was successful

Error VOL024: error in the list

   Media [MED003]   item not found

   Media [MED004]   item not found

   Media [MED033]   item not found
```

Unsuccessful Archive Eject request

NOTES

Media that is allocated to an Archive Eject request is not available for other allocation until an enter completes.

A pending Archive Eject request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (control-c).

ERRORS

VOL206 archive type not supported - Only scsi, acsls, and vault archive types are supported for this command.

SEE ALSO

vsarchiveenter(l), **vsexport**(l), **vsmove**(l), **vsarchiveqry**(l)

NAME

vsarchiveenter − Enters media that has been ejected out of an archive into another archive.

SYNOPSIS

vsarchiveenter [**-Ihv**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] [**-s** *portID*] [**-F** *type*] **-a** *archivename* *mediaID...*

DESCRIPTION

vsarchiveenter(l) is issued from the command line to request execution of the Media Manager Archive Enter command.

A client uses the Archive Enter command to physically enter ejected media into an archive. Media that was ejected are still known by the software, but are unavailable for client allocation. Media entering does not require operator intervention.

Upon receipt of an Archive Enter request, the software verifies the specified media exists and the archive can accept it. The specified archive commanded to physically enter the medium into the archive system. Archives can be either attended or unattended.

OPTIONS

mediaID...

Specifies a list of one or more media to be moved.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

The media ID must be known to the software and not be in an archive. Media cannot be entered if it is being exported, checked out, or checked in. Media that has been imported (see vsimport) prior to calling this command will be physically entered into the archive and system, and the corresponding Library Operator Interface (LOI) request will be cleared.

The media cannot be entered into an an archive that has reached its media type or media class capacity.

-a *archivename*

Specifies the name of the archive to which the specified media are to be moved.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

The archive must be online and support the media type and media class. If the vsarchiveenter command was issued after a vsmove or vsimport command for the same media and the archivenames do not match, then this command will fail.

-v Indicates that verbose output is desired.

If **-v** is specified, status is returned on all media specified in the Archive Enter request.

If **-v** is not specified, status is returned on only those media that were not successfully processed.

-s *portID*

Specifies the import/export port the media is entered from (if applicable).

If the portID is not specified for archives of type scsi and acsls, then a default of 0,0,0,0 will be tried. **vsarchiveqry -s** can be used to obtain the portID for an archive.

If the portID is specified for stage archives, it is ignored.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained

in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

EXIT STATUS

0 The **vsarchiveenter** command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Archive Enter request, verbose option specified.

vsarchiveenter -a vault1 MED016 MED023 MED044 MED048 **-v**

Requests the Media Manager software to enter media MED016, MED023, MED044, and MED048 into a stage archive; and to return status on every specified medium.

Output returned:

```
Archive Enter 4 of 4 media was successful

   Media [MED016]      no error

   Media [MED028]      no error

   Media [MED043]      no error

   Media [MED048]      no error
```

Successful Archive Enter request, verbose option not specified.

vsarchiveenter -a vault1 MED013 MED016 MED028 MED031

Requests the Media Manager software to enter media MED013, MED016, MED028, and MED028 into the specified stage archive.

Output returned:

```
  Archive enter 4 of 4 media was successful.
```

Error(s) with verbose option specified (nonexistent media)

vsarchiveenter -s 0,0,15,16 -a lib1 MED013 MED016 MED022 MED034 **-v**

Requests the Media Manager software to enter media MED013, MED016, MED022, and MED034 and to return status on every specified medium.

Output returned:

```
Archive enter 1 of 4 media was successful

Error VOL024: error in the list

   Media [MED013]   no error

   Media [MED016]   media class is full

   Media [MED022]   media class is full

   Media [MED034]   media class is full
```

Error(s) with verbose option not specified

vsarchiveeenter -s 0,0,15,16 -a lib1 MED003 MED004 MED013 MED028 MED033 MED043

Requests the Media Manager software to enter media MED003, MED004, MED013, MED028, MED033, and MED043.

Output returned:

```
Archive enter 3 of 6 media was successful

Error VOL024: error in the list

   Media [MED003]   media class is full

   Media [MED004]   media class is full
```

```
Media [MED033]   media class is full
```

Unsuccessful Archive Enter request

NOTES

Media that is allocated to an Archive Enter request is not available for other allocation until the enter completes.

A pending Archive Enter request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (control-c).

ERRORS

VOL206	archive type not supported - Only scsi, acsls, and vault archive types are supported for this command
VOL107	an archive was specified that does not exist
VOL081	the specified archive was not online
VOL107	the specified archive does not exist

SEE ALSO

vsarchiveeject(l), **vsimport**(l), **vsmove**(l), **vsarchiveqry**(l)

NAME

vsarchiveqry − Report the status of media archives.

SYNOPSIS

vsarchiveqry [**-Ihcdmstv**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] [**-F** *type*] **-a**|*archivename*

DESCRIPTION

The **vsarchiveqry**(l) command generates a status report for all media archives. The user may limit the report to one particular archive.

REPORT STATUS

In its most verbose form (specified by the **-v** option), the report lists all of the following information for each media archive:

Archive Name
The name of the media archive

Archive Type
The type of archive. Values are: **Stage**, **SCSI**, **ACSLS**

Current State
The current state of the archive. Values are: **ERROR**, **On-line**, **Off-line**, **Diagnostic**, **Unavailable**, **Nonexistent**

Archive Mode
The current mode of the archive. Values are: **Unattended**, **Attended**

Fill Mode (deprecated)
The configured fill mode of the archive. Values are: **None**

Configure State
The current configuration state of the archive. Values are: **Not Being Configured**, **Being Configured**, **Being Reconfigured**, **Being Deleted**, **Terminating**, **Starting**, **Cycling**

Drive ID(s)
The drive IDs for all drives associated with the archive

Media ID(s)
The media IDs for all media associated with the archive

I/E Port(s)
The port IDs (in **X,X,X,X** format) for all import/export ports in the archive

For each media class associated with the archive, the follow information is listed:

MediaClass
The name of the media class

Media Type
The type of media associated with this media class. Values are: **LTO**, **3592**, **T10K**, **LTOW**,

Class Capacity %
The percentage of media in this media class that can be allocated to the archive

Class Capacity
The maximum number of media allowed in this media class

Current Fill Level
The current number of media in this media class

Action The action to take on media in this media class once watermark thresholds have been hit: **None, Notify**

For each media type associated with the archive, the follow information is listed:

Media Type
The type of media. Values are: **LTO**, **3592**, **T10K**, **LTOW**,

Archive Capacity
The maximum number of media of this type allowed in the archive

Current Fill Level
The current number of media of this type in the archive

Assigned Locations
The current number of bins in the archive occupied by media of this type

Auto Checkin
An indication of whether auto-checkin into the archive is enabled for media of this type. Values are: **On**, **Off**

Auto Import
An indication of whether auto-import into the archive is enabled for media of this type. Values are: **On**, **Off**

Action The action to take on media of this type once watermark thresholds have been hit. Values are: **None**, **Notify**

Import Media Class
The media class associated to media of this type upon import into the archive

Import Manufacturer
The name of the media manufacturer specified when the media was entered

Import Batch
The manufacturer's batch identifier specified when the media was entered

OPTIONS

archivename
Identifies the archive to be listed in the report.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-a Specifies that all archives are to be listed in the report.

-c Indicates that detailed information on all media classes associated with the specified archive(s) are to be reported.

-d Indicates that all drives associated with the specified archive(s) are to be reported.

-m Indicates that all media IDs associated with the specified archive(s) are to be reported.

-s Indicates that all import/export ports associated with the specified archive(s) are to be reported.

-t Indicates that detailed information on all media types associated with the specified archive(s) are to be reported.

-v Indicates that all media classes, all drives, all media IDs, all media types, and all import/export ports associated with the specified archive(s) are to be reported.

Specifying the **-v** option is equivalent to specifying the **-c**, **-d**, **-m**, **-s**, and **-t** options.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

EXIT STATUS

0 The **vsarchiveqry** command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Archive Query

vsarchiveqry vault01 **-ct**

Requests the Media Manager software to return the media classes and the media types associated with the vault01 archive.

Output returned:

```
---------------------------------------------
Archive Query Report   May 24 12:43:18 1993          1
---------------------------------------------

Archive:  vault01
```

```
-----------------------------------------------
              Archive Type:  Stage
             Current State:  On-Line
              Archive Mode:  Attended
                 Fill Mode:  None
           Configure State:  Not Being Configured

MediaClass:  F0_LTO_ADDBLANK
             MediaType:  LTO
      Class Capacity %:  100%
        Class Capacity:  500000
    Current Fill Level:  2
                Action:  None

Media Type:  LTO
     Archive Capacity:  500000
   Current Fill Level:  2
   Assigned Locations:  2
         Auto Checkin:  Off
          Auto Import:  Off
               Action:  None
```

Successful Archive Query

vsarchiveqry archive1 **-v**

Requests the Media Manager software to return the drives, the media, the media classes, the media types, and the import/export ports associated with the archive1 archive.

Output returned:

```
-----------------------------------------------
Archive Query Report   May 24 12:59:38 1993           1
-----------------------------------------------

Archive:  archive1

-----------------------------------------------
              Archive Type:  SCSI
             Current State:  On-Line
              Archive Mode:  Attended
                 Fill Mode:  None
           Configure State:  Not Being Configured

   Drive ID(s):     1       2

   Media ID(s):     med001        med002        med003
                    med004
MediaClass:  F0_LTO_ADDBLANK
             MediaType:  LTO
      Class Capacity %:  50%
        Class Capacity:  40
    Current Fill Level:  2

                Action:  None
```

```
Media Type: LTO
     Archive Capacity:  40
   Current Fill Level:  2
   Assigned Locations:  2
         Auto Checkin:  Off
          Auto Import:  On
               Action:  None
   Import Media Class:  FO_LTO_ADDBLANK
  Import Manufacturer:
         Import Batch:

   I/E Port(s):  0,0,15,16
```

Unsuccessful Archive Query

vsarchiveqry BadArchiveName **-d**

Requests the Media Manager software to return the drives associated with the BadArchiveName archive.

Output returned:

```
Archive query was unsuccessful

Error VOL008: item not found
```

NOTES

The Archive Query command does not trigger unsolicited status messages from the Media Manager software.

A pending or executing Archive Query request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request is also aborted by sending the SIGINT signal (control-c).

SEE ALSO

vsarchivevary(l)

NAME

vsarchivevary − Varies the state of an archive.

SYNOPSIS

vsarchivevary [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] [**-F** *type*] **-s** *state archivename*

DESCRIPTION

vsarchivevary(1) is issued from the command line to request execution of the Media Manager Archive Vary command.

The Archive Vary command is used to change the state of a configured archive. The name of the archive and the target state (on-line, off-line, or diagnostic) must be specified.

Upon receipt of an Archive Vary command, the software attempts to change the state of the specified archive. The return code presented to the client indicates the success or failure of the command.

OPTIONS

archivename

Identifies the archive to be varied.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-s *state* Identifies the state into which the archive is placed.

Valid archive states are: on-line, off-line, and diagnostic. The archive states, online,offline, and diagnostic are abbreviated as on, of, and d respectively.

-I Indicates command line options are read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

EXIT STATUS

0 The **vsarchivevary** command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful archive vary

vsarchivevary shelf1 **-s** diagnostic

Requests the Media Manager software to vary the shelf1 archive to the diagnostic state.

Output returned:

```
Vary of archive [shelf1] to state [diagnostic] was   successful
```

Unsuccessful archive vary

vsarchivevary BadArchiveName **-s** on-line

Requests the Media Manager software to vary the BadArchiveName archive to the online state.

Output returned:

```
Vary of archive [BadArchiveName] to state [on  line] was   unsuccessful

Error VOL013: invalid archive
```

NOTES

The Media Manager software rejects all incoming requests that could physically command an off-line or diagnostic archive (for example: Mount, Dismount, and Move).

The Media Manager software processes commands that interact strictly with the database (for example: Query Mount, Create Drive Pool, and Create Archive Media Class), regardless of the state of the associated archive.

All components associated with an offline or diagnostic archive, such as media, drives, and physical hardware, are unavailable.

The Archive Vary command does not trigger unsolicited status messages from the Media Manager software.

A pending Archive Vary request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control\). The request also is aborted by sending the SIGINT signal (control-c).

SEE ALSO

vsarchiveqry(l), **vsdrivevary**(l)

NAME

vsarcmedclasscreate − Associates an existing media class to an archive.

SYNOPSIS

vsarcmedclasscreate [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] [**-c** *capacityPercent*] **-a** *archive* *mediaclass*

DESCRIPTION

The **vsarcmedclasscreate**(l) command associates an existing media class to an archive. This association is referred to as an archive media class.

The media class must be created beforehand with the **vsmedclasscreate**(l) command. Afterward, the media class may be associated with one or more archives.

OPTIONS

-I Indicates command line options are to be read from stdin.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

-c *capacityPercent*

This parameter specifies the percentage of media contained in the media class which can be allocated to the specified archive. This defaults to 0 when not specified but should typically be set to 100.

-a *archive*

Identifies the archive name that the media class is to be associated with. The archive name parameter is a user-specified name by which the archive is known to the user.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

mediaclass

The name of the media class. The media class name parameter is a user-specified name by which the media class is known to the user.

Valid media class names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255
The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

Associate a media class called *abc* with archive *lib1*

vsarcmedclasscreate -c 100 **-a** lib1 abc

NOTES

To cancel the request generated by this command, use the **vscancelreq** command, or send a SIGQUIT signal (control-\).

The command can be aborted by sending a SIGINT signal (control-c), however this will not cancel the request.

SEE ALSO

vsarcmediaclassdelete(l), **vscancelreq**(l)

NAME

vsarcmedclassdelete − Deletes an archive media class for the Media Manager system.

SYNOPSIS

vsarcmedclassdelete [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] **-a** *archive mediaclass*

DESCRIPTION

vsarcmedclassdelete(l) is issued from the command line to request execution of the Media Manager Archive Media Class Delete command. The Media Manager software must be active in order to run this command.

This will remove the association of a media class from an archive.

OPTIONS

-I Indicates command line options are to be read from stdin.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

-a *archive*

Identifies the archive name that the media class is associated with. The archive name parameter is a user-specified name by which the archive is known to the user.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

mediaclass

The media class name parameter is a user-specified name by which the media class is known to the user.

Valid media class names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255

The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

Delete association of a media class called *abc* from archive *lib1*

vsarcmedclassdelete -a lib1 abc

NOTES

To cancel the request generated by this command, use the **vscancelreq** command, or send a SIGQUIT signal (control-\).

The command can be aborted by sending a SIGINT signal (control-c), however this will not cancel the request.

SEE ALSO

vsarcmediaclasscreate(l), **vscancelreq**(l)

NAME

vsaudit − Performs archive inventory verification.

SYNOPSIS

vsaudit *archivename* [**-Ihp**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsaudit(l) is issued from the command line to request execution of the Media Manager Audit command.

Upon receipt of an Audit command request, the Media Manager software performs an inventory verification of the specified archive.

If the specified archive is robotically controlled, the robot scans each physical bin location and verifies that the database is consistent with the actual location of media. Any noted inconsistencies are returned to the client, logged in a system log file, and the software initiates corrective action, based on the circumstances of the discrepancy.

If the specified archive is a manual archive, the archive operator is directed to generate the audit report. The operator then directs the report to be printed or to verify the information online. Either way, the operator performs the inventory and corrects any reported discrepancies. Discrepancies are resolved by issuing appropriate media management commands (for example, Eject) to relocate media to the appropriate locations. Audits of manual archives do not return a discrepancy list.

Audit requests are for full archive audits only; no subset audits are permitted from the command line. Subset audits are conducted from the system operator GUI. Full archive audits are lengthy and should be requested with discretion.

OPTIONS

archivename

Specifies the name of the archive to audit.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-I Indicates command line options are read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-p Causes a physical inventory of a library.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsaudit** command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful audit request

vsaudit shelf1

Requests the Media Manager to audit the shelf1 archive.

Output returned:

```
------------------------------------------------

Audit Report           May 24 12:43:18 1993           1

------------------------------------------------

Archive:  shelf1

------------------------------------------------

no discrepancies found
```

Unsuccessful audit request

vsaudit BadArchiveName

Requests the Media Manager software to audit the BadArchiveName archive.

Output returned:

```
Audit of archive [BadArchiveName] was unsuccessful

Error VOL013: invalid archive
```

NOTES

With the exceptions of the manual archives, a pending or executing Audit request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (controlc).

A pending or executing Audit request is reprioritized using the Media Manager Reprioritize command. The Media Manager Reprioritize command is not available from the command line. The Reprioritize command is available to the client through either the API or the RPC interface.

A pending or executing STK ACS product family archive audit requires a Cassette Autoloader Port (CAP). If the CAP is busy, the Audit command can be queued. This results in intermediate status that indicates the Audit command is waiting for a busy CAP to be freed.

In the DataTower or STK ACS product family database, the Media Manager software does not actually

track media location to the bin level, only down to the Manipulator Unit (MU) level. However, the logic and the Media Manager software responses are similar to the bin tracking performed in the DataLibrary software with no internal database.

The Audit command does not trigger unsolicited status messages from the Media Manager software.

The total length of time the CLI software waits for a command status, in synchronous mode, from the Media Manager software is (VSID_RETRY_LIMIT plus 1) multiplied by VSID_TIMEOUT_VALUE. Because of the time required for robotic audits, the time-out value or retries may need to be increased from the CLI default values.

SEE ALSO

None

NAME

vscancelreq − cancels an outstanding request for the Media Manager system.

SYNOPSIS

vscancelreq [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] **-c** *command requestid*

DESCRIPTION

vscancelreq(l) is issued from the command line to request execution of the Media Manager Cancel Request command. The Media Manager software must be active in order to run this command.

Any command in the command queue, except the dismount command, can be cancelled. The arguments needed for this command can be obtained by issuing the Get Request Identifiers command.

OPTIONS

-I Indicates command line options are to be read from stdin.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

-c *command*

This is command identifier for the request being cancelled.

requestid

The request identifier of the request to cancel.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255

The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

First list the active requests and then cancel an outstanding mount request.

vsgetreqids

Output returned:

```
1122518995˜Mount˜14
```

Then cancel the request using the results

vscancelreq -c 14 1122518995

SEE ALSO

vsgetreqids(l)

NAME

vscheckin − Logically checks media into the Media Manager system that were previously checked out of the Media Manager system.

SYNOPSIS

vscheckin *mediaID...* [**-a** *archivename*] [**-v**] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vscheckin(l) is issued from the command line to request execution of the Media Manager CheckIn command.

A client requests execution of the Media Manager CheckIn command to logically check media into the Media Manager system. Only media that have been previously checked out can be checked in.

Checkin is a logical operation. After a medium is logically checked in to the Media Manager system, the medium is physically entered into an archive before becoming available for client use (mounting,...). A medium is physically entered into the Media Manager system via the Enter functionality that is available from the appropriate archive's console display. The Enter functionality is not available from the command line.

OPTIONS

mediaID...

Specifies a list of one or more media to be checked in.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-a *archivename*

Specifies the name of the destination archive for the media to be entered into after they are checked in.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-v Indicates that verbose output is desired.

If the **-v** option is specified, status is returned on every medium specified in the **vscheckin**(l) command.

If **-v** is not specified, status is returned on only those media that were not successfully checked in.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vscheckin** command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Checkin request with verbose option specified

vscheckin MED012 MED014 MED023 **-v**

Requests the Media Manager software to check MED012, MED014, and MED023 into the archive from which they were checked out and to return status on each medium.

Output returned:

```
Check in 3 of 3 media was successful

   Media [MED012]      no error

   Media [MED014]      no error

   Media [MED023]      no error
```

Successful Checkin request with verbose option not specified

vscheckin MED013 **-a** shelf2

Requests the Media Manager software to check MED013 into the shelf2 archive and to return status on a medium only if processing for that medium failed.

Output returned:

```
Check in 1 of 1 media was successful
```

Error(s) with verbose option specified

vscheckin MED011 MED014 MED021 **-v**

Requests the Media Manager software to check MED011, MED014, and MED021 into the archive from which they were checked out and to return status on every specified medium.

Output returned:

```
Check in 1 of 3 media was successful

Error VOL024: error in the list
```

```
Media [MED011]   invalid action or location state for
                 operation
Media [MED014]   no error
Media [MED021]   item not found
```

Error(s) with verbose option not specified.

vscheckin MED001 MED002 MED093 **-a** stage1

Requests the Media Manager software to check MED001, MED002, and MED093 into the stage1 archive and to return status on a medium only if processing for that medium failed.

Output returned:

```
Check in 1 of 3 media was successful
Error VOL024: error in the list
   Media [MED001]   archive not associated with media class
   Media [MED093]   item not found
```

NOTES

Media must be checked out of the Media Manager system before the Checkin command request is valid.

Media checked out of one archive can be checked in to another archive, as long as the receiving archive is configured to support the media's MediaClass group and the receiving archive is not at capacity for the media's media type.

Media checked out from more than one archive can be checked in as a single group into a single new archive (assuming necessary archive media class associations exist).

Media that are checked out from more than one archive and are checked in as a single group without a target archive specified on the Checkin command are returned to their respective check-out archives.

Failure of the Checkin request for one or more media in a list does not fail the request for all media in the list.

The Checkin command triggers unsolicited status messages from the Media Manager software to the client software.

A pending or executing Checkin request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (control-c).

SEE ALSO

vscheckout(l)

NAME

vscheckout − Checks media out of the Media Manager system.

SYNOPSIS

vscheckout *mediaID...* [**-t***comment*] [**-v**] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vscheckout(l) is issued from the command line to request execution of the Media Manager Checkout command.

A client uses the Checkout command to logically remove media from the Media Manager system. Media that are checked out are still known by the software, but are unavailable for client allocation.

Upon receipt of a Checkout request, the Media Manager software marks the specified media for checkout. If the specified media are contained in archives, the Media Manager software adds the media to the Eject list of the containing archive. An operator selects the Eject functionality from the appropriate archive's console display to physically remove the checked-out media from the containing archive.

OPTIONS

mediaID...

Specifies a list of one or more media to be checked out of the Media Manager system.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-t *comment*

Provide a comment to be associated with each checked-out media. This comment is provided on the Eject list (a GUI display) from the archive console associated with the archive containing the media.

-v Indicates that verbose output is needed.

If the **-v** option is specified, status is returned on every medium specified in the **vscheckout**(l) command.

If **-v** is not specified, status is returned on only those media that were not successfully checked out.

-I Indicates command line options are read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vscheckout**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Checkout request with verbose option specified

vscheckout MED003 MED004 MED005 **-t** "Transfer to Library" **-v**

Requests the Media Manager software to check MED003,MED004,and MED005 out of the Media Manager system and to return status on every specified medium.

Output returned:

```
Check out 3 of 3 media was successful

   Media [MED003]      no error

   Media [MED004]      no error

   Media [MED005]      no error
```

Successful Checkout request with verbose option not specified.

vscheckout MED003 MED004 MED005 MED006 MED007 MED008

Requests the Media Manager software to check MED003, MED004, MED005, MED006, MED007, and MED008 out of the Media Manager system and to return status on a medium only if processing for that medium failed.

Output returned:

```
Check out 6 of 6 media was successful
```

Error(s) with verbose option specified

vscheckout MED010 MED011 MED012 MEDa13 **-v**

Requests the Media Manager software to check MED010, MED011, MED012, and MED13 out of the Media Manager system and to return status on every specified medium.

Output returned:

```
Check out 2 of 4 media was successful

Error VOL024: error in the list

   Media [MED010]      invalid action or location state for operation

   Media [MED011]      no error
```

```
Media [MED012]      no error

Media [MED013]      item not found
```

Error(s) with verbose option not specified

vscheckout MED010 MED011 MED012 MEDa13

Requests the Media Manager software to check MED010, MED011, MED012, and MEDa13 out of the Media Manager system and to return status on a medium only if processing for that medium failed.

Output returned:

```
Check out 2 of 4 media was successful

Error VOL024: error in the list

   Media [MED010]      invalid action or location state for operation

   Media [MED013]      item not found
```

NOTES

Failure of the Checkout request for one or more media in a list does not fail the request for all media in the list.

A currently allocated medium is checked out of the Media Manager system. Attempts to physically eject an allocated medium fail until the medium is no longer in use.

A medium marked for checkout is unmarked (removed from the Eject list) by the Clear Eject command. An operator removes a medium from the Eject list by performing an Eject Fail operation from the appropriate archive's console display. The Eject Fail functionality is not available from the command line.

The Clear Eject command is available to clients, whereas, Fail Eject is an operator-only command.

The Checkout command triggers unsolicited status messages from the Media Manager software.

A pending or executing Checkout request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request is aborted by sending the SIGINT signal (control-c).

SEE ALSO

vscheckin(l), **vsclearеject**(l)

NAME

vsclearejectˉ Removes the specified media from the archive's Eject list

SYNOPSIS

vsclearejectˉ *mediaID...* [**-v**] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] [**-V** *number***]**

DESCRIPTION

vsclearejectˉ(l) is issued from the command line to request execution of the Media Manager Clear Eject command.

A client uses the Clear Eject command to reverse the scheduled eject of one or more media from an archive.

Ejects can be generated during processing of the Media Manager Checkout Export, Mount, and Move commands. Ejects can also be generated during automigration.

The Clear Eject command essentially undoes the completion of these commands. Media are removed from the Eject list and returned to the available state.

For example, if a client issues an **export** command for a specific medium, the specified medium is scheduled for removal by adding the medium to the Eject list for the archive associated with the medium. If the client decides the medium should not be removed from its associated archive, the client issues the Clear Eject command, and the Media Manager software removes the medium from the Eject list, thus voiding the Export request.

OPTIONS

mediaID...

Specifies the media to remove from the Eject list.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-v Indicates that verbose output is desired.

If **-v** is specified, return status on every media specified in the command.

If **-v** is not specified, return status on only the media that were not successfully processed.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsclearej**ect(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful with verbose option specified

vscleareject MED017 MED021 MED023 **-v**

Requests the Media Manager software to remove MED017, MED021, and MED023 from the Eject list and to return status on each specified medium.

Output returned:

```
Clear Eject 3 of 3 media was successful

   Media [MED017]      no error

   Media [MED021]      no error

   Media [MED023]      no error
```

Successful with verbose option not specified

vscleareject MED016 MED018 MED020 MED021

Requests the Media Manager software to remove MED016, MED018, MED020, and MED021 from the Eject list and to return status on a medium only if processing for that medium failed.

Output returned:

```
Clear Eject 4 of 4 media was successful
```

Error(s) with verbose option specified

vscleareject MED012 MED013 MEd014 MED051 **-v**

Requests the Media Manager software to remove MED012, MED013, MED014, and MED051 from the Eject list and to return status on every specified medium.

Output returned:

```
Clear Eject 2 of 4 media was successful

Error VOL024: error in the list

   Media [MED012]      no error

   Media [MED013]      item not marked for ejection
```

```
Media [MED014]      no error

Media [MED051]      item not found
```

Error(s) with verbose option not specified

vsclearejectMED012 MED013 MED014 MED051

Requests the Media Manager software to remove MED012, MED013, MED014, and MED051 from the Eject list and to return status on a medium only if processing for that medium failed.

Output returned:

```
Clear Eject 2 of 4 media was successful

Error VOL024: error in the list

   Media [MED013]      medium not marked for ejection

   Media [MED051]      item not found
```

NOTES

The Clear Eject request fails for a medium if the medium is already selected for eject by the operator.

Failure of the **vsclearejectrequest** for one or more media in a list does not fail the request for all media in the list.

An operator also removes a medium from the Eject list by performing an Eject Fail from the appropriate archive's console display. The Eject Fail functionality is not available from the command line.

The Clear Eject command triggers unsolicited status messages from the Media Manager software.

A pending Clear Eject request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control\). The request also is aborted by sending the SIGINT signal (controlc).

SEE ALSO

vscheckout(l), **vsexport**(l), **vsmount**(l), **vsmove**(l)

NAME

vsdismount − Dismounts a medium from a drive.

SYNOPSIS

vsdismount *mediaID* **-d** *driveID* [**-l** *lockID*] [**-u** *usagetime*] [**-e** *errorcount*] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsdismount(l) is issued from the command line to request execution of the Media Manager Dismount command.

A client uses the Dismount command to inform the software that the client is no longer using a drive and the medium mounted in the drive.

Upon receipt of a Dismount request for an automated archive, the software checks to see that the medium is ejected from the drive by the storage subsystem. If the medium is not ejected from the drive, the Dismount request fails and the Media Manager software returns a failure status to the client.

For automated archives, if the medium is ejected from the drive, the software commands the archive robotics to move the medium from the drive pickup point to a bin within the archive system. A successful return code is returned to the client after the medium movement is completed.

For manual archives, a dismount notice is sent to the appropriate archive's console display for action. An operator dismounts the specified medium and then notifies the Media Manager software that the medium dismount is complete. The Media Manager software returns a successful return code to the client only after the operator confirms the dismount is complete.

OPTIONS

mediaID

Identifies the medium to be dismounted.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-d *driveID*

Identifies the drive on which the medium is mounted.

-l *lockID*

Specifies the lock identifier associated with the drive if the drive is mounted with a lock identifier.

-u *usagetime*

The amount of time (in seconds) the drive is in use.

-e *errorcount*

The number of errors encountered while interacting with the drive.

-I Indicates command line options are read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsdismount**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful dismount

vsdismount MED012 **-d** 2

Requests the Media Manager software to dismount MED032 from drive 2.

Output returned:

```
Dismount of Media [MED032] from Drive [2] was successful
```

Unsuccessful dismount request

vsdismount MED016 **-d** 13

Requests the Media Manager software to dismount MED016 from drive 13.

Output returned:

```
Dismount of Media [MED016] from Drive [13] was unsuccessful

Error VOL081: drive not mounted
```

Unsuccessful dismount request

vsdismount MED016 **-d** 9

Requests the Media Manager software to dismount MED016 from drive 9.

Output returned:

```
Dismount of Media [MED016] from Drive [9] was unsuccessful

Error VOL044: medium not mounted
```

NOTES

The Dismount command triggers unsolicited status messages from the Media Manager software.

A pending Dismount request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control\). The request also is aborted by sending the SIGINT signal (controlc).

SEE ALSO

vsmount(l)

NAME

vsdrivecfg − Configures a specified drive for the Media Manager system.

SYNOPSIS

vsdrivecfg -c *driveID* **-t** *mediaType...*
[**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsdrivecfg -b *driveID* **-a** *archiveName* [**-s** *slotName*]
[**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsdrivecfg -u *driveID* **-a** *archiveName*
[**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsdrivecfg -r *driveID*
[**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsdrivecfg(l) is issued from the command line to request execution of the Media Manager Drive Configure command. The Media Manager software must be active in order to run this command.

This command is use to create or remove a drive, and associate or disassociate a drive with an archive. A drive can be created and associated with an archive at any time providing the Media Manager software is running. All created drives must be associated with an archive before it can be used to mount media. Drives can be defined before the creation of media classes.

OPTIONS

-c *driveID*

This will create a drive with the specified identifier which is used by other commands to reference the drive. The drive identifier must be a unique numeric integer and is restricted to a range from 1 to 2147383647 inclusive.

-b *driveID*

This will associate the specified drive with an archive.

-u *driveID*

This will disassociate the specified drive from an archive. An in-use drive cannot be disassociated from an archive.

-r *driveID*

This will remove the specified drive. A drive can only be removed if is not associated with any archives.

-a *archiveName*

The name of the archive which the drive is to be associated or disassociated. The archive name parameter is a user-specified name by which the archive is known to the user.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-t *mediaType...*

This list specifies the valid media types that can be used with the drive. Media types can be listed with the Media Type query command.

-s *slotName*

This specifies the location where the drive is physically located in the archive. These can be obtained for an archive by running the **dbdrvslot** utility.

-I Indicates command line options are to be read from stdin.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255

The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

Create a drive with an identifier of 1 that supports LTO and LTO Worm media types.

```
vsdrivecfg -c  1 -t LTO LTOW
```

Associate drive 1 with SCSI archive *lib1* that has a slotName of 0,0,12,256.

```
vsdrivecfg -b  1 -a lib1 -s  0,0,12,256
```

Remove drive 1. Note that the drive must first be disassociated.

```
vsdrivecfg -u  1 -a lib1
vsdrivecfg -r  1
```

NOTES

To cancel the request generated by this command, use the **vscancelreq** command, or send a SIGQUIT signal (control-\).

The command can be aborted by sending a SIGINT signal (control-c), however this will not cancel the request.

SEE ALSO

vsmedtypeqry(l), **vsdriveqry**(l), **dbdrvslot**(l), **vscancelreq**(l)

NAME

vsdriveqry − Queries for information on the specified drive(s).

SYNOPSIS

vsdriveqry *driveID...* [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsdriveqry -a [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsdriveqry(l) is issued from the command line to request execution of the Media Manager Drive Query command.

The Drive Query command is used to obtain information about one or more drives in a Media Manager system.

OPTIONS

driveID...

If the **-a** option is not specified, specify a list of one or more drives to be queried.

The number of drives that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-a Specifies the **-a** option to indicate all drives known to the Media Manager system are to be queried.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsdriveqry**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful drive query

vsdriveqry -a

Requests the Media Manager software to return information on every drive known to the Media Manager system.

Output returned:

```
    ------------------------------------------------

    Drive Query Report     Mar 10 11:00:32 1994         1

    ------------------------------------------------

    Drive ID:  1

    ------------------------------------------------

       Drive Type:                  Magnetic

       Associated Archive:          stage1

       Current State:               On-line

       Assignment:                  Free

       Usage Count:                 0

       Current Usage Time:          0

       Total Usage Time:            0

       Error Count:                 0

       Mount State:                 Unmounted

       Mounted Media ID:

       Media Type(s) Supported:     D2M

    Drive ID:  3

    ------------------------------------------------

       Drive Type:                  Magnetic

       Associated Archive:          shelf1

       Current State:               On-line

       Assignment:                  Free

       Usage Count:                 1
```

```
   Current Usage Time:          0

   Total Usage Time:            0

   Error Count:                 0

   Mount State:                 Unmounted

   Mounted Media ID:

   Media Type(s) Supported:     D2M

-----------------------------------------------

Drive ID:  4

-----------------------------------------------

   Drive Type:                  Magnetic

   Associated Archive:          shelf1

   Current State:               On-line

   Assignment:                  Free

   Usage Count:                 0

   Current Usage Time:          0

   Total Usage Time:            0

   Error Count:                 0

   Mount State:                 Unmounted

   Mounted Media ID:

   Media Type(s) Supported:     D2M
```

Unsuccessful drive query

vsdriveqry 35

Requests the Media Manager software to return information on drive 35. (Drive 35 does not exist.)

Output returned:

```
Drive query was unsuccessful.

Error VOL008: item not found.
```

NOTES

The Drive Query command does not trigger unsolicited status messages from the Media Manager software.

A pending Drive Query request is cancelled with the Media Manager Cancel command. The Media

Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control\). The request also is aborted by sending the SIGINT signal (controlc).

SEE ALSO

vsdrivevary(l)

NAME

vsdrivevary − Changes the state of a drive.

SYNOPSIS

vsdrivevary *driveID...* **-s** *state* [**-v**] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] [**-V** *number***]**

vsdrivevary -p *drivepool* **-s** *state* [**-v**] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] [**-V** *number***]**

DESCRIPTION

vsdrivevary(l) is issued from the command line to request execution of the Media Manager Drive Vary command.

The Drive Vary command is used to change the operational availability state of a drive.

A drive in the offline, unavailable, or diagnostic state is excluded from the software's drive selection algorithm.

A Mount or Lock request for an off-line, unavailable, or diagnostic drive fails.

Conversely, varying a drive to the online state makes it available for selection for Mount or Lock requests.

OPTIONS

driveID...

Specifies one or more individual drive(s) whose state is to be varied.

The number of drives that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-p *drivepool*

Specifies the name of a drive pool to vary the state of all drives associated with the drive pool.

Valid drive pool names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-s *state* Specifies the target state of the specified drive(s). Valid drive states are on-line, off-line, and diagnostic. The drive states, online, offline, and diagnostic can be abbreviated as on, of, and d respectively.

-v Indicates that verbose output is needed.

If **-v** is specified, list status information on all drives varied in the Media Manager system.

If **-v** is not specified, reports only those drives unsuccessfully varied.

-I Indicates command line options are read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsdrivevary**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful drive vary with verbose option specified

vsdrivevary -p drvpoolusr **-s** offline **-V**

Requests the Media Manager software to vary all drives in the drvpoolusr drive pool to the offline state and to return status on every drive in drvpoolusr.

Output returned:

```
Vary 8 of 8 drives to state [off  line] was successful.

   Drive [1]     no error

   Drive [4]     no error

   Drive [5]     no error

   Drive [7]     no error

   Drive [8]     no error

   Drive [11]    no error

   Drive [12]    no error

   Drive [14]    no error
```

Successful drive vary with verbose option not specified

vsdrivevary 2 **-s** diagnostic

Requests the Media Manager software to vary drive 2 to the diagnostic state and to return status on a drive only if processing for that drive failed.

Output returned:

```
Vary 1 of 1 drives to state [diagnostic] was successful.
```

Error(s) with verbose option specified

vsdrivevary 5 15 8 18 11 **-s** on-line **-v**

Requests the Media Manager software to vary drives 5, 15, 8, 18, and 11 to the online state and to return status on every specified drive.

Output returned:

```
Vary 3 of 5 drives to state [on-line] was successful

   Drive [5]        no error

   Drive [15]       invalid drive specified

   Drive [8]        no error

   Drive [18]       invalid drive specified

   Drive [11]       no error
```

Error(s) with verbose option not specified

vsdrivevary 5 15 8 18 11 **-s** off-line

Requests the Media Manager software to vary drives 5, 15, 8, 18, and 11 to the offline state and to return status on a drive only if processing for that drive failed.

Output returned:

```
Vary 3 of 5 drives to state [off-line] was successful

Error VOL024: error in the list

   Drive [15]       invalid drive specified

   Drive [18]       invalid drive specified
```

Unsuccessful Drive Vary request

vsdrivevary -**p** BadPoolName **-s** diagnostic

Requests the Media Manager software to vary every drive associated with the BadPoolName drive pool to the diagnostic state.

Output returned:

```
Error VOL030: invalid drive pool specified
```

NOTES

Mounted drives that have their state changed remain inuse. Varying a drive has no impact on client data transfer operations in progress and the client receives no automatic notification of a drive state change.

Drives can be varied, regardless of whether or not they are associated with an archive.

Drives can be varied, regardless of whether or not they are allocated; however, allocated drives that are not on-line cannot be dismounted.

The unavailable state is assignable only by the Media Manager software when a higher level component in the archive system is no longer online. For example, varying a CLM offline causes the associated drive to be viewed as unavailable.

The Drive Vary command does not trigger unsolicited status messages from the Media Manager software.

A pending Drive Vary request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (controlc).

SEE ALSO

vsdriveqry(l)

NAME

vsenter − Enters media into an archive.

SYNOPSIS

vsenter [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]
-s *portID* [**-e**] [**-c** *mediaclass*] [**-b** *batch* **-f** *manufacturer*] *archivename*

DESCRIPTION

vsenter(l) is issued from the command line to request execution of the Media Manager Enter command. The archive must be online to run this command.

The Enter command is used to enter all media that are in the import/export ports into an archive. It is primarily for entry of unknown media, however, it can also be used to re-enter checked out media. This command does not support **stage** type archives (vaults). To enter media into a **stage** archive refer to the Archive Enter command.

All media can be classified as known or unknown. Unknown media have never been introduced into a Media Manager system or were removed using the Export Media command. Known media are contained in an archive, are checked out using the Check out Media command, or are intransit to an archive as a result of a Move Media command, or Mount Media command.

Three ways exist to enter unknown media into an archive. One way is the Import Media command to specify a list of media. Another is to load a large number of media at one time using the bulk load method and then run the Audit command. Finally, the Enter Media command to enter all media in the archive import/export ports.

Multiple ways exist to enter known media into an archive. One way is the Archive Enter command to specify a list of media. Another way is to run the Audit command. Checked out media can be re-entered with the Checkin command or this command. When this command is used to re-enter checked out media, then any checked-out media detected in the import/export ports will be entered if the archive is configured for automatic checkin or if the **-e** option is specified.

OPTIONS

-I Indicates command line options are to be read from stdin.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

-s *portID*

Specifies the import/export port the media is entered from (if applicable). **vsarchiveqry -s** can be used to obtain the portID for an archive.

-e Provides the capability to enable automatic checkin. When the target archive is not configured with the auto checkin option, than all media that are checked out are left in the load port and a failure message is logged.

-c *mediaclass*

Identifies the mediaclass name with which any entered unknown media are to be associated.

-b *batch*

Specifies the manufacturer's batch that contains the media to be entered.

If this option is specified, then the **-f** *manufacturer* option must also be specified. Valid batch names may contain up to 32 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-f *manufacturer*

Specifies the name of the media manufacturer.

If this option is specified, then the **-b** *batch* option must also be specified. Valid manufacturer names may contain up to 32 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

archivename

Specifies the name of the archive to which the specified media are to be added.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255

The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

Enter media where import/export ports contain 2 unknown media and 1 duplicate media

vsenter -s 0,0,15,16 libc

Output returned:

```
------------------------------------------------------------------------------
Enter Report              16-Sep-2013 10:29:05                  1
------------------------------------------------------------------------------

Archive Name: libc
------------------------------------------------------------------------------

Media ID      : 000018
Error         : medium already exists in an archive

Media ID      : 000049
Error         : no error

Media ID      : 000048
Error         : no error
```

Enter media where there are no media in the import/export ports

vsenter -s 0,0,15,16 liba

Output returned:

```
Enter of media into archive [liba] was unsuccessful
Error VOL080: port is empty
```

Attempt to re-enter a checked out media into an archive which is not configured for auto checkin.

vsenter -s 0,0,15,16 liba

Output returned:

```
------------------------------------------------------------------------------
Enter Report               16-Sep-2013 08:42:27                  1
------------------------------------------------------------------------------

Archive Name: liba
------------------------------------------------------------------------------

Media ID      : 000216
Error         : medium must be checked in
```

Use the auto check-in option to re-enter a checked out media into an archive which is not configured for auto checkin.

vsenter -s 0,0,15,16 **-e** liba

Output returned:

```
------------------------------------------------------------------------------
Enter Report               16-Sep-2013 09:12:05                  1
------------------------------------------------------------------------------

Archive Name: liba
------------------------------------------------------------------------------

Media ID      : 000216
Error         : no error
```

NOTES

To cancel the request generated by this command, use the **vscancelreq** command, or send a SIGQUIT signal (control-\).

The command can be aborted by sending a SIGINT signal (control-c), however this will not cancel the request.

SEE ALSO

vsimport(l), **vsarchiveqry**(l), **vsarchiveenter**(l), **vscancel**(l)

NAME

vsexport − Marks media and related media information for removal from the Media Manager system.

SYNOPSIS

vsexport *mediaID...* [**-t** comment] [**-v**] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsexport(l) is issued from the command line to request execution of the Media Manager Export command.

A client uses the Export command to mark media and related information for removal from the Media Manager system. If the specified media are not associated with an archive, they are logically removed from the Media Manager system. If the specified media are associated with an archive, they are placed on the Eject list of the appropriate archive.

A client can also use the Export command to remove information about media that have been checked out of the archive and are physically out of the archive.

Upon receipt of an Export request, the software marks the specified media for eject and returns a successful return code to the client. The **Eject** button is highlighted on the operator's console to indicate that media need to be ejected from the archive.

To physically remove the media marked for export from the archive system, an operator must select the Eject functionality from the appropriate archive's console display. The Eject functionality is not available from the command line.

After media, specified on a **vsexport**(l) command, is physically removed from the archive system, the media is no longer under the control of the software, and all information related to exported media is deleted from the system.

OPTIONS

mediaID...

Specifies a list of one or more media to export.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-t *comment*

Provide a text message to be displayed on the archive console for each medium being exported. This comment is provided on the Eject list (a GUI display) from the archive console associated with the archive containing the media.

The length of the comment is restricted by the CLI software. Currently, the maximum allowed length is 80.

-v Indicates that verbose output is desired.

If **-v** is specified, list status information on all media specified on the command.

If **-v** is not specified, report on only those media for which processing failed.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsexport** command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful export request with verbose and comment options specified

vsexport MED012 MED014 MED016 **-t** Media to be shipped offsite **-v**

Requests to Media Manager software to place media MED012, MED014, and MED016 on the Eject list with the comment, *Media to be shipped off-site*, and to return status on every specified medium.

Output returned:

```
Export 3 of 3 media was successful

   Media [MED012]      no error

   Media [MED014]      no error

   Media [MED016]      no error
```

Successful export request with comment option specified and verbose option not specified

vsexport MED001 MED002 MED003 MED012 MED014 MED016 **-t** *Media to be shipped offsite*

Requests to Media Manager software to place media MED001, MED002, MED003, MED012, MED014, and MED016 on the Eject list with the comment, *Media to be shipped off-site*, and to return status on a medium only if processing for that medium failed.

Output returned:

```
Export 6 of 6 media was successful
```

Error(s) with verbose and comment options specified

vsexport MED007 MED014 MED021 MED028 MED053 MED042 **-t** *Media to be shipped offsite* **-v**

Requests to Media Manager software to place media MED007, MED014, MED021, MED028, MED053, and MED042 on the Eject list with the comment, *Media to be shipped off-site*, and to return status on every specified medium.

Output returned:

```
Export 4 of 6 media was successful

Error VOL024: error in the list

   Media [MED007]      no error

   Media [MED014]      invalid action or location state

                        for operation

   Media [MED021]      no error

   Media [MED028]      no error

   Media [MED053]      item not found

   Media [MED042]      no error
```

Error(s) with verbose option not specified and comment option specified

vsexport MED007 MED014 MED021 MED028 MED053 MED042 **-t** *Media to be shipped offsite*

Requests to Media Manager software to place media MED007, MED014, MED021, MED028, MED053, and MED042 on the Eject list with the comment, *Media to be shipped off-site*, and to return status on a medium only if processing on that medium failed.

Output returned:

```
Export 4 of 6 media was successful

Error VOL024: error in the list

   Media [MED014]      invalid action or location state

                        for operation

   Media [MED053]      item not found
```

NOTES

The Export command cannot be cancelled. Media can be unmarked for export via the Clear Eject request or if the operator fails the eject.

A medium that is marked for ejection from the archive system cannot be reallocated to satisfy a client request, except to satisfy a query of the medium. Any other request (except Clear Eject) received for that medium fails.

An allocated medium can be marked for export. Attempts to physically eject an allocated medium fail until the medium is no longer inuse.

The Export command triggers unsolicited status messages from the Media Manager software to the client

software.

SEE ALSO

vscleareject(l), **vsimport**(l)

NAME

vsgetreqids − Lists the outstanding requests for the Media Manager system.

SYNOPSIS

vsgetreqids

DESCRIPTION

vsgetreqids(l) is issued from the command line to request execution of the Media Manager Get Request Identifiers command. The Media Manager software does not need to be active to run this command, however there would be output to display when the software is not running.

This will return a list of requests that are active in the Media Manager system.

OUTPUT

Each line of output contains a request identifier, a command type and a command identifier with each field separated by a "˜". The syntax for a line of output would be:

Request Identifier˜Command Type˜Command Identifier

Request Identifier
: The request id associated with the command. It can be used with the Request Query command to obtain more details about the request or the Cancel Request command to cancel the request.

Command Type
: Text identifying the type of the command.

Command Identifier
: The numeric identifier for the command. This value is needed by the Cancel Request command.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255
: The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

List active requests. In this example there are outstandings requests for an audit, mounts, dismount, eject and enter operations.

vsgetreqids

Output returned:

```
1122519038˜Audit˜7
1122518995˜Mount˜14
1122518996˜Mount˜14
1122518999˜Dismount˜16
1122519025˜Archive Eject˜47
1122519034˜Archive Enter˜48
```

SEE ALSO

vsrequestqry(l), **vscancelreq**(l)

NAME

vsimport − Logically adds media to the Media Manager system.

SYNOPSIS

vsimport *mediaID...* **-a** *archivename* **-Bc** *mediaclass* [**-f** *manufacturer* **-b** *batch*] [**-v**] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsimport(l) is issued from the command line to request execution of the Media Manager Import command.

A client uses the Import command to logically add media to the Media Manager system. Upon receipt of an Import request, the specified media are added to the Media Manager system. If a non-unique media identifier is specified, the Import for that medium fails.

The Import is a logical operation. Media must be physically entered into an archive before they are available for client use (mounting,...). Entry is performed when an operator selects the Enter functionality from the appropriate archive's console display. The Enter functionality is not available from the command line.

OPTIONS

mediaID...

Specifies a list of one or more media to import.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-a *archivename*

Identifies the archive into which the media are to be entered.

The Media Manager software sends Enter commands for the media to be entered to the console for the specified archive.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-c *mediaclass*

Identifies the MediaClass name with which the imported media are to be associated.

Valid MediaClass names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-f *manufacturer*

Specifies the name of the media manufacturer.

If the **-f** *manufacturer* option is specified, the **-b** *batch* option must also be specified.

If the **-f** *manufacturer* option is not specified, the **-b** *batch* option cannot be specified.

-b *batch*

Specifies the manufacturer's batch that contains the media to be entered.

If the **-b** *batch* option is specified, the **-f** *manufacturer* option must also be specified.

If the **-b** *batch* option is not specified, the **-f** *manufacturer* option cannot be specified.

-v Indicates verbose output is desired.

If **-v** is specified, list status information on all media specified on the command.

If **-v** is not specified, report on only those media for which processing failed.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained

in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsimport**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Import request with verbose option specified

vsimport MED003 MED018 MED021 MED030 MED036 **-a** shelf1 **-c** medclasssh1med **-v**

Requests the Media Manager software to import media MED003, MED018, MED021, MED030, and MED036 into the medclasssh1med MediaClass group in the shelf1 archive and to return status on every specified medium.

Output returned:

```
Import 5 of 5 media was successful

   Media [MED003]      no error

   Media [MED018]      no error

   Media [MED021]      no error

   Media [MED030]      no error

   Media [MED036]      no error
```

Successful Import request with verbose option not specified

vsimport MED003 MED018 MED021 MED030 MED03 **-a** shelf1 **-c** medclasssh1med

Requests the Media Manager software to import media MED003, MED018, MED021, MED030, and MED036 into the medclasssh1med MediaClass group in the shelf1 archive and to return status on a medium only if processing for that medium failed.

Output returned:

```
Import 5 of 5 media was successful
```

Error(s) with verbose option specified

vsimport MED003 MED018 MED021 MED030 MED036 **-a** shelf1 **-c** medclasssh1med **-v -f** "MediaMaker ABC" **-b** 1001

Requests the Media Manager software to import media MED003, MED018, MED021, MED030, and MED036 into the medclasssh1med MediaClass group in the shelf1 archive and to return status on every specified medium. "MediaMaker ABC" is the manufacturer of these media and these media were part of batch "1001."

Output returned:

```
Import 3 of 5 media was successful

Error VOL024: error in the list

   Media [MED003]      no error

   Media [MED018]      item already exists

   Media [MED021]      item already exists

   Media [MED030]      no error

   Media [MED036]      no error
```

Error(s) with verbose option not specified

vsimport MED003 MED018 MED021 MED030 MED036 **-a** shelf1 **-c** medclasssh1med **-f** "MediaMaker ABC" **-b** 1001

Requests the Media Manager software to import media MED003, MED018, MED021, MED030, and MED036 into the medclasssh1med MediaClass group in the shelf1 archive and to return status on a medium only if processing for that medium failed. "MediaMaker ABC" is the manufacturer of these media and these media were part of batch "1001."

Output returned:

```
Import 3 of 5 media was successful

Error VOL024: error in the list

   Media [MED018]      item already exists

   Media [MED021]      item already exists
```

Unsuccessful Import request

vsimport MED003 MED018 MED021 MED030 MED036 **-a** BadArchiveName **-c** medclassmed

Requests the Media Manager software to import media MED003, MED018, MED021, MED030, and MED036 into the medclassmed MediaClass group in the BadArchiveName archive and to return status on a medium only if processing for that medium failed.

Output returned:

```
Import of media was unsuccessful

Error VOL013: invalid archive
```

NOTES

Import is a logical operation. Media must be physically entered into an archive by an operator before they are available for general use. A successful Import request results in the media identifier being placed on the receiving archive's Enter list.

Media identifier values must be unique throughout a Media Manager system. Non-unique media identifiers are rejected.

If the Enter fails for the medium to be imported, the medium is placed Intransit.

Media identifiers of media being imported into manual archives may contain alphanumeric and special characters including spaces. However, spaces cannot be used as leading or trailing characters. If media in a manual archive can later be moved into an automated archive, the media identifiers must also conform to any naming restrictions imposed by the automated archive. For example, special characters may not be allowed in media identifiers in the automated archive.

The media type for the media is determined by the media type of the specified MediaClass group.

After the MediaClass capacity is reached, no more media can be imported into the MediaClass group.

The Import command triggers unsolicited status messages from the Media Manager software to the client software.

A pending Import request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (controlc).

SEE ALSO

vsexport(l)

NAME

vslock − Obtains exclusive use of one or more drives.

SYNOPSIS

vslock *driveID...* [**-q** *quantity*] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vslock -p *drivepool* [**-x** *driveID...*] [**-q** *quantity*] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vslock(l) is issued from the command line to request execution of the Media Manager Lock command.

The Lock command is used to obtain exclusive use of a drive or a set of drives.

The lock identifier assigned to the locked drive(s) is returned to the client. This lock identifier must be used by clients on subsequent requests (such as Mount) for those drive(s).

A request to lock a drive that is busy (mounted or previously locked) is queued until the drive becomes available. In addition, intermediate status is returned to indicate the reason a request is being queued.

A Lock command that specifies a drive pool or a list of drives should also indicate the number of drives from the pool/list to be locked. The software selects the drives to lock from within the pool/list according to drive availability.

A Lock command cannot specify a drive pool or a list of drives that spans archives, they must be associated with a single archive.

A Lock command reserves one drive for exclusive use if a quantity is not specified on the command.

The Media Manager software considers only on-line drives as candidates to be locked. If a sufficient number of on-line drives in the same archive are unavailable to satisfy a Lock command, the Lock command fails.

If there is a sufficient number of online drives in the same archive to satisfy a Lock request, but the number of available online drives is not sufficient, the request waits until sufficient drives become available. Partial locks are not set.

OPTIONS

driveID...

Specifies a list of one or more candidate drives to reserve (lock) for exclusive use.

The number of drives that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-p *drivepool*

Specifies the drive pool name from which candidate drive(s) are reserved for exclusive use.

Valid drive pool names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-x *driveid...*

Specifies a list of one or more drive(s) contained in the specified drive pool that are NOT reserved for exclusive use.

The number of drives that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-q *quantity*

Specifies the number of drives to be locked. If the **-q** *quantity* option is not specified, the number of drives to be locked defaults to 1.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vslock**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Lock request that specifies a list of one or more drives

vslock 4 8 12

Requests the Media Manager software to lock (reserve for exclusive use) one of the drives 4, 8, and 12.

Output returned:

```
Lock [1] drives locked with lock id[1719790788]

      1          Drive[4]
```

Successful Lock request that specifies a drive pool and a quantity

vslock -p drvpoolmed **-q** 2

Requests the Media Manager software to lock (reserve for exclusive use) 2 drives from drive pool drvpoolmed.

Output returned:

```
Lock [2] drives locked with lock id[1719790788]

      1          Drive[40]

      2          Drive[41]
```

Unsuccessful Lock request, too many drives requested to be reserved

vslock -p drvpooltwr **-q** 5

Requests the Media Manager software to lock (reserve for exclusive use) 5 drives from drive pool drvpooltwr (drvpooltwr contains only 2 drives.)

Output returned:

```
Lock was unsuccessful

Error VOL122: not enough available drives in pool
```

Unsuccessful Lock request

vslock -p BadDrivePool **-q** 1

Requests the Media Manager software to lock (reserve for exclusive use) 1 drive from drive pool BadDrivePool.

Output returned:

```
Lock was unsuccessful

Error VOL030: invalid drive pool specified
```

NOTES

It is important to keep

Any Mount or Dismount request that contains the proper lock identifier has access to a locked drive.

If a Mount request does not specify a lock identifier for a locked drive, whether the drive is available for use or not, the Mount request waits until the drive is both unlocked and available.

If a Mount request specifies a drive pool, but does not specify a lock identifier, only available unlocked drives in the specified drive pool are considered to satisfy the Mount request. If there are no available unlocked drives in the specified drive pool, the Mount request waits until a drive from the specified drive pool becomes available and unlocked.

A Lock command that is queued and awaiting resources is cancelled via the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (controlc).

An **unlock** command is issued when the client no longer needs drives for exclusive use.

The Lock command does not trigger unsolicited status messages from the Media Manager software.

Intermediate status may be returned if the Lock command is queued.

SEE ALSO

vsdismount(l), **vsmount**(l), **vsunlock**(l)

NAME

vsmanualeject − Logically eject the specified media from the Media Manager system.

SYNOPSIS

vsmanualeject [**-v**] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] *mediaID...*

DESCRIPTION

vsmanualeject(l) is issued from the command line to request execution of the Media Manager Manual Eject command. The Manual Eject command is generally used when media in an archive cannot be accessed by the robot. Although the command can be used for any reason that a medium cannot be accessed such as:

A medium label cannot be read by a robot barcode reader.

A medium has fallen on the archive floor.

A medium becomes jammed in a tape drive.

A medium is unavailable for any reason, such as a nonoperational robot or archive equipment.

After executing the Manual Eject command, the specified media are placed in the intransit state. This state indicates that media are known by Media Manager software but are not physically in an archive.

Unlike the Export Media command, media information is retained in the Media Manager database after the Manual Eject command is executed so these media remain known to the Media Manager software.

The Move media command is recommended for re-entering media that were manually ejected but the Enter command or the bulk load method using the Audit command can also be used.

The Media Manager software must be active in order to run this command however the archive where the media are located must be logically offline within the Media Manager software. Prior to running this command, an operator should have already physically removed the media from the archive.

OPTIONS

-v Indicates verbose output is desired. When this option is used, it will show status information on all media specified on the command otherwise, only those media for which processing failed will be reported.

-I Indicates command line options are to be read from stdin.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

mediaID...
Specifies one or more media to manually eject.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255
The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

Attempt to manually eject 3 media when the archive is online.

vsmanualeject -v 000047 000048 000049

Output returned

```
Manual eject 0 of 3 media was successful
Error VOL024: error in the list
        Medium [000047]             Archive not offline
        Medium [000048]             Archive not offline
        Medium [000049]             Archive not offline
```

Manually eject 3 media when the archive is offline.

vsmanualeject -v 000047 000048 000049

Output returned

```
Manual eject 3 of 3 media was successful
        Medium [000047]             no error
        Medium [000048]             no error
        Medium [000049]             no error
```

NOTES

To cancel the request generated by this command, use the **vscancelreq** command, or send a SIGQUIT signal (control-\).

The command can be aborted by sending a SIGINT signal (control-c), however this will not cancel the request.

SEE ALSO

vsexport(l), **vsenter**(l), **vsmove**(l), **vscancelreq**(l)

NAME

vsmedclasscreate − Creates a media class for the Media Manager system.

SYNOPSIS

vsmedclasscreate [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] [**-C**] [**-x** *highMarkPercent*] [**-n** *notificationComment*] **-c** *capacity* **-t** *mediaType mediaclass*

DESCRIPTION

vsmedclasscreate(l) is issued from the command line to request execution of the Media Manager Media Class Create command. The Media Manager software must be active in order to run this command.

After an archive is successfully configured and registered with Media Manager, at least one media class must be created. Media classes provide logical organization of media into smaller groups.

Media classes have several important characteristics:

When a medium is entered into an archive, it is also entered into a media class. The medium can be in one, and only one, media class.

Once a medium in entered into a media class, it can only be moved into another media class with the Reclassify Media command or a Mount Media command with the reclassify option selected.

Only one media type can exist within a media class.

Media classes can be associated with more than one archive.

OPTIONS

-I Indicates command line options are to be read from stdin.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

-C This option indicates that the Mount Media command can perform mounts by specifying this media class.

-x *highMarkPercent*

The high mark percent indicates a capacity break point that is calculated as a percentage of class capacity. After the high mark capacity is reached, a notification is logged and displayed in the system syslog. An optional notification comment is logged with the syslog message when the **-n** option is specified. The syslog message occurs for every media added to the media class after the

high mark has been exceeded. No more media may be entered into the media class when class capacity is reached, regardless of which archive it is associated with. Syslog notification messages stop whenever the amount of media drops below the high mark. This can be achieved by ejecting media, or using the Reclassify Media command to move media into another media class.

-n *notificationComment*
The notification comment parameter specifies a message that is included in the system logs when the media class fill level exceeds the high mark threshold.

-c *capacity*
The capacity parameter indicates the maximum number of media that can be associated with this media class. Media classes are designed to be associated with multiple archives. However, a media class can be wholly associated with one archive. Therefore, plan media class capacities to cover the expected size of the archives. No software restriction is placed on the media class capacity. Class capacity can exceed the total combined capacity of all archives, or can be as small as one.

-t *mediaType*
The media type parameter identifies the type of media used in the media class. As noted before, only one media type can be specified for a media class. The valid media types can be obtained with the **vsmedtypeqry -a** command.

mediaclass
The media class name parameter is a user-specified name by which the media class is known to the user.

Valid media class names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255
The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

Create a media class called *abc* with a capacity of 100 LTO media

vsmedclasscreate -c *100* **-t** *LTO abc*

Create a media class called *abc* with a capacity of 100 LTO media, which issues a notification with an additional comment of "*time to add media*" when the media class is 75 percent full

vsmedclasscreate -c 100 **-t** LTO **-x** 75 **-n** "TIme to add media" abc

NOTES

To cancel the request generated by this command, use the **vscancelreq** command, or send a SIGQUIT signal (control-\).

The command can be aborted by sending a SIGINT signal (control-c), however this will not cancel the request.

SEE ALSO

vsmount(l), **vsmedclassdelete**(l), **vsreclassify**(l), **vsmedtypeqry**(l), **vscancelreq**(l)

NAME

vsmedclassdelete − Delete a media class for the Media Manager system.

SYNOPSIS

vsmedclassdelete [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*] *mediaclass*

DESCRIPTION

vsmedclassdelete(l) is issued from the command line to request execution of the Media Manager Media Class Delete command. The Media Manager software must be active in order to run this command.

When a media class is no longer needed it can be deleted using the Delete Media Class command. However, the deletion of the media class is only allowed if it not associated with any archives. The archive association can be removed using the **vsarcmedclassdelete** command.

OPTIONS

-I Indicates command line options are to be read from stdin.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

mediaclass

Specifies the name of the media class to be deleted. The media class name parameter is a user-specified name by which the media class is known to the user.

Valid media class names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255

The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

Delete media class called abc

vsmedclassdelete abc

NOTES

To cancel the request generated by this command, use the **vscancelreq** command, or send a SIGQUIT signal (control-\).

The command can be aborted by sending a SIGINT signal (control-c), however this will not cancel the request.

SEE ALSO

vsarcmedclassdelete(l), **vscancelreq**(l)

NAME

vsmedclassqry − Queries for the attributes of a specified MediaClass group or all MediaClass groups.

SYNOPSIS

vsmedclassqry *mediaclass* [**-m**|**-v**] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsmedclassqry -a [**-m**|**-v**] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsmedclassqry(l) is issued from the command line to request execution of the Media Manager MediaClass Query command.

The MediaClass Query command is used to obtain information about one or all MediaClass groups in the Media Manager system. The members of the MediaClass group and any additionally requested information on each medium is returned to the client.

OPTIONS

mediaclass

Specifies a single MediaClass name on which to request information.

Valid MediaClass names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-a Requests information on all MediaClass groups known to the Media Manager system.

-m Requests a list of media identifiers for all media associated with each reported MediaClass group.

-v (verbose) Requests detailed information for all media associated with each reported MediaClass group.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsmedclassqry**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful MediaClass Query with neither -m nor -v specified

vsmedclassqry -a

Requests the Media Manager software to return information on every MediaClass group known to the Media Manager system. No mediaspecific information is requested.

Output returned:

```
-----------------------------------------------

Media Class Query Report   May 14 10:09:11 1993      1

-----------------------------------------------

Media Class: medclassstgusr

-----------------------------------------------

   Media Type:                  USRTYPE
   Capacity:                    20
   Current Fill Level:          2
   High Mark %:                 80%
   Mountable by Class:          Yes
   Notify Comment:              Media exceed high mark
   RPC Option:                  No Callback

Media Class: medclasssh2med

-----------------------------------------------

   Media Type:                  D2M
   Capacity:                    20
   Current Fill Level:          3
   High Mark %:                 80%
   Mountable by Class:          Yes
   Notify Comment:              Media exceed high mark
   RPC Option:                  Enterprise Callback
   Enterprise ID:               3

Media Class: medclasssh2sml

-----------------------------------------------

   Media Type:                  D2S
   Capacity:                    20
   Current Fill Level:          2
   High Mark %:                 80%
   Mountable by Class:          Yes
```

```
   Notify Comment:              Media exceed high mark
   RPC Option:                  Standard Callback
   HostName:                    copper
   Program Number:              1
   Version Number:              1
   Procedure Number:            1
   Protocol:                    TCP
```

Successful Media Class Query request with -m option specified

vsmedclassqry medclasssh1usr -m

Requests the Media Manager software to return detailed media information for every medium in the medclasssh1usr MediaClass group.

Output returned:

```
   ------------------------------------------------

   Media Class Query Report   May 14 09:45:57 1993      1

   ------------------------------------------------

   Media Class: medclasssh1usr

   ------------------------------------------------

      Media Type:  USRTYPE

      Capacity:                 20
      Current Fill Level:       3
      High Mark %:              80%
      Mountable by Class:       Yes
      Notify Comment:           Media exceed high mark
      RPC Option:               No Callback
```

Media ID(s): MED007 MED025 MED040

Successful Media Class Query request with -v option specified

vsmedclassqry medclasssml -v

Requests the Media Manager software to return detailed media information for every medium in the medclasssml MediaClass group.

Output returned:

```
   ------------------------------------------------

   Media Class Query Report   May 14 09:45:57 1993      1

   ------------------------------------------------

   Media Class: medclasssml

   ------------------------------------------------

      Media Type:               D2S
```

```
   Capacity:                20
   Current Fill Level:      8
   High Mark %:             80%
   Mountable by Class:      Yes
   Notify Comment:          Media exceed high mark
   RPC Option:              No Callback

MediaID: MED005

   Media Type:              D2S
   Media Class:             medclasssml
   Assignment:              Free
   Location State:          Archive
   Current Archive:         stage1
   Pending Archive:
   Action State:            None
   Import Date:             May 14 09:30:09 1993
   Last Access:             May 14 09:45:43 1993
   Mount Count:             2
   Move Count:
   Manufacturer:            MediaMaker
   Batch:                   1001

MediaID: MED017

   Media Type:              D2S
   Media Class:             medclasssml
   Assignment:              Allocated
   Location State:          Archive
   Current Archive:         shelf2
   Pending Archive:
   Action State:            None
   Import Date:             May 14 09:28:41 1993
   Last Access:             May 14 09:40:21 1993
   Mount Count:             1
   Move Count:              1
   Manufacturer:
   Batch:

MediaID: MED038

   Media Type:              D2S
   Media Class:             medclasssml
   Assignment:              Free
   Location State:          Archive
   Current Archive:         shelf2
   Pending Archive:         shelf1
   Action State:            Move
   Import Date:             May 14 09:30:09 1993
   Last Access:
   Mount Count:             1
   Move Count:              2
   Manufacturer:
   Batch:
```

```
MediaID: MED051

   Media Type:                 D2S
   Media Class:                medclasssml
   Assignment:                 Free
   Location State:             Intransit
   Current Archive:
   Pending Archive:            shelf1
   Action State:               Import
   Import Date:                May 14 09:30:09 1993
   Last Access:
   Mount Count:                0
   Move Count:                 0
   Manufacturer:
   Batch:
```

Unsuccessful Media Class Query

vsmedclassqry UnknownClass

Requests the Media Manager software to return information on the UnknownClass MediaClass group.

Output returned:

```
Query of Media Class [UnknownClass] was unsuccessful

Error VOL008: item not found
```

NOTES

The Media Class Query command does not trigger unsolicited status messages from the Media Manager software.

A pending Media Class Query request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (controlc).

SEE ALSO

vsmedqry(l)

NAME

vsmedqry − Queries for the attributes of one or more specified media.

SYNOPSIS

vsmedqry *mediaID...* [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsmedqry -a [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsmedqry(l) is issued from the command line to request execution of the Media Manager Media Query command.

The Media Query command is used to obtain information about one, many, or all media in the Media Manager system. The values of the attributes of the media are returned to the client.

OPTIONS

mediaID...

Specifies a list of one or more media to be queried.

The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-a Specifies the **-a** option to indicate information is to be reported on all media known to the Media Manager system.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

OUTPUT

Media Type	The type of media.
Media Class	The media class designated for the tape.
Assignment	Indicates whether the tape is available for mounting. Values are Allocated or Free.

Location State	The location of the tape. Values are Archive, Checkout, or Intransit.
Current Archive	The current library in which the media is located.
Pending Archive	Indicates whether the media is associated with another library.
Action State	Indicates when and how the media is moving.
Import Date	The date and time the media was added to the archive.
Last Access	The date and time when the media was last used.
Mount Count	The number of times the tape has been mounted.
Move Count	The number of times the tape has been moved.
Manufacturer	The name of the media manufacturer, if one was specified during an enter.
Batch	Indicates the manufacturer's batch that contained the media, if one was specified during an enter.
Current State	If the media is in an archive, the state of the archive where the media is located. Values are On-line, Off-line, Diagnostic, or Unavailable.

EXIT STATUS

0	The **vsmedqry**(l) command is successfully processed.
-1	An error is detected by either the CLI software or the API software.
>0	An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Media Query request

vsmedqry -a

Requests the Media Manager software to return information on every medium known to the Media Manager system.

Output returned:

```
------------------------------------------------

Media Query Report  May 14 10:34:30 1993            1

------------------------------------------------

Media ID:  MED001

------------------------------------------------

   Media Type:                  USRTYPE
   Media Class:                 medclassstgusr
   Assignment:                  Free
   Location State:              Archive
   Current Archive:             stage1
   Pending Archive:
   Action State:                None
   Import Date:                 May 13 12:06:15 1993
   Last Access:                 May 13 13:12:09   1993
   Mount Count:                 1
   Move Count:                  2
   Manufacturer:
   Batch:
```

```
   Current State:              On-line

Media ID:  MED015

------------------------------------------------

   Media Type:                 D2M
   Media Class:                medclassmed
   Assignment:                 Free
   Location State:             Intransit
   Current Archive:
   Pending Archive:            shelf1
   Action State:               Move
   Import Date:                May 13 16:25:39 1993
   Last Access:                May 13 16:40:53   1993
   Mount Count:                0
   Move Count:                 0
   Manufacturer:               MediaMaker ABC
   Batch:                      1001
   Current State:

Media ID:  MED027

------------------------------------------------

   Media Type:                 D2M
   Media Class:                medclassmed
   Assignment:                 Free
   Location State:             Archive
   Current Archive:            shelf2
   Pending Archive:            shelf1
   Action State:               Move
   Import Date:                May 12 14:36:10 1993
   Last Access:                May 13 16:13:53 1993
   Mount Count:                0
   Move Count:                 0
   Manufacturer:
   Batch:
   Current State:              On-line

Media ID:  MED039

------------------------------------------------

   Media Type:                 D2M
   Media Class:                medclassmed
   Assignment:                 Allocated
   Location State:             Archive
   Current Archive:            shelf1
   Pending Archive:            shelf2
   Action State:               Move
   Import Date:                May 12 14:36:10 1993
   Last Access:                May 13 14:36:10 1993
   Mount Count:                1
```

```
   Move Count:                 1
   Manufacturer:
   Batch:
   Current State:              On-line
```

Media Query request with errors

vsmedqry MED027 BadMedium MED039

Requests the Media Manager software to return information on media MED027, BadMedium, and MED039.

Output returned:

```
-----------------------------------------------

Media Query Report  May 25 16:06:09 1993           1

-----------------------------------------------

Media ID:  MED027

-----------------------------------------------

   Media Type:                 D2M
   Media Class:                medclassmed
   Assignment:                 Free
   Location State:             Archive
   Current Archive:            shelf2
   Pending Archive:            shelf1
   Action State:               Move
   Import Date:                May 13 16:36:39 1993
   Last Access:                May 24 12:23:25 1993
   Mount Count:                0
   Move Count:                 0
   Manufacturer:
   Batch:
   Current State:              On-line

Media ID:  BadMedium

-----------------------------------------------

   Error:  item not found


Media ID:  MED039

-----------------------------------------------

   Media Type:                 D2M
   Media Class:                medclassmed
   Assignment:                 Allocated
   Location State:             Archive
   Current Archive:            shelf1
   Pending Archive:            shelf2
```

```
Action State:                Move
Import Date:                 May 13 16:36:39 1993
Last Access:                 May 24 12:23:25 1993
Mount Count:                 1
Move Count:                  1
Manufacturer:
Batch:
Current State:               On-line
```

NOTES

A Media Query request can query any media in the Media Manager system. The media specified in a single Media Query request are not required to be located in the same archive.

A pending Media Query request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control\). The request also is aborted by sending the SIGINT signal (controlc).

The Media Query command does not trigger unsolicited status messages from the Media Manager software.

SEE ALSO

vsmedclassqry(l)

NAME

vsmedstateqry − Query for media that are in the Intransit or Unknown states.

SYNOPSIS

vsmedstateqry -i [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsmedstateqry -u [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsmedstateqry(l) is issued from the command line to request execution of the Media Manager Media State Query command.

A client uses the Media State Query command to obtain information about media in the Intransit or Unknown states. The query returns a list of media identifiers.

A medium is considered to be in the Intransit state if it is waiting to be entered into an archive as a result of **Import**, **Mount**, **Move**, **Check-out**, or migration activity processing.

A medium is considered to be the Unknown state if its location is not known as the result of a Manual Eject or a Failed Enter activity.

The **vsmedstateqry**(l) command supports no command-specific options.

OPTIONS

-i Return information about media in the Intransit state.

-u Return information about media in the Unknown state.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The help option takes precedence over any other option entered on a command. When the help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsmedstateqry**(l) command was successfully processed.

-1 An error was detected by either the CLI software or the API software.

>0 An error was detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful media state query

vsmedstateqry -i

Requests the Media Manager software to return a list of all media in the Intransit state.

Output returned:

```
------------------------------------------------

Media State Query Report   08-May-2012 10:54:02

------------------------------------------------

Media ID(s):  MED007  MED025  MED040
```

Unsuccessful media state query

vsmedstateqry -i

Requests the Media Manager software to return a list of all media in the Intransit state.

Output returned:

```
Media State query was unsuccessful

Error VOL008: item(s) not found
```

NOTES

Only media in the Intransit state or Unknown state are queried and reported to the client.

The Media State Query command does not trigger unsolicited status messages from the Media Manager software.

A pending Media State Query request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (control-c).

SEE ALSO

vsmedqry(l)

NAME

vsmedtypeqry − Queries for the attributes of one or more media types.

SYNOPSIS

vsmedtypeqry *mediatype*... [**-Ih**]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsmedtypeqry -a [**-Ih**]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsmedtypeqry(l) is issued from the command line to request execution of the Media Manager Media Type Query command.

The Media Type Query command is used to obtain information about one, several, or all media type(s) in the Media Manager system. The values of the attributes of the media types are returned to the client.

OPTIONS

mediatype...

Specifies a list of one or more media type(s) to be queried.

Either systemspecified media type(s) and/or userdefined media type(s) can be specified.

Valid media type names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

The number of media types that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 32.

-a Indicates that information is to be returned on all media types known to the Media Manager system.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsmedtypeqry**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Media Type Query request

vsmedtypeqry -a

Requests the Media Manager software to return information on every media type known to the Media Manager system.

Output returned:

```
------------------------------------------------

Media Type Query Report  May 24 12:43:18 1993         1

------------------------------------------------

MediaType:  LTO
------------------------------------------------

   Capacity:           400000.00 megabytes
   Number of Sides:    1


Media Type: 3592
------------------------------------------------------------------------------
          Capacity:  300000.00 megabytes
   Number of Sides:  1


Media Type: T10K
------------------------------------------------------------------------------
          Capacity:  500000.00 megabytes
   Number of Sides:  1
```

Media Type Query request with errors.

vsmedtypeqry LTO USRTYPE

Requests the Media Manager software to return information on media types D2S and USRTYPE.

Output returned:

```
------------------------------------------------

Media Type Query Report  May 24 12:43:18 1993         1
```

```
-----------------------------------------------

Media Type: LTO
------------------------------------------------------------------------------
          Capacity:  400000.00 megabytes

   Number of Sides:    1

Media Type:  USRTYPE
------------------------------------------------------------------------------
   Error:  item(s) not found
```

NOTES

The Media Type Query command does not trigger unsolicited status messages from the Media Manager software.

A pending Media Type Query request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (controlc).

SEE ALSO

vsmedclassqry(l), **vsmedqry**(l)

NAME

vsmount − Mounts a medium onto a drive.

SYNOPSIS

vsmount *mediaID...* **-d** *driveID* [**-i**|**-u**] [**-l** *lockID*] [**-Ih**]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsmount -c *mediaclass* **-d** *driveID* **[**-n *newmediaclass***]** [**-i**|**-u**] [**-l** *lockID*] [**-Ih**]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsmount *mediaID...* **-p** *drivepool* [**-x** *driveID...*] [-i|-u**]** [**-l** *lockID*] [**-Ih**]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vsmount -c *mediaclass* **-p** *drivepool* [**-x** *driveID...*] [**-n** *newmediaclass*] [**-i**|**-u**] [**-l** *lockID*] [**-Ih**]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsmount(l) is issued from the command line to request execution of the Media Manager Mount command.

A client uses the Mount command to mount a medium onto a drive.

When issuing a Mount command, the client can specify one of the following.

A single medium
A list of media
A MediaClass group

and either

A specific drive
A drive pool
A drive pool with the exclusion of drive(s)

The **vsmount**(l) command supports a lock identifier parameter. This parameter is required if the drive to be used in satisfying the Mount request has been previously locked with a Lock request. If a Mount request is issued for a locked drive and does not specify a lock identifier, the Mount request waits until the requested drive is unlocked.

The client can also specify how to handle a Mount request that requires an interarchive movement of the medium. This option indicates whether an inter-archive medium movement should or should not be attempted and whether to consider if the source and destination archives are operator-attended or not.

If more than one media and/or a drive pool is specified on the **vsmount**(l) command, the Media Manager software applies a selection algorithm to select a medium/drive pair from the list of media and available drives.

For manual archives, a Mount notice is sent to the operator console for action. The operator is then responsible for confirmation to the Media Manager software when the mount is complete. The Media Manager software returns a return code to the client after the operator action is complete.

A Mount request is queued for later processing if:

The specified/selected drive is busy.
The specified/selected medium is busy.
The selected/specified drive is locked and a lock identifier was not specified on the Mount request.

When a Mount request is queued for later processing, the Media Manager software returns intermediate sta-

tus to the client that specifies the reason the Mount request was queued.

OPTIONS

mediaID...

Specifies a list of one or more media from which the medium to be mounted is to be selected.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-d *driveID*

Specifies a drive that can be used to satisfy the Mount request.

-c *mediaclass*

Specifies a MediaClass name from which the medium to be mounted is to be selected.

-p *drivepool*

Specifies a drive pool from which the drive to satisfy the Mount request can be selected.

Valid drive pool names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-x *driveid...*

Specifies a list of one or more drive(s) contained in the specified drive pool that are to be excluded from consideration when allocating drive(s) to satisfy the Mount request.

The number of drives that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-n *newmediaclass*

Specifies the destination MediaClass group for the reclassification of the medium selected to satisfy the Mount request.

Specifies the **-n** *newmediaclass* option only if the selected medium is to be reclassified to a different MediaClass group.

-i Indicates that a mount requiring an interarchive move fails if either the source and destination archive is marked as unattended.

-u Indicates that a mount requiring an interarchive move is to be performed, regardless of whether either the losing or gaining archive is attended or unattended.

Neither **-i** nor **-u**

Indicates that any mount requiring an interarchive move is to be failed.

-l *lockID*

Specifies the associated lock identifier if a locked drive(s) is/are specified to satisfy the Mount request.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsmount**(1) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Mount request that specifies a list of one or more media and a specific drive

vsmount MED026 **-d** 9

Requests the Media Manager software to mount medium MED026 on drive 9.

Output returned:

```
Mount of Media [MED026] onto Drive [9] was successful
```

Successful Mount request that specifies a MediaClass group and a specific drive

vsmount -c medclassmed **-d** 10 **-n** medclasssh1med

Requests the Media Manager software to mount a medium from MediaClass medclassmed onto drive 10 and to reclassify the selected medium to MediaClass group medclasssh1med.

Output returned:

```
Mount of Media [MED027] onto Drive [10] was successful
```

Successful Mount request that specifies a specific medium and a drive pool with an exclusion drive list

vsmount MED022 **-p** drvpoolusr **-x** 4 14

Requests the Media Manager software to mount medium MED022 onto any drive from drive pool drvpoolusr, excluding drives 4 and 14.

Output returned:

```
Mount of Media [MED022] onto Drive [12] was successful
```

Successful Mount request that specifies a MediaClass name and a drive pool

vsmount -c medclasssml **-p** drvpooltwr

Requests the Media Manager Software to mount a medium from the medclasssml MediaClass group onto a drive from the drvpooltwr drive pool.

Output returned:

```
Mount of Media [MED041] onto Drive [7] was successful
```

Successful Mount request when all specified drives are in use

vsmount MED034 **-d** 12

Requests the Media Manager software to mount medium MED034 onto drive 12. (MED022 is currently mounted on drive 12.)

Output returned:

```
Mount waiting due to busy drive

Mount of Media [MED034] onto Drive [12] was successful
```

Successful Mount request when specified medium is in use

vsmount MED034 **-p** drvpoolmed **-x** 12

Requests the Media Manager software to mount medium MED034 on any drive in drive pool drvpoolmed except drive 12.

Output returned:

```
Mount waiting due to busy medium

Mount of Media [MED034] on Drive [14] was successful
```

Successful Mount request reclassifying the selected medium

vsmount -c medclassstgmed **-p** drvpoolmed **-n** medclassmed

Requests the Media Manager software to mount any medium from MediaClass group medclassstgmed onto any drive in drive pool drvpoolmed, and to change the MediaClass association of the selected medium from medclassstgmed to medclassmed.

Output returned:

```
Mount of Media [MED048] onto Drive [3] was successful
```

Before execution of the **vsmount**(l) request, MED048 was associated with medclassstgmed. After execution of the **vsmount**(l) request, MED048 is associated with medclassmed. The **vsmedqry**(l) command can be used to verify the MediaClass association of a specific medium.

Unsuccessful Mount request requiring an interarchive move

vsmount MED023 **-d** 11

Requests the Media Manager software to mount medium MED023 located in the stage1 archive onto drive 11 associated with the tower1 archive.

Output returned:

```
Media could not be mounted onto drive
```

```
Error VOL110: mount crosses archives
```

Unsuccessful Mount request with unknown media specified

vsmount BadMedium **-d** 5

Requests the Media Manager software to mount medium BadMedium onto drive 5.

Output returned:

```
Media could not be mounted onto drive

Error VOL029: invalid media specified
```

NOTES

The time required to satisfy a specific mount request is dependent on the number of available drives and pending Mount requests.

A drive that is specified in a Mount request may not be the ideal drive on which to mount the specified medium. It may take considerably longer to mount the medium onto a specified drive than if a drive pool is specified.

The **-i** and **-u** options have no effect on a Mount request that does not require an inter-archive medium movement.

If a specified drive was previously locked, the lock identifier assigned to that drive must be supplied before that drive is considered in the selection process.

If a specified or selected drive was previously locked and the Mount request does not specify a lock identifier, Media Manager software returns a message to the client that the selected drive is locked and the Media Manager software is waiting for the drive to become unlocked to continue execution of the command.

If the Mount request specifies a MediaClass group and the **-n** *newmediaclass* option is specified, the reclassify to a different MediaClass group occurs only after the Media Manager software selects the medium to satisfy the Mount request. Only the selected medium is reclassified. The remaining media in the MediaClass group are not reclassified.

If the **-n** *newmediaclass* option is specified, the receiving MediaClass group is checked for compatible media type, as well as for adequate room for another medium (i.e., fill level less than capacity). If either of these conditions is not satisfied, the Mount request fails.

A pending Mount request (waiting for a drive or a medium) is cancelled with the Media Manager Cancel request. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (control-c).

The Mount request fails if no specified drive is online.

If a medium and/or drive is specified and either the medium or drive (or both) are presently in use, the Mount request waits for resources and a message is returned to the client that indicates the reason for the delay.

When specifying a drive pool that contains drives that support different types of media, only those drives that support the media type of the media specified in the Mount request are considered for selection.

If a list of media specified in a Mount request contains media of more than one type, the request fails.

When a medium/drive pairing requires the medium be moved within a single archive system (such as cross-aisle) the mount may take a while to complete. The **-i** and **-u** options do NOT apply to intra-archive system movement.

When a Mount request with groups of media and/or drives is submitted, the Media Manager software attempts to select a drive/medium pair where the drive and medium are associated with the same archive. If multiple drive/medium pai are candidates, the Media Manager software selects a drive/medium pair from the archive with an available drive.

If no drive/medium pair associated with the same archive exists, the Media Manager software then selects a drive/medium pair where the drive and medium are associated with a different archive. If multiple drive/medium pair are candidates, the Media Manager software selects a drive/medium pair from the archive with the largest number of drives. If all archives contain the same number of drives, the Media Manager software then selects a drive/medium pair from the archive with the largest number of media.

When specifying a mount by MediaClass group, and the specified MediaClass group is associated with more than one archive, no inter-archive medium/drive pairing is permitted. The medium selected from the MediaClass group must be in the same archive as the selected drive; otherwise the Mount request fails.

When a medium is ejected (as a result of Export, or Checkout, no check is made to determine if a queued Mount request exists for the ejected medium. As a result, the Mount request remains queued until a drive is freed. At that time, the Mount request fails because the medium is not available. In other words, the request queue is not checked for impact on pending requests each time a resource changes its availability and after a medium/drive pair is identified. The Media Manager software does not attempt re-pairing based on changed availability of resources.

The Mount command triggers unsolicited status messages from the Media Manager software to the client software.

Mount requests may require interarchive medium movement or a Mount request can be queued waiting for an inuse medium or drive. The client may want to increase the *timeout* value or the *retries* value so the CLI **vsmount** request does not timeout while waiting for the Mount request to complete.

If the Mount request allows movement, Media Manager checks the destination archive for available space. If no space is available, the mount fails.

SEE ALSO

vsdismount(l), **vsreclassify**(l)

NAME

vsmove − Moves media from one archive to another.

SYNOPSIS

vsmove [**-Ihiwv**]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

[**-F** *type*] **-a** *archivename mediaID...*

DESCRIPTION

vsmove(l) is issued from the command line to request execution of the Media Manager Move command.

A client uses the Move command to direct the movement of media from one archive to another. Interarchive media movement requires operator intervention. The operator must eject the media from their current archives and enter them into the target archive. The Eject and Enter functionalities are available from the appropriate archive's console display. The Eject and Enter functionalities are not available from the command line.

Upon receipt of a Move request, the software verifies the specified media exist, the target archive supports the media type of the specified media, and there exists an appropriate archive MediaClass association with the target archive. The current archive of each specified medium is commanded to eject the medium. An Eject request, specifying the target archive, is displayed on the archive console of each losing archive. The operator must select and manually eject/remove the media on the list. After a medium is selected for ejection, the target archive displays a corresponding **Enter** request for that medium. The operator must then manually enter the media specified on the list into the target archive.

The client has the option of specifying the **-w** option in the Move request.

When the **-w** option is specified, the Media Manager software waits until processing of the Move command completes before returning status to the client. The client must monitor the media movement (e.g., unsolicited status messages) to know when the media are entered into the target archive system.

When the **-w** option is not specified, the Media Manager software returns a status code to the client after the specified medium (media) are placed on the Eject list of the source archive.

When the media are ejected from the original archive and entered into the target archive, the Media Manager system generates unsolicited status messages if any of the moved media are associated with MediaClass groups that are configured to generate unsolicited communication from the Media Manager software.

The Move command is issued for "homeless" media. A homeless medium is an Intransit medium that has no pending movement activity.

OPTIONS

-a *archivename*

Specifies the name of the archive to which the specified media are to be moved.

Valid archive names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

mediaID...

Specifies a list of one or more media to be moved.
A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.
The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-i Indicates this command is to be processed only if both the source and destination archives are operator-attended.

-w Indicates the Media Manager software waits until the command processing completes before returning status to the client.

If the move requires an interarchive move, Media Manager software waits until the move completes, whether the source and destination archives are attended or unattended. When the

-w option is not specified, final status is returned as soon as move processing begins.

-v Indicates that verbose output is desired.

If **-v** is specified, status is returned on all media specified in the Move request.

if **-v** is not specified, status is returned on only those media that were not successfully processed.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

-F *type* Sets the output format to the specified *type*. Valid values are **TEXT** (default), **XML**, or **JSON**.

TEXT is the "legacy" textual format.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, for example in web services. You may validate the XML output using the XSD (see **fsxsd**(1) for details). See http://en.wikipedia.org/wiki/XML for more information.

JSON (JavaScript Object Notation) is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript programming language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for virtually every programming language. The JSON format is often used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web application, serving as an alternative to XML. See http://json.org for more information.

EXIT STATUS

0 The **vsmove**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Move request, wait until move completes before returning status to client, verbose option specified.

vsmove MED016 MED023 MED044 MED048 **-a** stage1 **-wv**

Requests the Media Manager software to move media MED016, MED023, MED044, and MED048 to archive stage1; to wait until the move completes before returning status to the client; and to return status on every specified medium.

Output returned:

```
Move 4 of 4 media was successful

   Media [MED016]      no error

   Media [MED028]      no error

   Media [MED043]      no error

   Media [MED048]      no error
```

Successful Move request, verbose option not specified

vsmove MED013 MED016 MED028 MED031 **-a** shelf1

Requests the Media Manager software to move media MED013, MED016, MED028, and MED028 to archive shelf1. Since the **-i** option is NOT specified, the move is to complete whether the losing and gaining archives are attended or unattended.

Output returned:

```
Move 4 of 4 media was successful.
```

Error(s) with verbose option specified (target archive is unattended)

vsmove MED013 MED016 MED022 MED034 **-a** shelf1 **-iv**

Requests the Media Manager software to move media MED013, MED016, MED022, and MED034 to the shelf1 archive only if both the source archive(s) and the destination archive are attended and to return status on every specified medium.

Output returned:

```
Move 0 of 4 media was successful

Error VOL024: error in the list

   Media [MED013]   target archive mode is unattended

   Media [MED016]   target archive mode is unattended

   Media [MED022]   target archive mode is unattended
```

```
Media [MED034]   target archive mode is unattended
```

Error(s) with verbose option not specified

vsmove MED003 MED004 MED013 MED028 MED033 MED043 **-a** shelf2

Requests the Media Manager software to move media MED003, MED004, MED013, MED028, MED033, and MED043 to the shelf2 archive.

Output returned:

```
Move 3 of 6 media was successful

Error VOL024: error in the list

   Media [MED003]   item not found

   Media [MED004]   medium already exists in an archive

   Media [MED033]   archive not associated with media
                    class
```

Unsuccessful Move request

vsmove MEDabc MEDxyz **-a** BadArchive

Requests the Media Manager software to move media MEDabc and MEDxyz to the BadArchive archive.

Output returned:

```
Move of media was unsuccessful

Error VOL107: invalid target archive
```

NOTES

Movement of media is between archives, not within archives. Media that is allocated to a Move request is not available for other allocation until the move completes.

If the **-w** option is not specified, status from the Move request indicates only the initial validity of the move. Actual completion of the move can only be traced via callback processing, media querying, or operator monitoring.

The Move command does trigger unsolicited status messages from the Media Manager software.

If the **-w** option is specified on a Move request, the client can increase the *timeout* value or the *retries* value so the CLI **vsmove**(l) command does not time-out while awaiting completion of the Move request.

A pending Mount request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (control-c).

SEE ALSO

vscleareject(l)

NAME

vsping − Pings the Media Manager system.

SYNOPSIS

vsping [**-Ih**]

DESCRIPTION

vsping(l) is issued from the command line to request execution of the Media Manager Ping command.

The **vsping**(l) command allows a user to check the availability of the software (in other words, a means for the client systems to "ping" the Media Manager system.) If the software responds to the **vsping**(l), it is assumed by the client that the software client interface is available and functioning.

The client is not required to use the **vsping**(l) command before sending other commands, but **vsping**(l) is available for clients to verify that software is running.

If no options are specified, **vsping**(l) tries to ping the Media Manager system on the host machine at Media Manager's default program number.

The **vsping**(l) command supports no commandspecific options.

OPTIONS

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

EXIT STATUS

0 The **vsping**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful request, Media Manager software is available

vsping

Requests the Media Manager software to acknowledge receipt of the Ping request.

Output returned:

```
Media Manager is available.
```

Successful request, Media Manager software is not available

vsping

Requests the Media Manager software to acknowledge receipt of the Ping request

Output returned:

```
Media Manager is not available

Error CMD022: could not send command to volume server.
```

NOTES

The client is not required to use **vsping**(l) before issuing other commands.

The **vsping**(l) command is a relatively fast operation.

No Media Manager software status messages are returned in response to the **vsping**(l) command.

The **vsping**(l) command does not trigger unsolicited status messages from the Media Manager software.

SEE ALSO

None

NAME

vspoolcfg − Configures a specified drive pool for the Media Manager system.

SYNOPSIS

vspoolcfg -p *drivepool* **-c** *driveID...* [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vspoolcfg -p *drivepool* **-i** *driveID...* [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vspoolcfg -p *drivepool* **-r** *driveID...* [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vspoolcfg -p *drivepool* **-d** [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vspoolcfg(l) is issued from the command line to request execution of the Media Manager Drive Pool Configure command. The Media Manager software must be active in order to run this command.

When an client uses the **vsmount** command and the specified drive is busy, the command is placed in a command queue or failed. To increase the likelihood of an available drive, the **vsmount** command may specify a collection of drives on which to perform the mount. This collection can be a list of specific drives or a named collection of drives. Such a named collection is defined as a drive pool and is created using this command. Features of a drive pool are:

All defined drives are available for inclusion in a drive pool

A drive does not need to be associated with an archive to be a drive pool member

A drive can be a member of multiple drive pools

Any mix of drive types can be included in a drive pool as long as the subject media types are supported by the Media Manager system

A drive pool may contain zero or more drives.

Besides increasing the likelihood of a successful mount, drive pools can also be used to segregate specific drives to particular user groups. This can reduce the possibility of drive contention between such groups.

OPTIONS

-p *drivepool*

Specifies a single drive pool to be configured.

Valid drive pool names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-c *driveID*

This will create the drive pool specified by the **-p** option and add the *driveID* to that pool.

-r *driveID*

This will remove a drive from the drive pool specified by the **-p** option.

-i *driveID*

This will insert a drive into the drive pool specified by the **-p** option.

-d This will delete the drive pool specified by the **-p** option. All drives must removed from the pool before it can be deleted.

-I Indicates command line options are to be read from stdin.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The command is successfully processed.

255 An error is detected by either the CLI software or the API software.

>0 and <255

The returned exit status corresponds to the error detected by the Media Manager software.

EXAMPLES

Create poolabc drive pool with drive 1

vspoolcfg -p poolabc **-c** 1

Add drive 2 and 3 to the drive pool

vspoolcfg -p poolabc **-i** 2 3

Remove drive 2 from the drive pool

vspoolcfg -p poolabc **-r** 2

Delete the drive pool. Note all the drives must be removed from the pool first.

vspoolcfg -p poolabc **-r** 1 3 **vspoolcfg -p** poolabc **-d**

NOTES

After a pool is created, modified, or deleted, if Tertiary Manager is running, either **fsconfig -R** should be run or Tertiary Manager cycled, in order for Tertiary Manager to refresh it's internal cache.

To cancel the request generated by this command, use the **vscancelreq** command, or send a SIGQUIT signal (control-\).

The command can be aborted by sending a SIGINT signal (control-c), however this will not cancel the request.

SEE ALSO

vspoolqry(l), **vscancelreq**(l) **fsconfig**(l)

NAME

vspoolqry − Queries for information on a specified drive pool or on all drive pools known to the Media Manager system.

SYNOPSIS

vspoolqry *drivepool* [**-v**] [**-Ih**]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

vspoolqry -a [**-v**] [**-Ih**]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vspoolqry(l) is issued from the command line to request execution of the Media Manager Drive Pool Query command.

The Drive Pool Query command is used to report information about one or all drive pools in the Media Manager system. The Drive Pool Query command returns the list of drives contained in each drive pool. Detailed information on each individual drive is obtained by specifying the -v (verbose) option.

OPTIONS

drivepool

Specifies a single drive pool to be queried.

Valid drive pool names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-v Indicates that verbose output is desired.

If **-v** is specified, detailed information about each drive associated with the specified drive pool(s) is returned.

If **-v** is not specified, only the identifiers of the drives associated with the specified drive pool(s) is returned.

-a Specifies the **-a** option to request information on all drive pools known to the Media Manager system.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vspoolqry**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Drive Pool Query request with verbose option specified

vspoolqry drvpoolsml **-v**

Requests the Media Manager to return detailed drive information for the every drive in the drvpoolsml drive pool.

Output returned:

```
------------------------------------------------

Drive Pool Query Report   May 25 15:03:52 1993      1

------------------------------------------------

Drive Pool: drvpoolsml

------------------------------------------------

Drive ID: 4

   Drive Type:                 Magnetic
   Associated Archive:         shelf2
   Current State:              Off-line
   Assignment:                 Free
   Usage Count:                1
   Mount State:                Unmounted
   Mounted Media ID:

Drive ID: 7

   Drive Type:                 Magnetic
   Associated Archive:         tower1
   Current State:              On-line
   Assignment:                 Allocated
   Usage Count:                1
   Mount State:                Mounted
   Mounted Media ID:           MED041
```

```
Drive ID: 13

   Drive Type:                  Magnetic
   Associated Archive:          shelf1
   Current State:               Diagnostic
   Assignment:                  Free
   Usage Count:                 0
   Mount State:                 Unmounted
   Mounted Media ID:
```

Successful Drive Pool Query request with verbose option not specified

vspoolqry -a

Requests the Media Manager to return a list of Drive IDs for every drive pool known to the Media Manager system.

Output returned:

```
-----------------------------------------------

Drive Pool Query Report   May 26 15:45:52 1993      1

-----------------------------------------------



Drive Pool: drvpoolsml

-----------------------------------------------

   Drive ID(s):     2    4    6    7
                    9    11   13   14

Drive Pool: drvpooltwr

-----------------------------------------------

   Drive ID(s):     7    11

Drive Pool: drvpoolstg

-----------------------------------------------

   Drive ID(s):     1    3    9
```

Unsuccessful Drive Pool Query

vspoolqry NoPool **-v**

Requests the Media Manager to return a list of drive identifiers for the NoPool drive pool.

Output returned:

```
Query of drive pool [NoPool] was unsuccessful

Error VOL008: item not found
```

NOTES

The Drive Pool Query command does not trigger unsolicited status messages from the Media Manager software.

A pending Drive Pool Query request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (control-c).

SEE ALSO

None

NAME

vsqrymount − Queries for drives that could be used in a subsequent mount of a specified medium.

SYNOPSIS

vsqrymount *mediaID* [-Ih]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsqrymount(l) is issued from the command line to request execution of the Media Manager Query Mount command.

A client uses the Query Mount command to determine which drives are available for use in a subsequent Mount command for the specified medium. The Query Mount output lists the drives in the order of preference, based (in order of relative importance) upon their availability, proximity to the medium, and usage time.

Upon receipt of the Query Mount request, the software determines which archive contains the specified medium.

If the specified medium is not in an archive, a null list of drives is returned to the client.

If the medium is in an archive, the software determines which drives in that archive (and only that archive) are suitable (based on the medium's type) for mounting the medium.

OPTIONS

mediaID

Specifies the medium for which a list of drives supporting the medium's type is being requested.

A valid medium identifier may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus

1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsqrymount**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Query Mount request

vsqrymount MED024

Requests the Media Manager software to return a list of drives that are candidates to be mounted with MED024.

Output returned:

```
Medium [MED024] can be mounted on the following drives:

      1       Drive [14]

      2       Drive [12]
```

Unsuccessful Query Mount request, specified medium already mounted

vsqrymount MED041

Requests the Media Manager software to return a list of drives that are candidates to be mounted with MED041.

Output returned:

```
Query Mount for medium [MED041] was unsuccessful.

Error VOL043: medium mounted

      1       Drive [11]

      2       Drive [7]
```

Unsuccessful Query Mount request

vsqrymount MED003

Requests the Media Manager software to return a list of drives that are candidates to be mounted with MED003.

Output returned:

```
Query Mount for medium [MED003] was unsuccessful.

Error VOL008: item not found
```

NOTES

The returned list of drives are known to be suitable for mounting the specified medium, but those drives are not available if they are currently in use.

Drives that are not online are not considered suitable for mounting and are, therefore, not returned in response to the query.

If a Query Mount is issued against a medium that is found to be currently mounted, the output to the client includes a message that the medium is mounted, in addition to a list of suitable drives.

If a Query Mount is issued against a medium that is currently allocated for mounting (but has not completed the mount move), the output includes a message that the medium is assigned.

The ordering of the returned drive list is based on the medium's current physical location. Drives that are not mounted are listed before drives that have a medium mounted on them. Consequently, for a mounted medium, the drive on which the medium is currently mounted may not be the first drive on the returned list.

The Query Mount command does not trigger unsolicited status messages from the Media Manager software.

A pending Query Mount request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control\). The request also is aborted by sending the SIGINT signal (control-c).

SEE ALSO

vsmount(l)

NAME

vsreclassify − Changes the MediaClass name of media.

SYNOPSIS

vsreclassify *mediaID..* **-c** *currentmediaclass* **-n** *newmediaclass* [**-v**] [**-Ih**]

[**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsreclassify(l) is issued from the command line to request execution of the Media Manager **reclassify** command.

A client uses the **reclassify** command to change the MediaClass name of one or more media.

Upon receipt of the **reclassify** request, the software verifies that each specified media identifier references a media of the type supported by the target MediaClass group.

If all media are of the appropriate media type, the software verifies that the target MediaClass group is not filled to capacity.

If the target MediaClass group is filled to capacity, the **reclassify** fails and a failure return code is returned to the client.

If the target MediaClass group is not filled to capacity, only as many media as it takes to reach the capacity are reclassified. Any remaining media specified in the **reclassify** command request are not reclassified and have a failure indicator returned to the client.

OPTIONS

mediaID...

Specifies a list of one or more media to be reclassified.

The number of media that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-c *currentmediaclass*

Specifies the MediaClass group with which the specified media are currently associated.

Valid MediaClass names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-n *newmediaclass*

Specifies the new MediaClass group with which the specified media are to be associated.

Valid MediaClass names may contain up to 16 alphanumeric characters, including spaces. Leading and trailing spaces are not permitted.

-v Specifies that the verbose output is desired.

If **-v** is specified, status will be returned on all media specified in the command.

If **-v** is not specified, status is returned on only the media that were not successfully reclassified.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsreclassify**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Reclassify request with verbose option specified

vsreclassify MED002 MED023 MED044 **-c** medclassstgsml **-n** medclasssml **-v**

Requests the Media Manager software to reclassify media MED002, MED023, and MED044 from MediaClass group medclassstgsml to medclasssml and to return status on every specified medium.

Output returned:

```
Reclassify of 3 of 3 media into class [medclasssml] was successful

   Media [MED003]      no error

   Media [MED023]      no error

   Media [MED044]      no error
```

Successful Reclassify request with verbose option not specified

vsreclassify MED002 MED023 MED044 **-c** medclasssml **-n** medclassstgsml

Requests the Media Manager software to reclassify media MED002, MED023, and MED044 from MediaClass group medclasssml to medclassstgsml and to return status on a medium only if processing for that medium was unsuccessful.

Output returned:

```
Reclassify of 3 of 3 media into class [medclassstgsml] was successful
```

Error(s) with verbose option specified

vsreclassify MED013 MED017 MED020 MED032 MED041 BadMedium **-c** medclasssml **-n** medclasssh2sml **-v**

Requests the Media Manager software to reclassify media MED013, MED017, MED020, MED032, MED041, and BadMedium from MediaClass group medclasssml to medclasssh2sml and to return status on every specified medium.

Output returned:

```
Reclassify of 2 of 6 media into class [medclasssh2sml] was successful

Error VOL024:  error in the list

   Media [MED013]   class does not support media type

   Media [MED017]   no error

   Media [MED020]   invalid current class

   Media [MED032]   no error

   Media [MED041]   archive not associated with media
                  class

   Media [BadMedium]  item not found
```

Error(s) with verbose option not specified

vsreclassify MED013 MED017 MED020 MED032 MED041 BadMedium **-c** medclasssml **-n** medclasssh2sml

Requests the Media Manager software to reclassify media MED013, MED017, MED020, MED032, MED041, and BadMedium from MediaClass group medclasssml to medclasssh2sml and to return status on a medium only if processing on that medium was unsuccessful.

Output returned:

```
Reclassify of 2 of 6 media into class [medclasssh2sml] was   successful

Error VOL024:  error in the list

   Media [MED013]   class does not support media type

   Media [MED020]   invalid current class

   Media [MED041]   archive not associated with media
                  class

   Media [BadMedium]  item not found
```

Unsuccessful Reclassify request

vsreclassify MED042 **-c** medclassmed **-n** BadClass

Requests the Media Manager software to reclassify the medium MED042 from MediaClass group medclassmed to BadClass and to return status on a medium only if processing on that medium was unsuccessful.

Output returned:

```
Reclassify of media into class [BadClass] was unsuccessful
```

```
Error VOL147: invalid target class
```

NOTES

The **reclassify** command cannot be cancelled.

Pending Mount requests are not affected by the reclassification of media.

If the capacity of the target MediaClass group is exceeded by the reclassification, only as many media as necessary to reach capacity are reclassified; the reclassification of any remaining media fails.

The capacity of an archive media class is a soft limit. If the capacity of an archive media class is exceeded, the entire **reclassify** request is processed unless the capacity of the associated MediaClass group is reached. When the capacity of the archive media class is reached, applicable High Mark processing is initiated.

An attempt to reclassify a medium into its current MediaClass group fails.

If reclassifying a medium places it in a MediaClass group that does not have the medium's present location as a preferred location, the medium is NOT moved just to place it into a preferred area. Later, if the medium is mounted then dismounted, or ejected then entered, an attempt is made to place the media in a preferred location as defined by the target archive media class.

If a medium to be reclassified is in an archive, the target MediaClass group must be associated with that archive.

A medium that does not reside in an archive can be reclassified.

The **reclassify** command triggers unsolicited status messages from the Media Manager software.

SEE ALSO

vsmount(l), **vsmedclassqry**(l)

NAME

vsrequestqry - Queries for information about a specified request.

SYNOPSIS

vsrequestqry *requestID* [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsrequestqry(l) is issued from the command line to request execution of the Media Manager Request Query command.

A client uses the **vsrequestqry**(l) command to obtain information about an outstanding software request. The client must specify the Media Manager software assigned request identifier of the request being queried.

Upon receipt of a Request Query request, the software searches its request queue for the specified request identifier. If the specified request is not found, status is returned to the client that indicates a non-existent request. If the request is found, the attribute values of the request are returned to the client.

OPTIONS

requestID

Specifies the Media Manager identifier of the request to be queried.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified. The default host name is the host name of the computer where the CLI command is issued.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsrequestqry**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Request Query

vsrequestqry 1014402136

Requests the Media Manager software to return status on request 1014402136

Output returned:

```
------------------------------------------------

Request Query Report   May 24 12:43:18 1993         1

------------------------------------------------

Request ID:  1014402136

------------------------------------------------

   Request Type:                Move
   Priority:                    15
   Time:                        May 24 12:41:54 1993
   Current State:               Executing
```

Unsuccessful Request Query

vsrequestqry 1014402137

Requests the Media Manager software to return status on request 1014402137

Output returned:

```
------------------------------------------------

Request Query Report   May 24 12:54:43 1993         1

------------------------------------------------

   Error:  item not found
```

NOTES

The client must know the request identifier to be queried.

After execution of a request completes, a relatively short period of time exists where the request shows a state of complete. Afterwards, all knowledge of the request is removed from the Media Manager system and any subsequent queries for the command fail.

The Request Query command returns information for only one request per execution.

The Request Query command does not trigger unsolicited status messages from the Media Manager software.

A pending Request Query request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control-\). The request also is aborted by sending the SIGINT signal (control-c).

SEE ALSO

None

NAME

vsunlock - Releases exclusive use of one or more drives.

SYNOPSIS

vsunlock *driveID...* **-l** *lockID* [-v] [**-Ih**] [**-P** *priority*] [**-R** *retries*] [**-T** *timeout*]

DESCRIPTION

vsunlock(l) is issued from the command line to request execution of the Media Manager unlock command.

The **unlock** command is used to release exclusive use of a drive or a set of drives. The list of drive(s) to be unlocked and the assigned lock identifier for those drives must be specified.

A client needing to release a subset of locked drives, locked via a single lock command, can issue multiple **unlock** requests. Each unlock request can specify a subset of the drives held by the client. The software releases only those drives specified in the unlock request.

OPTIONS

driveID...

Specifies a list of one or more drives to be released (unlocked) from exclusive use.

The number of drives that can be specified is restricted by the CLI software. Currently, the maximum allowed number is 64.

-l *lockID*

Indicates the lock identifier assigned to the specified, locked drive(s).

-v Indicates that verbose output is desired.

If the **-v** option is specified, status is returned on every drive specified in the **vsunlock**(l) command.

If **-v** is not specified, status is returned on only those drives that were not successfully unlocked.

-I Indicates command line options are to be read from stdin.

The **-I** option provides the capability for the client to enter command options on the command line, in a text file, or a combination of both. When the **-I** option is specified, command options are first read from the command line. The Media Manager software then reads any options contained in the specified text file.

-h Requests help for the entered command.

The Help option returns the usage for the entered command, a copyright notice, and the version number of the CLI software.

The Help option takes precedence over any other option entered on a command. When the Help option is specified, no command line processing is performed.

An exit code of 0 is returned to the client when the Help option is specified. The default host name is the host name of the computer where the CLI command is issued.

-P *priority*

The execution priority of the entered command.

Assignable priority values are restricted to a range from 1 (highest) to 32 (lowest) inclusive.

The default priority value is 15.

-R *retries*

The number of retries the CLI software attempts if a timeout is returned by the API software.

The default retries value is 3.

-T *timeout*

The amount of time (in seconds) the API software waits for status from the Media Manager software before returning a timeout to the CLI software. Total wait time for a command is (retries plus 1) multiplied by time-out value.

The default time-out value is 120 seconds.

EXIT STATUS

0 The **vsunlock**(l) command is successfully processed.

-1 An error is detected by either the CLI software or the API software.

>0 An error is detected by the Media Manager software. The returned exit code corresponds to the error code given by the Media Manager software.

EXAMPLES

Successful Unlock request with verbose option specified

vsunlock -v 4 8 12 -l 1719790788

Requests the Media Manager software to unlock drives 4, 8, and 12, all of which have an assigned lock identifier of 1719790788, and to return status on every specified drive.

Output returned:

```
Unlock [3] drives unlocked with lock id [1719790788]

      Drive [4]          no error
      Drive [8]          no error
      Drive [12]         no error
```

Successful Unlock request without verbose option specified

vsunlock 4 8 -l 1719790788

Requests the Media Manager software to unlock drive 4 and 3, with an assigned lock identifier of 1719790788, and to return status on a drive only if the Unlock request for that drive was unsuccessful.

Output returned:

```
Unlock [2] drives unlocked with lock id [1719790788]
```

Unsuccessful Unlock identifier with verbose option

vsunlock -v 40 41 42 43 -l 1719790788

Requests the Media Manager software to unlock drives 40, 41, 42, and 43, three of which have an assigned lock identifier of 1719790788, and to return status on every specified drive.

Output returned:

```
Unlock [3] drives unlocked with lock id [1719790788]

Error VOL024: error in the list

      Drive [40]         invalid lock id
      Drive [41]         no error
      Drive [42]         no error
      Drive [43]         no error
```

Unsuccessful Unlock identifier without verbose option

vsunlock 40 41 42 43 -l 1719790788

Requests the Media Manager software to unlock drives 40, 41, 42, and 43, three of which have an assigned lock identifier of 1719790788 and to return status on a drive only if the Unlock request for that drive was unsuccessful.

Output returned:

```
Unlock [3] drives unlocked with lock id [1719790788]

Error VOL024: error in the list

        Drive [40]          invalid lock id
```

NOTES

The client can release a subset of drives locked by a single Lock request.

The Media Manager software fails an Unlock request for a drive if the lock identifier specified in the Unlock request does not match the lock identifier assigned to the drive.

The Unlock command does not trigger unsolicited status messages from the Media Manager software.

A pending Unlock request is cancelled with the Media Manager Cancel command. The Media Manager Cancel command is issued from the command line by sending a SIGQUIT signal (control\). The request also is aborted by sending the SIGINT signal (control-c).

SEE ALSO

vslock(l)

NAME

vsxsd - generate XSD files for commands which output XML.

SYNOPSIS

vsxsd [**-Bhl**] *command*

DESCRIPTION

vsxsd(l) is issued from the command line to request execution of the Media Manager xsd command. A client uses the vsxsd command to generate an XSD. An XSD (XML Schema Definition) and is often used to validate XML.

OPTIONS

-h Print the **vsxsd**(l) usage (help)

-l Print the commands that have XSD output.

command

Command to generate the XSD specification for.

EXAMPLE

You can use the **vsxsd**(l) command and the Linux command **xmllint**(1) to validate XML output against the XSD schema.

For example, these commands create XML and XSD output and then validates the XML against the XSD using the Linux **xmllint**(1) command.

```
> vsmove -F xml -a tape T00001 > vsmove.xml
> vsxsd vsmove > vsmove.xsd
> xmllint --schema vsmove.xsd vsmove.xml
```

SEE ALSO

xmllint(1)

NAME

/usr/adic/wsar_agent/bin/wsar_agent − WSAR Agent on StorNext MDC systems

SYNOPSIS

wsar_agent

DESCRIPTION

The WSAR agent application runs on StorNext MDC systems and does the work of scheduling tasks associated with requests submitted by the StorNext web services. The agent is started by the *wsar_agent_control* script on StorNext MDC systems, as directed by the */usr/adic/wsar_agent/config/wsar_agent.cfg* configuration file. The worker should not be started by hand.

The WSAR agent process writes log messages to a file in the */usr/adic/wsar_agent/logs* directory. The level of logging is configurable in the *wsar_agent.cfg* file.

FILES

/usr/adic/wsar_agent/log/job_manager.log

/usr/adic/wsar_agent/config/wsar_agent.cfg

SEE ALSO

wsar_agent.cfg(5), **wsar_agent_control**(1M), **wsar_client**(1M)

NAME

wsar_agent.cfg − Web Service Async Request Agent configuration file

SYNOPSIS

/usr/adic/wsar_agent/config/wsar_agent.cfg

DESCRIPTION

The Web Service Async Request Agent configuration file *wsar_agent.cfg* enumerates the operating parameters of the WSAR Agent process. The WSAR agent reads the file on startup to set its parameters, all of which remain in effect throughout the life of the process. Changes to any of the parameters requires a restart the WSAR Agent, in order for them to take effect.

SYNTAX

The **wsar_agent.cfg** file exists in the /usr/adic/wsar_agent/config directory on the MDC. It defines the address and credentials for database access, WSAR agent logging preferences, and the thread pool size for each of the WSAR agent job types.

Note: All network names and numerical IP addresses must be resolvable, either by DNS or by the /etc/hosts file.

The total of worker-job-fast threads and worker-job-slow threads cannot exceed 1000. When **wsar_agent** is started, if the total exceeds this value then both parameters will be set to 500.

NOTE: The **wsar_agent.cfg** file is not modified.

Following are the entries from the wsar_agent.cfg.proto file with an explanation of each.

[**job-retention=30d**] Length of time to retain a job status in the database. Jobs whose completion time is past the specified retention time will be automatically removed from the database. The retention time can be set in units of minutes, hours, or days. To specify minutes put an 'm' suffix after the value, to specify hours put an 'h' suffix, or to specify days, use a 'd' suffix. A retention value with no suffix defaults to units of minutes.

[**max-connections=30**] The maximum number of simultaneous client connections allowed.

[**log-level=5**] The internal WSAR Agent logging level. These correspond to the levels described in the syslog.3 man page. Values are 0-7 and the default is 5 (LOG_NOTICE). A higher value includes log messages from lower levels.

[**log-size=100**] Maximum size of the log file in MBytes before rolling. The range is 10..1024 MBytes, and the default is 100 MBytes.

[**job-queue-host=localhost**] The hostname or IP address of the local WSAR agent messaging service used to communicate with the webservice client.

[**job-queue-port=12234**] The port number on which the WSAR agent messaging service listens.

[**database-host=localhost**] The host name or IP address of the database server.

[**database-port=3307**] The port number on which the database server listens.

[**database-name=wsar_agent**] The database name.

[**database-user=wsar_agent**] The database login user name.

[**database-password=wsar_agent**] The database login password.

[**worker-job-fast=4**] The number of worker threads to allocate to process fast running web service requests.

[**worker-job-slow=4**] The number of worker threads to allocate to process slow running web service requests.

[**worker-job-callback=1**] The number of worker threads to allocate to process job completions and optional callback.

[**worker-job-cancel=1**] The number of worker threads to allocate to process job cancellation requests.

[**scratch-pool-size-fast=100**] The number of entries in the fast request pool. This pool contains jobs that will be sent to idle fast worker threads. It is recommended that this value be larger than **worker-job-fast** setting.

[**scratch-pool-size-slow=100**] The number of entries in the slow request pool. This pool contains jobs that will be sent to idle slow worker threads. It is recommended that this value be larger than **worker-job-slow** setting.

EXAMPLE

Refer to the */usr/adic/wsar_agent/config/wsar_agent.cfg.proto* sample configuration file.

FILES

/usr/adic/wsar_agent/config/wsar_agent.cfg

SEE ALSO

wsar_agent(1M), **wsar_client**(1M), **wsar_agent_control**(1M)

NAME

/usr/adic/wsar_agent/bin/wsar_agent_control − Initialization script for WSAR Agent on MDCs

SYNOPSIS

wsar_agent_control { *start* | *[* **-t** *seconds] stop* | *restart* | *status* }

DESCRIPTION

The WSAR Agent worker process that runs on StorNext MDCs is controlled by the StorNext MDC startup mechanism.

This initialization script responds to the normal **start**, **stop**, **restart**, and **status** commands. This allows the WSAR Agent to be started and stopped any time the StorNext MDC is HA Primary, or not configured for HA and running StorNext.

Starting WSAR Agent

The **wsar_agent_control start** command will start the WSAR Agent only after StorNext Storage Manager is fully operational. It is automatically executed at the end of startup of StorNext.

Stopping WSAR Agent

The **wsar_agent_control** [**-t** *seconds*] *stop* allows workers' in-progress tasks to complete. When the **-t** *seconds* option is used, the tasks are allowed a maximum of *seconds* time to stop, at which point the worker processes are forced to stop. This is not recommended as it can result in some transfer-path reservations not being released until the reservations have been manually cleared from the WSAR.

The **wsar_agent_control** *stop* command is automatically executed before stopping StorNext.

SEE ALSO

wsar_agent(1M), **wsar_client**(1M), **wsar_agent.cfg**(5)

NAME

/usr/adic/wsar_agent/bin/wsar_client − Web Service Asynchronous Request client utility

SYNOPSIS

wsar_client [**-f**] [**-p** *priority*] [**-w**] *command* [arg [arg ...]]

wsar_client -a *action*

wsar_client -c *job_id* [**-w**]

wsar_client {**-l** *limit*|**-L**} [**-s** *state*] [**-t** *types*]

wsar_client -h

DESCRIPTION

The Web Service Asynchronous Request (WSAR) client utility runs on StorNext MDC systems and can be used to submit jobs, send control messages, or send status query requests to the WSAR agent.

The client utility uses the configuration settings found in */usr/adic/wsar_agent/config/wsar_agent.cfg* for agent socket connection information.

The WSAR client utility writes log messages to the log file found in the */usr/adic/wsar_agent/logs* directory. The level of logging is configurable in the *wsar_agent.cfg* file.

OPTIONS

−a *action*

The **−a** option can be used to send a control message to the WSAR agent. Valid actions are start, suspend, resume, or ping.

−c *job_id*

The **−c** option can be used to check the status of the job identified by the specified *job_id*.

−f The **−f** option is used to assign the job to the short-run job queue. Jobs are assigned to the long-run queue by default.

−h The **−h** option is used to display the command usage for this utility.

−l *limit* The **−l** option is used to list entries on the job queue. A limit value of zero lists all entries.

−L The **−L** option is used to list the first 10 entries on the job queue.

−p *priority*

The **−p** option can be used to set a priority on a submitted job. Valid priorities are between 1 and 9.

−s *state* The **−s** can be used to limit the entries displayed with the **-l or -L** option to those matching the specified comma-separated list of job *states*. Valid state values are: queued, running, completed, error, aborting, or aborted. The **-s** option is only valid with the **-l** option. Jobs in any state are listed, first by priority and then by job ID.

−t *types*

The **−t** option can be used to limit the entries listed with the **-l or -L** option to those matching the specified comma-separated list of job *types*. Valid job type values are fast, slow, abort, or priority. The default is to list all job types. The **-t** option is only valid with the **-l or -L** option.

−w The **−w** option can be used to wait for a job to complete before returning the status. The **-w** option is only valid when submitting a job or with the **-c** option.

NOTES

Jobs submitted through this interface are not checked for syntactic correctness. Errors may not be apparent until job results are listed.

FILES

/usr/adic/wsar_agent/logs/wsar_agent.log

/usr/adic/wsar_agent/config/wsar_agent.cfg

SEE ALSO

wsar_agent.cfg(5), **wsar_agent**(1M), **wsar_agent_control**(1M)